```
# Forcing shinyapps.io to use latest versions of packages
# https://koffi-sessie.shinyapps.io/test_keystat/
library(devtools)

if (!requireNamespace("shiny", quietly = TRUE)) {
  install.packages("shiny")
}
if (!requireNamespace("shinydashboard", quietly = TRUE)) {
  install.packages("shinydashboard")
}

# if (!requireNamespace("glue", quietly = TRUE)) {
#   install.packages("glue")
#    # install.packages("https://cran.r-project.org/src/contrib/Archive/glue/glue_1.6.2.tar.gz", repos = NULL, type = "source")
#
#   # devtools::install_version("glue", version = "1.6.2", repos = "http://cran.us.r-project.org")
#
# }

# Pour obtenir les ratings
# https://www.knowesg.com/company-esg-ratings

# Ici je peux obtenir des données
#           https://www.spglobal.com/spdji/en/indices/equity/sp-500/?currency=USD&returntype=P-#overview

# https://www.morningstar.com/company/about-us
# https://www.morningstar.com/company/about-us
```

Highchart

https://www.datanovia.com/en/lessons/highchart-interactive-line-plot-in-r/

Yahoo finance

https://github.com/larroy/yfinanceng/blob/d437c5900b16b6725c349625da873c1e48b16835/yfinanceng/ticker.py#L38

library(alphavantager)

av_api_key("I844IIXKI2QIFXIR")

plotly

https://plotly.com/r/time-series/

https://plotly.com/r/candlestick-charts/

3D Scatter Plots in R

https://plotly.com/r/3d-scatter-plots/

https://stt4230.rbind.io/tutoriels_etudiants/hiver_2018/plotly/

Bon tableau pour les commentaires sur le sharpe Ratio ESG

https://www.etmoney.com/learn/mutual-funds/sharpe-ratio-formula-calculation-and-importance-limitations/

Analyse de la contribution de chaque titre à la volatilité

https://rviews.rstudio.com/2017/09/13/asset-contribution-to-portfolio-volatility/

Analyse technique - Leçon 1 - Les lignes de tendances

https://www.abcbourse.com/apprendre/11_lecon1.html

```
# Vente et achat
# https://rpubs.com/TPeter2000/1024236

# https://bookdown.org/kochiuyu/technical-analysis-with-r-second-edition2/technical-indicators.html

# Analyse technique
# https://business-science.github.io/tidyquant/articles/TQ02-quant-integrations-in-tidyquant.html

# Portfolio optimization
# https://www.codingfinance.com/post/2018-05-31-portfolio-opt-in-r/

# Add some plot in datatable
# https://www.r-bloggers.com/2019/07/plotting-the-columns-of-a-datatable/

# Display a bar chart embedded into a data table (DT) in an rshiny app
# https://towardsdatascience.com/top-7-packages-for-making-beautiful-tables-in-r-7683d054e541

library(bslib)
library(caTools)
library(MCMCpack)
library(rportfolio) #pour calculer le ratio d'information simple, alpha de jensen et autre
library(sparkline)
library(tibble)
library(tidyquant)
# devtools::install_github("Fredysessie/Financial.data")
# library(Financial.data)
library(shiny)
```

```
# library(IntroCompFinR)
library(fPortfolio)
library(qrmdata)
library(qrmtools)
# install.packages("IntroCompFinR", repos="http://R-Forge.R-project.org")
library(waiter)
library(ftExtra)
library(moments)
library(GGally)
library(readr)
library(jsonlite)
library(slickR)
# library(Euronext)
library(BRVM)
library(glue)
library(echarts4r)
library(prophet)
library(fBasics)
library(nortest)
library(urca)
library(shinydashboard)
library(plotly)
library(ggplot2)
library(shinyWidgets)
library(shinythemes)
library(datasets)
library(xts)
library(fontawesome)
library(quantmod)
```

```
library(bsicons)
library(gsheet)
library(httr)
library(stringr)
library(dplyr)
library(stringi)
library(V8)
library(tidyr)
library(rvest)
library(formattable)
library(data.table)
library(kableExtra)
library(rlang)
library(lubridate)
library(purrr)
library(DT)
library(highcharter)
library(tidyverse)
library(tseries)
library(timeSeries)
library(PolynomF)
library(forecast)
library(highr)
library(chromote)
library(htmltools)
library(quadprog)
library(httr2)
library(thematic)
library(magrittr)
```

```
library(colourpicker)
library(psych)
library(vtable)
library(RCurl)
library(XML)
library(shinydashboardPlus)
library(bs4Dash)
library(xml2)
library(viridis)
library(treemap)
library(readxl)
library(rugarch)
library(rmgarch)
library(shinycssloaders)
library(ichimoku)
library(PerformanceAnalytics)
library(PortfolioAnalytics)
library(shinycustomloader)
library(shinyjs)
library(timetk)
# remotes::install_github("deepanshu88/summaryBox")
library("summaryBox")
library(optimx)
library(forcats)
library(RColorBrewer)
library(akima)

# Version avec chromote
```

```
EN_Ticker_hcData <- function(ticker, from = NULL, to = Sys.Date(), stock_type = 'Eq_Ind', escape = FALSE) {


  Ready.HcData.func <- function(ticker, escape = FALSE, from, to, stock_type) {

    if (escape) {

      url_ <- paste0("https://live.euronext.com/en/product/equities/", toupper(ticker))

      chart_max <- glue::glue("https://live.euronext.com/en/intraday_chart/getChartData/{basename(url_)}/max")


      # Initialisation de ChromoteSession

      b <- ChromoteSession$new()

      on.exit(b$close())  # Fermer la session à la fin


      tryCatch({

        # Navigation vers l'URL

        b$Page$navigate(url_)


        # Attente que la page soit complètement chargée

        for (i in 1:30) {  # Limité à 30 essais (60 secondes avec un intervalle de 2 s)

          is_loaded <- b$Runtime$evaluate("

            document.readyState === 'complete'

          ")$result$value


          if (is_loaded) break

          Sys.sleep(2)

        }


        # Vérification du chargement complet

        if (!is_loaded) {
```

```
  warning("La page n'a pas pu être complètement chargée.")
  return(NULL)
 }

 # Injection de la fonction JavaScript pour récupérer les données via fetch_chart_data
 b$Runtime$evaluate("
  function fetch_chart_data(url) {
   return fetch(url)
    .then(response => response.json())
    .then(json => ajax_secure_dataFilter(json, 'json', false));
  }
 ")

 # Utilisation de fetch_chart_data pour récupérer les données
 chart_data <- b$Runtime$evaluate(glue('fetch_chart_data("{chart_max}")'), awaitPromise = TRUE, returnByValue = TRUE)

 # Extraction des données
 if (!is.null(chart_data$result$value)) {
  raw_data <- chart_data$result$value

  # Convertir la liste JSON en dataframe
  df <- data.frame(
   Date = sapply(raw_data, function(x) x$time),
   Price = sapply(raw_data, function(x) x$price),
   Volume = sapply(raw_data, function(x) x$volume),
   stringsAsFactors = FALSE
  )
```

```
    # Nettoyer les données
    df <- df %>%
     dplyr::mutate(
       Date = as.POSIXct(Date, format = "%Y-%m-%d"),
       Price = as.numeric(Price),
       Volume = Volume
       # Volume = as.integer(Volume)
     ) %>%
     dplyr::filter(Date >= as.POSIXct(from) & Date <= as.POSIXct(to))



    df$Date <- as.Date(df$Date)


    # print('head(df)')
    # print(head(df))
    # print('tail(df)')
    # print(tail(df))


    return(df)
   } else {
    warning("Aucune donnée n'a été récupérée via fetch_chart_data.")
    return(NULL)
   }
  }, error = function(e) {
   warning("Erreur lors de la navigation ou de la récupération des données : ", conditionMessage(e))
   return(NULL)
  })
 } else {
  warning("La récupération des données est désactivée (escape = FALSE).")
```

```
    return(NULL)
  }
 }

 # Validation des dates
 from <- as.Date(ifelse(is.null(from), "1970-01-01", strftime(from, "%Y-%m-%d")))
 to <- as.Date(strftime(to, "%Y-%m-%d"))

 if (from > to) stop("'from' doit être inférieur ou égal à 'to'")

 # Appeler la fonction principale
 Ready.HcData.func(ticker, escape, from, to, stock_type)
}

# Fonction utilisant httr2
# EN_Ticker_hcData <- function(ticker, from = NULL, to = Sys.Date(), stock_type = 'Eq_Ind', escape = FALSE) {
#
#   Ready.HcData.func <- function(ticker, escape = FALSE, from, to, stock_type) {
#     if (escape) {
#       url_ <- paste0("https://live.euronext.com/en/product/equities/", toupper(ticker))
#                                                                   chart_max <- glue::glue("https://live.euronext.com/en/intraday_chart/getChartData/{basename(url_)}/max")
#
#       tryCatch({
#         # Effectuer une requête GET sur l'URL
#         response <- httr2::request(chart_max) %>%
#           httr2::req_perform() %>%
#           httr2::resp_body_json()
```

```
#
#       # Vérification des données récupérées
#       if (!is.null(response)) {
#         # Convertir la liste JSON en dataframe
#         df <- data.frame(
#           Date = sapply(response, function(x) x$time),
#           Price = sapply(response, function(x) x$price),
#           Volume = sapply(response, function(x) x$volume),
#           stringsAsFactors = FALSE
#         )
#
#         # Nettoyer les données
#         df <- df %>%
#           dplyr::mutate(
#             Date = as.POSIXct(Date, format = "%Y-%m-%d"),
#             Price = as.numeric(Price),
#             Volume = Volume
#           ) %>%
#           dplyr::filter(Date >= as.POSIXct(from) & Date <= as.POSIXct(to))
#
#         df$Date <- as.Date(df$Date)
#
#         return(df)
#       } else {
#         warning("Aucune donnée n'a été récupérée via l'API.")
#         return(NULL)
#       }
#     }, error = function(e) {
#       warning("Erreur lors de la récupération des données : ", conditionMessage(e))
```

```
#       return(NULL)
#     })
#   } else {
#     warning("La récupération des données est désactivée (escape = FALSE).")
#     return(NULL)
#   }
# }
#
#  # Validation des dates
#  from <- as.Date(ifelse(is.null(from), "1970-01-01", strftime(from, "%Y-%m-%d")))
#  to <- as.Date(strftime(to, "%Y-%m-%d"))
#
#  if (from > to) stop("'from' doit être inférieur ou égal à 'to'")
#
#  # Appeler la fonction principale
#  Ready.HcData.func(ticker, escape, from, to, stock_type)
# }



# Fonction EN_Ticker_hcData utilisant httr
# EN_Ticker_hcData <- function(ticker, from = NULL, to = Sys.Date(), stock_type = 'Eq_Ind', escape = FALSE) {
#  if (!escape) {
#    warning("La récupération des données est désactivée (escape = FALSE).")
#    return(NULL)
#  }
#
#  # Validation des dates
#  from <- as.Date(ifelse(is.null(from), "1970-01-01", strftime(from, "%Y-%m-%d")))
```

```
#   to <- as.Date(strftime(to, "%Y-%m-%d"))
#
#   if (from > to) stop("'from' doit être inférieur ou égal à 'to'")
#
#   tryCatch({
#     # Construction des URLs
#     base_url <- paste0("https://live.euronext.com/en/product/equities/", toupper(ticker))
#
#     print("base_url")
#     print(base_url)
#
#     # Première requête pour obtenir les informations nécessaires
#     response <- httr::GET(base_url)
#     if (status_code(response) != 200) {
#       warning("Impossible d'accéder à l'URL de base")
#       return(NULL)
#     }
#
#     print("status_code(response)")
#     print(status_code(response))
#
#     # Construction de l'URL pour les données historiques
#                 chart_url <- paste0("https://live.euronext.com/en/intraday_chart/getChartData/",
toupper(ticker), "/max")
#
#     print("chart_url")
#     print(chart_url)
#
#     # En-têtes HTTP nécessaires
```

```
#   headers <- c(
#     "User-Agent" = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36",
#     "Accept" = "application/json",
#     "Referer" = base_url
#   )
#
#   # Requête pour obtenir les données
#   response <- httr::GET(chart_url, add_headers(.headers = headers))
#
#   if (status_code(response) != 200) {
#     warning("Impossible de récupérer les données historiques")
#     return(NULL)
#   }
#
#   # Conversion du contenu JSON en liste R
#   raw_data <- jsonlite::fromJSON(rawToChar(response$content))
#
#   # Création du dataframe
#   df <- data.frame(
#     Date = as.Date(sapply(raw_data, function(x) x$time)),
#     Price = as.numeric(sapply(raw_data, function(x) x$price)),
#     Volume = sapply(raw_data, function(x) x$volume),
#     stringsAsFactors = FALSE
#   )
#
#   # Filtrage des données selon la période demandée
#   df <- df %>%
#     dplyr::filter(Date >= from & Date <= to)
```

```
#
#    print("str df")
#    print(str(df))
#
#    return(df)
#
#  }, error = function(e) {
#    warning("Erreur lors de la récupération des données : ", conditionMessage(e))
#    return(NULL)
#  })
# }


# Fonction pour abréger le texte avec remplacements
abbreviate_text <- function(text) {
 # Remplacements des mots
 replacements <- c(
  "Index" = "Ind.", "Alternative" = "Alt.","Weighting" = "W_", "Weight" = "W.", "Decrement" = "Decr.",
  "Governance" = "Gov.", "Adjusted" = "Adj.", "Dividend" = "Div.", "Leverage" = "Lev.",
  "Transition" = "Trans.", "Discretionary" = "Disc.", "Environment" = "Env.",
  "Objective" = "Obj.", "Orientation" = "Or.", "Europe" = "EU.", "Sustainable" = "Sust.",
  "Responsible" = "Resp.", "Investment" = "Inv.", "Transatlantic" = "Transat.",
  "Biodiversity" = "Bio.", "BNP Paribas" = "BNP.", "Developed" = "Dev.",
  "Eurozone" = "EU.zn", "Westfield" = "West.", "Telecommunications" = "Telecom."
 )

 # Appliquer les remplacements
 for (word in names(replacements)) {
  text <- gsub(word, replacements[[word]], text)
```

```
  }

  # Vérifier la longueur du texte
  if (nchar(text) > 20) {
    if (grepl("-", text)) {
      parts <- strsplit(text, "-")[[1]]
      first_part <- parts[1]
      if (nchar(first_part) > 20) {
        return(text)
      } else {
        second_part <- parts[2]
        words <- strsplit(second_part, " ")[[1]]
        abbreviated_words <- sapply(words, function(word) {
          if (nchar(word) > 4) {
            return(paste0(substr(word, 1, 4), "."))
          } else {
            return(word)
          }
        })
        return(paste(first_part, paste(abbreviated_words, collapse = " ")))
      }
    } else {
      return(text)
    }
  } else {
    return(text)
  }
}
```

```
EN_Indices_List <- function() {
 url <- "https://live.euronext.com/en/products/indices/list"

 # Lire le contenu de la page
 page <- read_html(url)

 # Utiliser rvest pour extraire le lien du bouton "Last »"
 # page <- read_html(content(response, as = "text", encoding = "UTF-8"))
 last_button <- page %>% html_node("ul.pagination li:last-child a") %>% html_attr("href")

 # Trouver le nombre total de page
 # Extraire les derniers nombres après "page="
 # Extraire le nombre de pages à partir de l'URL
 last_number = str_match(last_button, "page=(\\d+)")[, 2] %>% as.integer()

 result_df <- as.data.frame(matrix(NA, ncol = 7, nrow = 0))

 for (elm in 0:last_number) {
  url =
paste0("https://live.euronext.com/en/products/indices/list?field_featured_indices_is_flagsh_valu
e=All&field_featured_indices_type_target_id=All&field_featured_indices_is_esg_value=All&field_fe
atured_indices_industry_target_id=All&field_featured_indices_region_target_id=All&field_featured
_indices_country_target_id=All&field_dataprovider=All&combine=&page=",
         elm)

  # Lire le contenu de la page
  page <- read_html(url)

  # Extraire les données de la table
  table_data <- page %>%
```

```
  html_table(fill = TRUE)%>%.[[1]]

 # table_data = table_data[[1]]

 # Extraire les liens des indices
 links <- page %>%
  html_nodes("table tbody tr a") %>%
  html_attr("href")

 # Ajouter les liens au tableau
 table_data$Ticker_adn <- links

 # Ajouter les liens au tableau
 table_data$Ticker_adn <- gsub("/en/product/indices/", "", table_data$Ticker_adn)

 result_df <- rbind(result_df, table_data)

}

names(result_df) <- c("Name", "Isin", "Symbol", "Last",
                      "Percentage change (in %)", "Date_Time", "YTD%", "Ticker_adn")

result_df$Last <- gsub(",", "", result_df$Last)

# Old version
# result_df$Last <- gsub("EUR ", "€", result_df$Last)
result_df$Last <- gsub("USD ", "$", result_df$Last)

# Gerer les caractère non-ASCII
```

```
 result_df$Last <- gsub("EUR ", "\u20AC", result_df$Last)
 result_df$Last <- gsub("USD ", "\u24", result_df$Last)

 # result_df$Ticker_adn <- paste0(result_df$Isin, "-", result_df$Symbol)

 return(result_df)

}

# Liste des URLs d'images ESG
esg_images <- c(
 # "https://cdnlearnblog.etmoney.com/wp-content/uploads/2022/09/ESG-funds_Featured.jpg",
 # "https://azprdbrkiiflsecurities.blob.core.windows.net/%24web/assets/Understanding_ESG_Funds_370cd76361.jpg",
 # "https://www.pwc.com/gr/en/csr/assets/Linking-Sustainability-and-CSR-to-Sustainable-investment.PNG",
 # "https://imageio.forbes.com/specials-images/imageserve/645c2c802ae574ecc62796dd/ESG-or-environmental-social-governance--The-company-development-of-a-nature/960x0.jpg?format=jpg&width=1440",
 # "https://i0.wp.com/ontrustcap.com/wp-content/uploads/2020/11/ESG.jpg?fit=1200%2C750&ssl=1",
 # "https://media.licdn.com/dms/image/v2/D4D12AQEudafA3HpyBA/article-cover_image-shrink_720_1280/article-cover_image-shrink_720_1280/0/1697696668130?e=1736985600&v=beta&t=wIm328M2MmbYl2eRSc3Gro63zTxF_tPxxL3ASlwkIN4",
 # "https://media.licdn.com/dms/image/v2/D5612AQFxzFYm8_XbQw/article-cover_image-shrink_720_1280/article-cover_image-shrink_720_1280/0/1684050998490?e=1736985600&v=beta&t=5N2ala2CRxKUqvZu0F0HnxjpJXJ9d9jmkJK_yaBmh_8",
 # "https://natlawreview.com/sites/default/files/styles/article_image/public/2023-09/ESG%20environmental%20Social%20governance%20SEC%20disclosure%20SMALL.jpg?h=95436b9a&itok=yaRvAok1",
```

```
 #          "https://media.licdn.com/dms/image/v2/C4D12AQEUz8HUQSKM4Q/article-cover_image-shrink_423_752/article-cover_image-shrink_423_752/0/1651631374569?e=1736985600&v=beta&t=ICQea8qAPTM6lZza4CvHm-ws4-JB2yBuXcq27l1XzRk",

 #                "https://www.investmentexecutive.com/wp-content/uploads/sites/3/2022/05/iStock-1329455838-Black_Kira-800x600px.jpg",

 # "https://www.financederivative.com/wp-content/uploads/2022/11/b-2.jpg",

 # "https://cdn.due.com/blog/wp-content/uploads/2021/02/stocks-1024x546.jpg"

 # # ,

 # #

 #          #          "https://images.unsplash.com/photo-1451187580459-43490279c0fa?ixlib=rb-1.2.1&auto=format&fit=crop&w=1000&q=80"

 "www/ESG-funds_Featured.jpg",

 "www/Understanding_ESG_Funds.jpg",

 "www/Sustainable-investment.PNG",

 "www/ESG-or-environmental-social-governance.png",

 "www/ESG.png",

 "www/ESG_simple.png",

 "www/ESG_simple_1.png",

 "www/ESG environmental Social governance SEC disclosure SMALL.jpg",

 "www/ESG_simple_2.jpeg",

 "www/ESG_big_format.jpg",

 "www/ESG_lives.jpg",

 "www/stocks-ESG.jpg"


)


# Fonction pour récupérer les données des fonds

# Fonction get_fund_data améliorée

get_fund_data <- function() {
```

```
tryCatch({
 # URL de base
 url <- "https://datawrapper.dwcdn.net/gj0Un/1/dataset.csv"

 column_types <- cols(
  Fund = col_character(),
  `Fund Size Base Currency` = col_number(),
  `Morningstar Category` = col_character(),
  `Morningstar Medalist Rating` = col_character(),
  `YTD Return (%)` = col_double(),
  `12-Month Return (%)` = col_double(),
  `5-Year Return Annlzd (%)` = col_double()
 )

 # data <- read_delim(url ,
 #                    delim = "\t",
 #                    col_types = column_types,
 #                    show_col_types = FALSE)%>%
 #   dplyr::mutate(
 #     # Remplacer les symboles Morningstar par les images correspondantes
 #     `Morningstar Medalist Rating` = case_when(
 #       str_detect(`Morningstar Medalist Rating`, "´") ~ "Bronze",
 #       str_detect(`Morningstar Medalist Rating`, "„") ~ "Silver",
 #       str_detect(`Morningstar Medalist Rating`, "Œ") ~ "Gold",
 #       TRUE ~ `Morningstar Medalist Rating`
 #     )
 #   ) %>%
 #   rename(
 #     Category = `Morningstar Category`,
```

```
#    Rating = `Morningstar Medalist Rating`,
#    YTD_Return = `YTD Return (%)`,
#    Month12_Return = `12-Month Return (%)`,
#    Year5_Return = `5-Year Return Annlzd (%)`
#  )

# Faire la requête HTTP
res <- httr::GET(url)

# Vérifier si la requête a réussi
if (status_code(res) == 200) {
 # print("Je suis dans la partie context")
 # Convertir le contenu binaire en texte
 content_text <- rawToChar(content(res, "raw"))

 # Lire les données
 data <- read_delim(
  content_text,
  delim = "\t",
  col_types = column_types,
  show_col_types = FALSE) %>%
  dplyr::mutate(
   # Remplacer les symboles Morningstar par les images correspondantes
   `Morningstar Medalist Rating` = case_when(
    str_detect(`Morningstar Medalist Rating`, "´") ~ "Bronze",
    str_detect(`Morningstar Medalist Rating`, "„") ~ "Silver",
    str_detect(`Morningstar Medalist Rating`, "Œ") ~ "Gold",
    TRUE ~ `Morningstar Medalist Rating`
   )
```

```
      ) %>%
      rename(
        Category = `Morningstar Category`,
        Rating = `Morningstar Medalist Rating`,
        YTD_Return = `YTD Return (%)`,
        Month12_Return = `12-Month Return (%)`,
        Year5_Return = `5-Year Return Annlzd (%)`
      )

    return(data)
  }

  return(get_mock_data())

 }, error = function(e) {
  return(get_mock_data())
 })
}

# library(rvest)
# library(dplyr)

benchmark_list_Euro <- read_excel("data/benchmark_list_Euro.xlsx")
# benchmark_list_Euro <- read_excel("benchmark_list_Euro.xlsx")

# print('head(benchmark_list_Euro)')
# print(head(benchmark_list_Euro))
```

```
Get_Benchmark_list <- function(source = 'Yahoo') {
 if(!is.character(source)){
  stop("'Source' should be a character and one of : Yahoo, Euronext")
 }

 source = str_to_sentence(source)

 if(!source %in% c("Yahoo", "Euronext")){
  stop("'Source' should be one of : Yahoo, Euronext")
 }

 if(source == 'Yahoo'){
  # Charger les packages nécessaires
  # URL de la page
  url <- 'https://finance.yahoo.com/markets/world-indices/'

  # Lire le contenu de la page
  page <- read_html(url)

  # Extraire le tableau en utilisant le sélecteur CSS
  table <- page %>%
   html_node('table.markets-table.freeze-col.yf-paf8n5.fixedLayout') %>%
   html_table()

  # print(dim(table))
  # print(head(table))

  # Vérifier si le tableau n'est pas nul
  if (!is.null(table)) {
```

```
# Extraire les liens href
links <- page %>%
  html_nodes('table.markets-table.freeze-col.yf-paf8n5.fixedLayout a') %>%
  html_attr('href')

# Filtrer les liens contenant '/quote/'
filtered_links <- links[grepl('/quote/', links)]

# Compléter les liens avec l'URL de base
full_links <- paste0('https://finance.yahoo.com', filtered_links)

# Ajouter les liens filtrés et complétés au tableau
table$Links <- full_links

# Garder que les éléments nécessaires
# table <- table[, c(1, 2, 10)]
table <- table[, c(1, 2, 10, 4:7)]

# Extraire le prix avant le premier signe "+" ou "-"
extracted_prices <- sub("([0-9,\\.]+)[+-].*", "\\1", table$Price)
extracted_prices <- gsub(",", "", extracted_prices) %>% as.numeric()

# Mettre à jour la colonne Price avec les prix extraits
table$Price <- extracted_prices

# Nettoyer les noms
table$Name <- gsub("DAX                    P", "DAX P", table$Name)
table$Name <- gsub("EURO STOXX 50          I", "EURO STOXX 50 I", table$Name)
```

```
    # Garder les colonnes nécessaires
    table <- table[, c(1:3)]
    # print("final_table")
    # print(dim(table))
    # print(head(table))

  } else {
    # Retourner un tableau par défaut si le tableau est nul
    table <- structure(list(
      Symbol = c("^GSPC", "^DJI", "^IXIC", "^NYA", "^XAX", "^BUK100P", "^RUT", "^VIX",
                 "^FTSE", "^GDAXI", "^FCHI", "^STOXX50E", "^N100", "^BFX", "MOEX.ME", "^HSI",
                 "^STI", "^AXJO", "^AORD", "^BSESN", "^JKSE", "^KLSE", "^NZ50", "^KS11", "^TWII",
                 "^GSPTSE", "^BVSP", "^MXX", "^IPSA", "^MERV", "^TA125.TA", "^CASE30", "^JN0U.JO",
                 "DX-Y.NYB", "^125904-USD-STRD", "^XDB", "^XDE", "000001.SS", "^N225", "^XDN",
"^XDA"),
      Name = c("S&P 500", "Dow Jones Industrial Average", "NASDAQ Composite", "NYSE COMPOSITE
(DJ)",
               "NYSE AMEX COMPOSITE INDEX", "Cboe UK 100", "Russell 2000", "CBOE Volatility Index",
"FTSE 100",
               "DAX P", "CAC 40", "EURO STOXX 50 I", "Euronext 100 Index", "BEL 20", "Public Joint-Stock
Company Moscow Exchange MICEX-RTS",
               "HANG SENG INDEX", "STI Index", "S&P/ASX 200", "ALL ORDINARIES", "S&P BSE SENSEX",
"IDX COMPOSITE", "FTSE Bursa Malaysia KLCI",
               "S&P/NZX 50 INDEX GROSS ( GROSS", "KOSPI Composite Index", "TWSE Capitalization
Weighted Stock Index", "S&P/TSX Composite index",
               "IBOVESPA", "IPC MEXICO", "S&P IPSA", "MERVAL", "TA-125", "EGX 30 Price Return Index",
"Top 40 USD Net TRI Index",
               "US Dollar Index", "MSCI EUROPE", "British Pound Currency Index", "Euro Currency Index",
"SSE Composite Index", "Nikkei 225",
               "Japanese Yen Currency Index", "Australian Dollar Currency Index"),
      Links = c("https://finance.yahoo.com/quote/%5EGSPC/",
"https://finance.yahoo.com/quote/%5EDJI/",
```

"https://finance.yahoo.com/quote/%5EIXIC/",
"https://finance.yahoo.com/quote/%5ENYA/",

"https://finance.yahoo.com/quote/%5EXAX/",
"https://finance.yahoo.com/quote/%5EBUK100P/",

"https://finance.yahoo.com/quote/%5ERUT/",
"https://finance.yahoo.com/quote/%5EVIX/",

"https://finance.yahoo.com/quote/%5EFTSE/",
"https://finance.yahoo.com/quote/%5EGDAXI/",

"https://finance.yahoo.com/quote/%5EFCHI/",
"https://finance.yahoo.com/quote/%5ESTOXX50E/",

"https://finance.yahoo.com/quote/%5EN100/",
"https://finance.yahoo.com/quote/%5EBFX/",

"https://finance.yahoo.com/quote/MOEX.ME/",
"https://finance.yahoo.com/quote/%5EHSI/",

"https://finance.yahoo.com/quote/%5ESTI/",
"https://finance.yahoo.com/quote/%5EAXJO/",

"https://finance.yahoo.com/quote/%5EAORD/",
"https://finance.yahoo.com/quote/%5EBSESN/",

"https://finance.yahoo.com/quote/%5EJKSE/",
"https://finance.yahoo.com/quote/%5EKLSE/",

"https://finance.yahoo.com/quote/%5ENZ50/",
"https://finance.yahoo.com/quote/%5EKS11/",

"https://finance.yahoo.com/quote/%5ETWII/",
"https://finance.yahoo.com/quote/%5EGSPTSE/",

"https://finance.yahoo.com/quote/%5EBVSP/",
"https://finance.yahoo.com/quote/%5EMXX/",

"https://finance.yahoo.com/quote/%5EIPSA/",
"https://finance.yahoo.com/quote/%5EMERV/",

"https://finance.yahoo.com/quote/%5ETA125.TA/",
"https://finance.yahoo.com/quote/%5ECASE30/",

"https://finance.yahoo.com/quote/%5EJN0U.JO/", "https://finance.yahoo.com/quote/DX-Y.NYB/",

"https://finance.yahoo.com/quote/%5E125904-USD-STRD/",
"https://finance.yahoo.com/quote/%5EXDB/",

```
            "https://finance.yahoo.com/quote/%5EXDE/",
"https://finance.yahoo.com/quote/000001.SS/",
            "https://finance.yahoo.com/quote/%5EN225/",
"https://finance.yahoo.com/quote/%5EXDN/",
            "https://finance.yahoo.com/quote/%5EXDA/")
    ), row.names = c(NA, -41L), class = c("tbl_df", "tbl", "data.frame"))
  }
 } else if(source == "Euronext") {

  # table = EN_Indices_List()

  table = benchmark_list_Euro

  # Vérifier si le tableau n'est pas nul
  if (!is.null(table)) {
   # table = table[, c(1,8)] #si j'utilise la fonction # table = EN_Indices_List()
   table = table

  }else{
   table = NULL

  }
 }

 return(table)

}

# Appeler la fonction pour obtenir le tableau
```

```
benchmark_list <- Get_Benchmark_list()
# print(benchmark_list)

# benchmark_list_Euro <- Get_Benchmark_list(source = "Euronext")



# Ma dataframe 'table' contient les colonnes 'Symbol' et 'Name'
Get_bench_data <- function(Name, from, to, na_method = 'keep', source = 'Yahoo') {
 # Vérifier que 'table' est bien un dataframe ou une liste
 if (!exists("benchmark_list") || !is.data.frame(benchmark_list)) {
  stop("'table' doit être un dataframe contenant les colonnes 'Name' et 'Symbol'.")
 }

 if(!is.character(source)){
  stop("'Source' should be a character and one of : Yahoo, Euronext")
 }

 source = str_to_sentence(source)

 # Appeler la fonction pour obtenir le tableau
 # benchmark_list <- Get_Benchmark_list(source = source)
 # print(benchmark_list)



 if(!source %in% c("Yahoo", "Euronext")){
  stop("'Source' should be one of : Yahoo, Euronext")
 }

 if(source == 'Yahoo'){
```

```
  # Rechercher le Symbol correspondant au Name
  symbol <- benchmark_list$Symbol[benchmark_list$Name == Name]

  # the_idx <- which(table$Name == Name) #ligne 385
  # symbol <- table$Symbol[the_idx]

  # Vérifier si le Symbol a été trouvé
  if (length(symbol) == 0) {
    stop("Le nom spécifié n'a pas été trouvé dans la table.")
  }

  # Télécharger les données en utilisant le Symbol trouvé
  data <- qrmtools::get_data(symbol, from = from, to = to)

  # print('head(data)')
  # print(head(data))
  # Transformer l'objet xts en dataframe
  df <- data.frame(Date = index(data), Value = coredata(data))
  colnames(df)[2] <- colnames(data) # Renommer la seconde colonne

  print("test lorsqu'on utilise yahoo")
  # print('head(df)')
  # print(head(df))

}else if(source == 'Euronext'){

  # Rechercher le Symbol correspondant au Name
  symbol_adn <- benchmark_list_Euro$Ticker_adn[benchmark_list_Euro$Name == Name]
```

```
 # Vérifier si le symbol_adn a été trouvé
 if (length(symbol_adn) == 0) {
  stop("Le nom spécifié n'a pas été trouvé dans la table.")
 }

 # Télécharger les données en utilisant le Symbol trouvé
 # data <- qrmtools::get_data(symbol, from = from, to = to)
 data <- EN_Ticker_hcData(ticker = symbol_adn, escape = TRUE, from = from, to = to)

 print("new test data")
 print(str(data))
 print('class(data)')
 print(class(data))

 df <- data[, c(1,2)]

 colnames(df) = c('Date', symbol_adn)

} else {
 return(NULL)
}

if(!is.data.frame(df)){
 stop("Vérifier votre connexion")
}

# Appliquer la méthode de gestion des valeurs manquantes
if (na_method == 'omit') {
 df <- na.omit(df)
```

```
 } else if (na_method == 'mean') {
   df[,2][is.na(df[,2])] <- mean(df[,2], na.rm = TRUE)
 } else if (na_method == 'zero') {
   df[,2][is.na(df[,2])] <- 0
 } else if (na_method == 'previous') {
   df[,2] <- zoo::na.locf(df[,2], na.rm = FALSE)
 }else{
   df <- df
 }

 df[,2] <- round(df[,2], 2)

 return(df)
}

# Exemple d'utilisation de la fonction
# result <- Get_bench_data("TA-125", from = "2010-01-01", to = Sys.Date(), na_method = "zero")
# result <- Get_bench_data("TA-125", from = "2010-01-01", to = Sys.Date(), na_method = "mean")
# result <- Get_bench_data("TA-125", from = "2010-01-01", to = Sys.Date(), na_method = "previous")
# result <- Get_bench_data("S&P 500", from = "2010-01-01", to = Sys.Date(), na_method = 'rep_by_mean')
# print(head(result))



# Fonction helper pour générer des données factices
get_mock_data <- function() {
 data.frame(
   Fund = c(
     "Baillie Gifford Positive Change B Acc",
```

```
    "Ninety One Global Environment I Acc",
    "Stewart Investors Global EM Sustainability",
    "Impax Environmental Markets Trust",
    "RobecoSAM Smart Energy Fund",
    "Pictet Global Environmental Opportunities",
    "Liontrust Sustainable Future Global Growth",
    "Brown Advisory US Sustainable Growth",
    "Schroder Global Energy Transition",
    "Jupiter Green Investment Trust"
  ),
  Category = c(
    "Global Large-Cap Growth Equity",
    "Sector Equity Ecology",
    "Global Emerging Markets Equity",
    "Sector Equity Ecology",
    "Sector Equity Alternative Energy",
    "Sector Equity Ecology",
    "Global Large-Cap Growth Equity",
    "US Large-Cap Growth Equity",
    "Sector Equity Alternative Energy",
    "Sector Equity Ecology"
  ),
  Rating = sample(c("Gold", "Silver", "Bronze", "Neutral"), 10, replace = TRUE),
  YTD_Return = round(runif(10, 10, 30), 2),
  Month12_Return = round(runif(10, 15, 40), 2),
  Year5_Return = round(runif(10, 8, 25), 2)
 )
}
```

```
vector_simulation =  seq(0,100000, by = 1000)[-1]


vector_simulation = c(100, 500, vector_simulation)


# Les footnotes à afficher

footnotes <- c(

 "[1] Data provided by Refinitiv.",

 "[2] Data provided by EDGAR Online.",

 "[3] Data derived from multiple sources or calculated by Yahoo Finance.",

 "[4]  Data provided by Morningstar, Inc.",

 "[5] Shares outstanding is taken from the most recently filed quarterly or annual report and Market Cap is calculated using shares outstanding.",

 "[6]  Implied Shares Outstanding of common equity, assuming the conversion of all convertible subsidiary equity into common.",

 "[7]  EBITDA is calculated by S&P Global Market Intelligence using methodology that may differ from that used by a company in its reporting.",

 "[8]  A company's float is a measure of the number of shares available for trading by the public. It's calculated by taking the number of issued and outstanding shares minus any restricted stock, which may not be publicly traded."

)


# footnotes <- c(

#   paste0("<sup>", "1" , "</sup>", "  Data provided by Refinitiv."),

#   paste0("<sup>", "2" , "</sup>", "  Data provided by EDGAR Online."),

#    paste0("<sup>", "3" , "</sup>", "  Data derived from multiple sources or calculated by Yahoo Finance."),

#   paste0("<sup>", "4" , "</sup>", "  Data provided by Morningstar, Inc."),

#  paste0("<sup>", "5" , "</sup>", " Shares outstanding is taken from the most recently filed quarterly or annual report and Market Cap is calculated using shares outstanding."),

#   paste0("<sup>", "6" , "</sup>", "  Implied Shares Outstanding of common equity, assuming the conversion of all convertible subsidiary equity into common."),
```

```
#   paste0("<sup>", "7" , "</sup>", "  EBITDA is calculated by S&P Global Market Intelligence using methodology that may differ from that used by a company in its reporting."),

#  paste0("<sup>", "8" , "</sup>", "  A company's float is a measure of the number of shares available for trading by the public. It's calculated by taking the number of issued and outstanding shares minus any restricted stock, which may not be publicly traded.")

# )



footnotes_Financial <- c(

  "mrq = Most Recent Quarter.",

  "ttm = Trailing Twelve Months.",

  "yoy = Year Over Year.",

  "lfy = Last Fiscal Year.",

  "fye  = Fiscal Year Ending."

)


# Fonction pour valider le format ISIN

is_valid_isin <- function(isin) {

  # Format ISIN: 2 lettres suivies de 10 caractères alphanumériques

  grepl("^[A-Z]{2}[A-Z0-9]{10}$", isin)

}


# Best Process

portf_analysis <- function(tickers, rf = 0.05, nport = 20, start_date = Sys.Date() - 365, end_date = Sys.Date()) {


  getSymbols(tickers, src = 'yahoo', from = start_date, to = end_date)


  # Calculate daily log returns

  returns <- do.call(merge, lapply(tickers, function(ticker) {
```

```
  dailyReturn(Cl(get(ticker)), type = "log")
}))

# Calculate annualized expected returns and covariance matrix
er <- colMeans(returns) * 252
covmat <- cov(returns) * 252

names(er) = tickers

dimnames(covmat) = list(tickers, tickers)



# Compute global minimum variance and tangency portfolios
gmin.port <- globalMin.portfolio(er, covmat)
tan.port <- tangency.portfolio(er, covmat, rf, shorts = FALSE)

# Compute the efficient frontier
ef <- efficient.frontier(er, covmat, alpha.min = -2, alpha.max = 1.5, nport = nport)

# Calculate the slope of the Capital Market Line (CML)
sr.tan <- (tan.port$er - rf) / tan.port$sd

# Define the 3D coordinates for the tangent line
max_sd <- max(ef$sd)
tangent_x <- seq(0, max_sd, length.out = 100)
tangent_y <- rf + sr.tan * tangent_x
tangent_z <- (tangent_y - rf) / tangent_x

tangent_line <- list(
```

```
  x = tangent_x,
  y = tangent_y,
  z = tangent_z
 )

 # Add a third dimension (Sharpe Ratio) for the 3D plot
 sharpe_ratios <- (ef$er - rf) / ef$sd

 # Plot the 3D graph
 fig <- plot_ly() %>%
  add_trace(x = ef$sd, y = ef$er, z = sharpe_ratios, type = 'scatter3d', mode = 'lines+markers',
         line = list(color = 'blue', width = 2),
         marker = list(color = 'blue', size = 4),
         name = 'Efficient Frontier') %>%
  add_markers(x = gmin.port$sd, y = gmin.port$er, z = (gmin.port$er - rf) / gmin.port$sd,
          marker = list(color = 'green', size = 10),
          name = 'Global Minimum Variance') %>%
  add_markers(x = tan.port$sd, y = tan.port$er, z = (tan.port$er - rf) / tan.port$sd,
          marker = list(color = 'red', size = 10),
          name = 'Tangency Portfolio') %>%
  add_trace(x = tangent_line$x, y = tangent_line$y, z = tangent_line$z, type = 'scatter3d', mode =
'lines',
         line = list(color = 'green', width = 3),
         name = 'Capital Market Line') %>%
  layout(scene = list(
   xaxis = list(title = "Risk (Standard Deviation)"),
   yaxis = list(title = "Expected Return"),
   zaxis = list(title = "Sharpe Ratio"),
   title = "3D Portfolio Analysis"
```

```
  ))

 # return(fig)

 return(list(fig = fig,
             ef = ef,
             tan.port = tan.port,
             gmin.port = gmin.port,
             covmat = covmat,
             er = er,
             sr.tan = sr.tan
 ))
}

# # Process 1 best with little correction
# portf_analysis <- function(tickers, rf = 0.05, nport = 20, start_date = Sys.Date() - 365, end_date = Sys.Date()) {
#
#   # # Step 1: Get stock data
#   # stock_data <- tq_get(tickers, from = start_date, to = end_date, get = 'stock.prices')
#   #
#   # # Step 2: Calculate expected returns and covariance matrix
#   # returns <- stock_data %>%
#   #   group_by(symbol) %>%
#   #   tq_transmute(select = adjusted, mutate_fun = periodReturn, period = 'daily', type = 'log') %>%
#   #   ungroup()
#   #
#   # er <- returns %>%
#   #   group_by(symbol) %>%
```

```
#   #   summarize(expected_return = mean(daily.returns, na.rm = TRUE)) %>%
#   #   pull(expected_return)
#   #
#   # names(er) <- tickers
#   # print(er)
#   #
#   # covmat <- returns %>%
#   #   spread(symbol, daily.returns) %>%
#   #   select(-date) %>%
#   #   na.omit() %>%
#   #   cov()
#
#   getSymbols(tickers, src = 'yahoo', from = start_date, to = end_date)
#
#   # Calculer les rendements quotidiens logarithmiques
#   returns <- do.call(merge, lapply(tickers, function(ticker) {
#     dailyReturn(Cl(get(ticker)), type = "log")
#   }))
#
#   # Calcul des rendements attendus et de la matrice de covariance
#   er <- colMeans(returns) * 252  # Rendements annuels attendus
#   covmat <- cov(returns) * 252  # Matrice de covariance annualisée
#
#   # if (gmin.port$er < risk.free)
#   #   stop("Risk-free rate greater than avg return on global minimum variance portfolio")
#
#
#   # Step 3: Compute the tangency and global minimum variance portfolios
#   gmin.port <- globalMin.portfolio(er, covmat)
```

```
#
#  tan.port <- tangency.portfolio(er, covmat, rf, shorts = FALSE)
#
#  # Step 4: Compute the efficient frontier
#  ef <- efficient.frontier(er, covmat, alpha.min = -2, alpha.max = 1.5, nport = nport)
#
#  # Calcul de la pente de la ligne du marché de capitaux (CML)
#  sr.tan = (tan.port$er - rf) / tan.port$sd
#
#  # Limit the tangent line to the maximum x-coordinate (risk) of the efficient frontier
#  max_sd <- max(ef$sd)
#  tangent_y <- rf + sr.tan * max_sd  # Calculate the y-value at the maximum x-value
#
#  # Step 6: Plot using Plotly
#  fig <- plot_ly() %>%
#   add_trace(x = ef$sd, y = ef$er, type = 'scatter', mode = 'lines+markers',
#           line = list(color = 'blue', width = 2),
#           marker = list(color = 'blue', size = 4),
#           name = 'Efficient Frontier') %>%
#   add_markers(x = gmin.port$sd, y = gmin.port$er,
#            marker = list(color = 'green', size = 10),
#            name = 'Global Minimum Variance') %>%
#   add_markers(x = tan.port$sd, y = tan.port$er,
#            marker = list(color = 'red', size = 10),
#            name = 'Tangency Portfolio') %>%
#   layout(title = "Portfolio Analysis",
#         xaxis = list(title = "Risk (Standard Deviation)"),
#         yaxis = list(title = "Expected Return"),
#         legend = list(orientation = 'h', x = 0.5, y = -0.2),
```

```
#         shapes = list(
#           # Tangent line limited to the efficient frontier range
#           list(type = "line", x0 = 0, y0 = rf, x1 = max_sd, y1 = tangent_y, line = list(color = 'green', width = 2))
#         ))
#
#   return(list(fig = fig,
#             ef = ef,
#             tan.port = tan.port,
#             gmin.port = gmin.port,
#             covmat = covmat,
#             er = er,
#             sr.tan = sr.tan
#   ))
# }


# test = portf_analysis(tickers, rf = 0.05, nport = 20, start_date = "2010-01-01", end_date = "2024-01-01")
# test = portf_analysis(tickers, rf = 0.05, nport = 20, start_date = Sys.Date() - 90, end_date = Sys.Date())
# test$fig
# test$ef$er
# test$ef$sd
# test$tan.port$er
# test$tan.port$sd
# test$tan.port$weights
# test$gmin.port
# test$gmin.port$er
# test$gmin.port$sd
# test$gmin.port$weights
```

```
# test$covmat

# test$er

# test$sr.tan




# # Process 2

# portf_analysis <- function(tickers, rf = 0.05, nport = 20, start_date = Sys.Date() - 365, end_date = Sys.Date()) {

#

#   # Obtenir les données de stock

#   getSymbols(tickers, src = 'yahoo', from = start_date, to = end_date)

#

#   # Calculer les rendements quotidiens logarithmiques

#   returns <- do.call(merge, lapply(tickers, function(ticker) {

#     dailyReturn(Cl(get(ticker)), type = "log")

#   }))

#

#   # print(head(returns))

#

#   # Calcul des rendements attendus et de la matrice de covariance

#   er <- colMeans(returns) * 252  # Rendements annuels attendus

#   names(er) = tickers

#   covmat <- cov(returns) * 252  # Matrice de covariance annualisée

#

#   dimnames(covmat) = list(tickers, tickers)

#

#   # Calculer les portefeuilles tangents et de variance minimale globale

#   gmin.port <- globalMin.portfolio(er, covmat)
```

```
#  tan.port <- tangency.portfolio(er, covmat, rf, shorts = FALSE)
#
#  # Calculer la frontière efficiente
#  ef <- efficient.frontier(er, covmat, alpha.min = -2, alpha.max = 1.5, nport = nport)
#
#  # Calcul de la pente de la ligne du marché de capitaux (CML)
#  sr.tan = (tan.port$er - rf) / tan.port$sd
#
#  # Étendre la ligne verte (CML) au-delà du point de tangence
#  x_extend <- seq(0, max(ef$sd) + 0.1, length.out = 100)
#  y_extend <- rf + sr.tan * x_extend
#
#  # Graphique avec Plotly
#  fig <- plot_ly() %>%
#   add_trace(x = ef$sd, y = ef$er, type = 'scatter', mode = 'lines+markers',
#           line = list(color = 'blue', width = 2),
#           marker = list(color = 'blue', size = 4),
#           name = 'Efficient Frontier') %>%
#   add_markers(x = gmin.port$sd, y = gmin.port$er,
#            marker = list(color = 'green', size = 10),
#            name = 'Global Minimum Variance') %>%
#   add_markers(x = tan.port$sd, y = tan.port$er,
#            marker = list(color = 'red', size = 10),
#            name = 'Tangency Portfolio') %>%
#   add_trace(x = x_extend, y = y_extend, type = 'scatter', mode = 'lines',
#           line = list(color = 'green', width = 2),
#           name = 'Capital Market Line') %>%
#   layout(title = "Portfolio Analysis",
#         xaxis = list(title = "Risk (Standard Deviation)"),
```

```
#         yaxis = list(title = "Expected Return"),
#         legend = list(orientation = 'h', x = 0.5, y = -0.2))
#
#   # return(fig)
#
#   return(list(fig = fig,
#             ef = ef,
#             tan.port = tan.port,
#             gmin.port = gmin.port,
#             covmat = covmat,
#             er = er,
#             sr.tan = sr.tan
#   ))
#
#
# }
```

```
# # Fonction pour extraire et ajouter les footnotes aux rownames
add_footnotes <- function(df, footnotes) {
  rownames_with_footnotes <- rownames(df)

  # Pour chaque numéro de note (1 à 8)
  for (i in 1:length(footnotes)) {
    pattern <- paste0(" ", i, "$")  # Cherche le numéro à la fin après un espace
    replacement <- paste0(" <sup>", i, "</sup>")
    rownames_with_footnotes <- gsub(pattern, replacement, rownames_with_footnotes)
```

```
 }

 rownames(df) <- rownames_with_footnotes

 return(df)
}

# Old version
# add_footnotes <- function(df, footnotes) {
#   # Extraire les indices de footnotes des rownames
#   rownames_with_footnotes <- gsub("(.*\\d)$", "\\1*", rownames(df))
#
#   # Ajouter les footnotes aux rownames
#   for (i in 1:length(footnotes)) {
#     rownames_with_footnotes <- gsub(paste0(i, "\\*"),
#                                     paste0("<sup>", i, "</sup>"),
#                                     rownames_with_footnotes)
#   }
#
#   rownames(df) <- rownames_with_footnotes
#   df
# }



# NEw

add_footnotes_F <- function(df) {
 rownames_with_footnotes <- rownames(df)
```

```
 # Liste des abréviations à remplacer
 replacements <- list(
  mrq = "Most Recent Quarter",
  ttm = "Trailing Twelve Months",
  yoy = "Year Over Year",
  lfy = "Last Fiscal Year",
  fye = "Fiscal Year Ending"
 )

 # Remplacer les patterns entre parenthèses
 for (abbrev in names(replacements)) {
  pattern <- paste0("\\(", abbrev, "\\)")
  replacement <- paste0(" <sup>", abbrev, "</sup>")
  rownames_with_footnotes <- gsub(pattern, replacement, rownames_with_footnotes)
 }

 rownames(df) <- rownames_with_footnotes
 return(df)
}

# add_footnotes_F <- function(df, footnotes) {
#   # Extraire les indices de footnotes des rownames
#   list_guide = unlist(str_extract_all(rownames(df), "\\([^()]+\\)"))
#
#   list_guide = unique(list_guide)
#
#   list_guide_sav = gsub("\\(|\\)", "",list_guide)
#
```

```
#   rownames_with_footnotes <- rownames(df)
#
#
#   for (i in 1:length(list_guide)) {
#     elem = paste0("\\", list_guide[i])
#     new_elm = paste0(list_guide_sav[i], "\\*")
#
#     rownames_with_footnotes = gsub(elem,
#                                    new_elm,
#                                    rownames_with_footnotes)
#   }
#
#   abbrev_ = c("mrq", "ttm", "yoy", "lfy", "fye")
#
#   # Ajouter les footnotes aux rownames
#   for (i in abbrev_) {
#     rownames_with_footnotes <- gsub(paste0(i, "\\*"),
#                                     paste0("<sup>", i, "</sup>"),
#                                     rownames_with_footnotes)
#   }
#
#   rownames(df) <- rownames_with_footnotes
#   df
# }



convert_to_numeric <- function(value) {
  if (str_detect(value, "M")) {
    return(as.numeric(str_replace(value, "M", "")) * 1e6)
```

```
  } else if (str_detect(value, "B")) {
    return(as.numeric(str_replace(value, "B", "")) * 1e9)

  } else if (str_detect(value, "T")) {
    return(as.numeric(str_replace(value, "T", "")) * 1e12)

  } else if (str_detect(value, "%")) {
    return(as.numeric(str_replace(value, "%", "")) / 100)

  } else {
    return(as.numeric(value))

  }
}



comp_graph_plot <- function(df, title = ""){
  if(!is.null(df)){
    df = na.omit(df)
    plot = df%>%hchart('column',
                       hcaes(x = `Info`, y = `Value`,
                             group = `Ticker`))%>%
      hc_xAxis(title = list(text = "")) %>%
      hc_title(text = title)

    return(plot)

  }else{
```

```
   return(NULL)
  }
}

# Fonction pour calculer le Sharpe ratio modifié
calc_sresg <- function(sc, returns, rf, esg, type_return = "daily") {
  annualization_factor <- switch(type_return,
                                 "daily" = 252,
                                 "weekly" = 52,
                                 "monthly" = 12,
                                 "yearly" = 1,
                                 stop("type_return doit être 'daily', 'weekly', 'monthly' ou 'yearly'"))

  annualized_return <- mean(returns) * annualization_factor
  annualized_rf <- mean(rf) * annualization_factor
  annualized_volatility <- sd(returns) * sqrt(annualization_factor)
  sr <- (annualized_return - annualized_rf) / annualized_volatility

  ifelse(sr > 0,
       sr * (1 + esg)^sc,
       sr * (1 + esg)^(-sc))
}

# Fonction à optimiser
optimize_sc <- function(sc, returns, rf, esg, type_return) {
  sresg <- calc_sresg(sc, returns, rf, esg, type_return)
  return(-mean(sresg) / sd(sresg))  # Retourner le négatif du ratio de Sharpe (car la fonction optimize() minimise)
}
```

```
# Fonction pour calculer le Sharpe ratio modifié et ajouter les colonnes

# Ancienne version mais seul problème les valeurs des colonnes Sharpe ratio modifiées sont toutes identiques

calculate_modified_sharpe <- function(df, sc_optimal, rf, type_return = "daily") {

 annualization_factor <- switch(type_return,

                    "daily" = 252,

                    "weekly" = 52,

                    "monthly" = 12,

                    "yearly" = 1)


 df$rf = rf


 df = na.omit(df)


 # df_new = df[, c('Date', 'Ticker_return', 'rf')]


 df$Sharpe_Ratio = NA


 df$Modified_SR = NA

 # the_std = sd(df$Ticker_return)



 list_NA = NULL

 # init_position = 1

 for (i in 1:nrow(df)) {

  the_std = sd(na.omit(df$Ticker_return[c(1:i)]))


  # print(round(the_std, 3))
```

```
  if(is.na(the_std) || the_std == 0){

    df[i, 'Sharpe_Ratio'] = NA

    df[i, 'Modified_SR'] = NA

    list_NA = append(list_NA, i)

  }else{

    df[[i, 'Sharpe_Ratio']] = ((df[[i, 'Ticker_return']] - df[[i, 'rf']]) * annualization_factor) / (the_std * sqrt(annualization_factor))


    df[[i, 'Modified_SR']] = ifelse(df[[i, 'Sharpe_Ratio']] > 0,
                                    df[[i, 'Sharpe_Ratio']]* (1 + df[[i, 'ESG']])^sc_optimal,
                                    df[[i, 'Sharpe_Ratio']]* (1 + df[[i, 'ESG']])^(-sc_optimal)
    )

  }

}

print(paste0("Nous avons ", length(list_NA), " lignes contenant NA"))


# df$Sharpe_Ratio = round(SharpeRatio(df$Ticker_return, Rf = df$rf, FUN = "StdDev", annualize = TRUE), 4)
```

```
 # verifier cette partie pour voir si je dois garder ou modifier la partie


 # df <- df %>%

 #   mutate(

 #    # Sharpe_Ratio = (mean(Ticker_return) * annualization_factor - mean(rf) * annualization_factor) / (sd(Ticker_return) * sqrt(annualization_factor)),

 #

 #    # j'ai essayé ça et s'affiche uniquement lorsque periode annuelle

 #         # Sharpe_Ratio = (mean(Ticker_return) * annualization_factor - mean(Index_return) * annualization_factor) / (sd(Ticker_return) * sqrt(annualization_factor)),

 #

 #     # Sharpe_Ratio = ((Ticker_return - Index_return) * annualization_factor) / (sd(Ticker_return) * sqrt(annualization_factor)),

 #

 #          # Sharpe_Ratio = ((Ticker_return - rf) * annualization_factor) / (sd(Ticker_return) * sqrt(annualization_factor)),

 #    # Modified_SR = calc_sresg(sc_optimal,

 #    #                 Ticker_return,

 #    #                 rf,

 #    #                 # ESG / 100,

 #    #                 ESG,

 #    #                 type_return)

 #    # ,

 #    #

 #    Modified_SR = ifelse((Ticker_return - rf) / sd(Ticker_return) * sqrt(annualization_factor) > 0,

 #                         ((Ticker_return - rf) / sd(Ticker_return) * sqrt(annualization_factor)) * (1 + ESG)^sc_optimal,

 #                 ((Ticker_return - rf) / sd(Ticker_return) * sqrt(annualization_factor)) * (1 + ESG)^(-sc_optimal))

 #
```

```
  #   )

  # df[, "Sharpe_Ratio"] <- round(df[, "Sharpe_Ratio"], 4)
  # df[, "Modified_SR"] <- round(df[, "Modified_SR"], 4)

  return(df)
}



# Nouvelle version à tester
# Ne fonctionne pas
# Main function to calculate both Sharpe ratio and Modified Sharpe ratio
# calculate_modified_sharpe <- function(df, sc_optimal, rf, type_return = "daily") {
#   annualization_factor <- switch(type_return,
#                          "daily" = 252,
#                          "weekly" = 52,
#                          "monthly" = 12,
#                          "yearly" = 1,
#                          stop("type_return must be 'daily', 'weekly', 'monthly' or 'yearly'"))
#
#   df <- df %>%
#     rowwise() %>%
#     mutate(
#       Sharpe_Ratio = calc_sharpe_ratio(Ticker_return, rf, annualization_factor),
#                    Modified_SR = calc_modified_sresg(Ticker_return, rf, ESG / 100, sc_optimal,
annualization_factor)
#     ) %>%
#     ungroup()
#
```

```
#   return(df)
# }




# Function to calculate the rolling Sharpe ratio and modified Sharpe ratio


calc_rolling_sresg <- function(df, sc, rf, esg_col, type_return = "daily") {
  annualization_factor <- switch(type_return,
                                 "daily" = 252,
                                 "weekly" = 52,
                                 "monthly" = 12,
                                 "yearly" = 1)


  # Calculate the rolling mean and standard deviation
  df <- df %>%
    mutate(
      rolling_mean_ticker = rollapply(Ticker_return, width = 252, FUN = mean, fill = NA, align = 'right', na.rm = TRUE),
      rolling_sd_ticker = rollapply(Ticker_return, width = 252, FUN = sd, fill = NA, align = 'right', na.rm = TRUE),
      rolling_mean_rf = rollapply(rf, width = 252, FUN = mean, fill = NA, align = 'right', na.rm = TRUE)
    ) %>%
    dplyr::filter(!is.na(rolling_mean_ticker), !is.na(rolling_sd_ticker), !is.na(rolling_mean_rf)) %>%
    mutate(
      annualized_return = rolling_mean_ticker * annualization_factor,
      annualized_rf = rolling_mean_rf * annualization_factor,
      annualized_volatility = rolling_sd_ticker * sqrt(annualization_factor),
      Sharpe_Ratio = (annualized_return - annualized_rf) / annualized_volatility,
```

```
    Modified_SR = ifelse(Sharpe_Ratio > 0,
                         Sharpe_Ratio * (1 + !!sym(esg_col) / 100)^sc,
                         Sharpe_Ratio * (1 + !!sym(esg_col) / 100)^(-sc))
   )

 return(df)
}



########## old v. ############
# Cette version est celle qui fonctionne bien mais je veux voir comment ajuster les axes
# La meilleure fonction à ajuster
# Date du 16/01/2025

customize_eff_front <- function(fig_dt) {
 # Vérification si les données existent
 if (is.null(fig_dt) || nrow(fig_dt) == 0) {
  return(NULL)
 }

 # Étape 1 : Filtrer les données en fonction de l'ESG
 hist_data <- hist(fig_dt$ESG, plot = FALSE, breaks = 3)
 filtered_list <- list()

 for (i in 1:(length(hist_data$breaks) - 1)) {
  if (hist_data$counts[i] > 0) {
   filtered_data <- fig_dt %>%
    dplyr::filter(ESG >= hist_data$breaks[i] & ESG < hist_data$breaks[i + 1])
   filtered_list[[paste0("Class_", i)]] <- filtered_data
```

```
 }
}

# Étape 2 : Séparer les données en rendements positifs et négatifs
dta_pos <- fig_dt %>% dplyr::filter(Return >= 0)
dta_neg <- fig_dt %>% dplyr::filter(Return < 0)

# Étape 3 : Organiser les données pour les rendements positifs
dta_pos_arrange <- dta_pos %>% arrange(Volatility)
final_dt_positive <- dta_pos_arrange[1, ]  # Initialisation avec le premier point

for (i in 2:nrow(dta_pos_arrange)) {
  if (dta_pos_arrange$Return[i] >= max(final_dt_positive$Return)) {
    final_dt_positive <- rbind(final_dt_positive, dta_pos_arrange[i, ])
  }
}

# Étape 4 : Organiser les données pour les rendements négatifs
dta_neg_arrange <- dta_neg %>% arrange(Volatility)
final_dt_negative <- dta_neg_arrange[1, ]  # Initialisation avec le premier point

for (i in 2:nrow(dta_neg_arrange)) {
  if (dta_neg_arrange$Return[i] <= min(final_dt_negative$Return)) {
    final_dt_negative <- rbind(final_dt_negative, dta_neg_arrange[i, ])
  }
}

# Étape 5 : Combiner les deux segments pour obtenir la frontière d'efficience
efficient_frontier <- rbind(final_dt_positive, final_dt_negative) %>% arrange(Return)
```

```
# Fonction pour arrondir les rendements
round_up <- function(x) {
  if (x > 0) {
    return(round(x + 0.1, 1))
    # return(round(x + 0.09, 1))
  } else {
    return(round(x - 0.1, 1))
    # return(round(x - 0.09, 1))
  }
}

global_eff_frontier <- efficient_frontier
global_eff_frontier$Return <- sapply(global_eff_frontier$Return, round_up)

# Étape 6 : Réplication de la frontière d'efficience pour chaque niveau ESG
test_esg <- unique(sort(round(fig_dt$ESG, 1)))
dta_final <- lapply(test_esg, function(esg_val) {
  global_eff_frontier %>%
    mutate(ESG = esg_val)
})

final_dt <- bind_rows(dta_final) %>% arrange(Return)

# Étape 7 : Génération de la matrice de volatilité pour le graphique
grid_return <- unique(final_dt$Return)
grid_esg <- unique(final_dt$ESG)
grid_combinations <- expand.grid(Return = grid_return, ESG = grid_esg)
```

```
volatility_matrix <- final_dt %>%
  dplyr::right_join(grid_combinations, by = c("Return", "ESG")) %>%
  dplyr::group_by(Return, ESG) %>%
  dplyr::summarise(Volatility = mean(Volatility, na.rm = TRUE), .groups = "drop") %>%
  tidyr::pivot_wider(names_from = ESG, values_from = Volatility) %>%
  dplyr::select(-Return) %>%
  as.matrix()

volatility_matrix[is.na(volatility_matrix)] <- NA

# Étape 8 : Création du graphique 3D avec plotly
fig <- plot_ly() %>%
  add_trace(
    data = fig_dt,
    type = 'scatter3d',
    mode = "markers",
    z = ~Volatility,
    y = ~Return,
    x = ~ESG,
    color = ~Profit,
    colors = c('#BF382A', "darkgreen", "#0C4B8E"),
    marker = list(size = 5)
  ) %>%
  add_surface(
    x = ~grid_esg,
    y = ~grid_return,
    z = ~volatility_matrix,
    colorscale = 'Viridis',
    opacity = 0.9, #0.8
```

```
  name = 'Efficient frontier'
 ) %>%
 colorbar(title = "Volatility level") %>%
 layout(
  title = "3D Portfolio Performance with Efficient Frontier",
  scene = list(
   xaxis = list(title = "ESG"),
   yaxis = list(title = "Return"),
   zaxis = list(title = "Volatility")
  ),
  legend = list(
   title = list(text = "Return type"),
   itemsizing = "constant"
  )
 )

# Étape 9 : Tracer la droite reliant les points de rendement maximal et minimal
fig_max_point <- fig_dt[which.max(fig_dt$Return), ]
fig_min_point <- fig_dt[which.min(fig_dt$Return), ]

# Calcul de la pente de la droite max_min_line
slope_return <- (fig_max_point$Return - fig_min_point$Return) /
 (fig_max_point$Volatility - fig_min_point$Volatility)

slope_esg <- (fig_max_point$ESG - fig_min_point$ESG) /
 (fig_max_point$Volatility - fig_min_point$Volatility)

# Étape 10 : Déterminer le point d'efficience avec la plus faible volatilité
reference_point <- efficient_frontier[which.min(efficient_frontier$Volatility), ]
```

```
# Génération des points sur la droite tangente
tangent_return <- seq(fig_min_point$Return,
                      fig_max_point$Return,
                      length.out = 100)

# Calcul de l'ordonnée à l'origine (b)
b <- fig_min_point$Return - slope_return * fig_min_point$Volatility

# Calcul des coordonnées y pour chaque x de tangent_return
tangent_volatility <- (tangent_return - b) / slope_return

# Affichage des résultats
tangent_points <- data.frame(Return = tangent_return, Volatility = tangent_volatility)
# print(tangent_points)

zero_rt_vol_df <- tangent_points%>%dplyr::filter(Return == 0)

if(nrow(zero_rt_vol_df)== 0){
  zero_rt_vol <- (0 - b) / slope_return
  zero_rt_vol_df <- data.frame(Return = 0, Volatility = zero_rt_vol)

  tangent_points <- rbind(tangent_points, zero_rt_vol_df)
}

# Get the diff between reference_point volatility and zero_rt_vol_df volatility
vol_diff = abs(reference_point$Volatility - zero_rt_vol_df$Volatility)

tg_line_ret <- round(tangent_points$Return, 2)
```

```
tg_line_vol <- round((tangent_points$Volatility - vol_diff), 4)

tangent_line <- data.frame(Return = tg_line_ret, Volatility = tg_line_vol)

tangent_line <- tangent_line%>%arrange(Return)

new_grid_return <- unique(tangent_line$Return)

# Step 10: Create the tangent line surface
tg_surface <- lapply(test_esg, function(esg_val) {
  tangent_line %>%
    mutate(ESG = esg_val)
})

tg_surface_df <- bind_rows(tg_surface)

grid_combinations_tg <- expand.grid(Return = unique(tg_surface_df$Return), ESG = unique(tg_surface_df$ESG))

tg_volatility_matrix <- tg_surface_df %>%
  dplyr::right_join(grid_combinations_tg, by = c("Return", "ESG")) %>%
  dplyr::group_by(Return, ESG) %>%
  dplyr::summarise(Volatility = mean(Volatility, na.rm = TRUE), .groups = "drop") %>%
  tidyr::pivot_wider(names_from = ESG, values_from = Volatility) %>%
  dplyr::select(-Return) %>%
  as.matrix()

tg_volatility_matrix[is.na(tg_volatility_matrix)] <- NA
```

```
 fig <- fig %>%
  add_surface(
    z = ~tg_volatility_matrix,
    y = ~new_grid_return,
    x = ~grid_esg,
    colorscale = list(c(0, 1), c("cyan", "cyan")),
    opacity = 0.1,
    showscale = FALSE,
    name = 'Tangent Surface'
  )


 return(fig)
}

#####################################



######## new v. ############
# La nouvelle version de la fonction
# customize_eff_front <- function(fig_dt) {
#   # Vérification si les données existent
#   if (is.null(fig_dt) || nrow(fig_dt) == 0) {
#     return(NULL)
#   }
#
#   # Étape 1 : Filtrer les données en fonction de l'ESG
```

```
#  hist_data <- hist(fig_dt$ESG, plot = FALSE, breaks = 3)
#  filtered_list <- list()
#
#  for (i in 1:(length(hist_data$breaks) - 1)) {
#   if (hist_data$counts[i] > 0) {
#    filtered_data <- fig_dt %>%
#     dplyr::filter(ESG >= hist_data$breaks[i] & ESG < hist_data$breaks[i + 1])
#    filtered_list[[paste0("Class_", i)]] <- filtered_data
#   }
#  }
#
#  # Étape 2 : Séparer les données en rendements positifs et négatifs
#  dta_pos <- fig_dt %>% dplyr::filter(Return >= 0)
#  dta_neg <- fig_dt %>% dplyr::filter(Return < 0)
#
#  # Étape 3 : Organiser les données pour les rendements positifs
#  dta_pos_arrange <- dta_pos %>% arrange(Volatility)
#  final_dt_positive <- dta_pos_arrange[1, ]  # Initialisation avec le premier point
#
#  for (i in 2:nrow(dta_pos_arrange)) {
#   if (dta_pos_arrange$Return[i] >= max(final_dt_positive$Return)) {
#    final_dt_positive <- rbind(final_dt_positive, dta_pos_arrange[i, ])
#   }
#  }
#
#  # Étape 4 : Organiser les données pour les rendements négatifs
#  dta_neg_arrange <- dta_neg %>% arrange(Volatility)
#  final_dt_negative <- dta_neg_arrange[1, ]  # Initialisation avec le premier point
#
```

```
#   for (i in 2:nrow(dta_neg_arrange)) {
#     if (dta_neg_arrange$Return[i] <= min(final_dt_negative$Return)) {
#       final_dt_negative <- rbind(final_dt_negative, dta_neg_arrange[i, ])
#     }
#   }
#
#   # Étape 5 : Combiner les deux segments pour obtenir la frontière d'efficience
#   efficient_frontier <- rbind(final_dt_positive, final_dt_negative) %>% arrange(Return)
#
#   # Fonction pour arrondir les rendements
#   round_up <- function(x) {
#     if (x > 0) {
#       return(round(x + 0.1, 1))
#       # return(round(x + 0.09, 1))
#     } else {
#       return(round(x - 0.1, 1))
#       # return(round(x - 0.09, 1))
#     }
#   }
#
#   global_eff_frontier <- efficient_frontier
#   global_eff_frontier$Return <- sapply(global_eff_frontier$Return, round_up)
#
#   # Étape 6 : Réplication de la frontière d'efficience pour chaque niveau ESG
#   test_esg <- unique(sort(round(fig_dt$ESG, 1)))
#   dta_final <- lapply(test_esg, function(esg_val) {
#     global_eff_frontier %>%
#       mutate(ESG = esg_val)
#   })
```

```
#
#  final_dt <- bind_rows(dta_final) %>% arrange(Return)
#
#  # Étape 7 : Génération de la matrice de volatilité pour le graphique
#  grid_return <- unique(final_dt$Return)
#  grid_volatility <- unique(final_dt$Volatility)
#
#  grid_esg <- unique(final_dt$ESG)
#  grid_combinations <- expand.grid(Return = grid_return, ESG = grid_esg)
#
#  ########### old v. #############
#  # volatility_matrix <- final_dt %>%
#  #   dplyr::right_join(grid_combinations, by = c("Return", "ESG")) %>%
#  #   dplyr::group_by(Return, ESG) %>%
#  #   dplyr::summarise(Volatility = mean(Volatility, na.rm = TRUE), .groups = "drop") %>%
#  #   tidyr::pivot_wider(names_from = ESG, values_from = Volatility) %>%
#  #   dplyr::select(-Return) %>%
#  #   as.matrix()
#  ####################
#
#  ######### new v. ########
#  volatility_matrix <- final_dt %>%
#    dplyr::right_join(grid_combinations, by = c("Return", "ESG")) %>%
#    dplyr::group_by(Return, ESG) %>%
#    dplyr::summarise(Volatility = mean(Volatility, na.rm = TRUE), .groups = "drop") %>%
#    tidyr::pivot_wider(names_from = ESG, values_from = Volatility) %>%
#    dplyr::select(-Return) %>%
#    as.matrix()
#
```

```
#  ###################
#
#  volatility_matrix[is.na(volatility_matrix)] <- NA
#
#  # Étape 8 : Création du graphique 3D avec plotly
#
#  ############ old v. ################
#  # fig <- plot_ly() %>%
#  #  # Ancienn version
#  #  # Cette version est meilleure mais disposition des axes pas bonnes
#  #  add_trace(
#  #    data = fig_dt,
#  #    type = 'scatter3d',
#  #    mode = "markers",
#  #    z = ~Volatility,
#  #    y = ~Return,
#  #    x = ~ESG,
#  #    color = ~Profit,
#  #    colors = c('#BF382A', "darkgreen", "#0C4B8E"),
#  #    marker = list(size = 5)
#  #  ) %>%
#  #  add_surface(
#  #    x = ~grid_esg,
#  #    y = ~grid_return,
#  #    z = ~volatility_matrix,
#  #    colorscale = 'Viridis',
#  #    opacity = 0.9, #0.8
#  #    name = 'Efficient frontier'
#  #  ) %>%
```

```
#  #  colorbar(title = "Volatility level") %>%
#  #  layout(
#  #    title = "3D Portfolio Performance with Efficient Frontier",
#  #    scene = list(
#  #      xaxis = list(title = "ESG"),
#  #      yaxis = list(title = "Return"),
#  #      zaxis = list(title = "Volatility")
#  #    ),
#  #    legend = list(
#  #      title = list(text = "Return type"),
#  #      itemsizing = "constant"
#  #    )
#  #  )
#
#  ####################################
#
#
#  ########### new v. ###############
#  fig <- plot_ly() %>%
#    add_trace(
#      data = fig_dt,
#      type = 'scatter3d',
#      mode = "markers",
#      x = ~Volatility,  # Risque sur l'axe horizontal
#      y = ~Return,      # Rendement sur l'axe vertical
#      z = ~ESG,         # ESG sur le dernier axe
#      color = ~Profit,
#      colors = c('#BF382A', "darkgreen", "#0C4B8E"),
#      marker = list(size = 5)
```

```
#   )%>%
#   add_surface(
#     x = ~grid_volatility,  # Remplacez "grid_esg" par "grid_volatility" (Risque)
#     y = ~grid_return,     # Rendement
#     z = ~volatility_matrix, # ESG devient maintenant le dernier axe
#     colorscale = 'Viridis',
#     opacity = 0.9,
#     name = 'Efficient frontier'
#   )%>%
#   colorbar(title = "Volatility level") %>%
#   layout(
#     title = "3D Portfolio Performance with Efficient Frontier",
#     scene = list(
#       xaxis = list(title = "Risk"),
#       yaxis = list(title = "Return"),
#       zaxis = list(title = "ESG Score"),
#       aspectmode = "manual",
#       aspectratio = list(x = 1, y = 1, z = 0.8), # Ajustez les proportions si nécessaire
#       camera = list(
#         eye = list(x = 1.5, y = 1.5, z = 1.5)  # Vue initiale
#       )
#     ),
#     legend = list(
#       title = list(text = "Return type"),
#       itemsizing = "constant"
#     )
#   )
#
#  ############
```

```
#
#  # Étape 9 : Tracer la droite reliant les points de rendement maximal et minimal
#  fig_max_point <- fig_dt[which.max(fig_dt$Return), ]
#  fig_min_point <- fig_dt[which.min(fig_dt$Return), ]
#
#  # Calcul de la pente de la droite max_min_line
#  slope_return <- (fig_max_point$Return - fig_min_point$Return) /
#    (fig_max_point$Volatility - fig_min_point$Volatility)
#
#  slope_esg <- (fig_max_point$ESG - fig_min_point$ESG) /
#    (fig_max_point$Volatility - fig_min_point$Volatility)
#
#  # Étape 10 : Déterminer le point d'efficience avec la plus faible volatilité
#  reference_point <- efficient_frontier[which.min(efficient_frontier$Volatility), ]
#
#  # Génération des points sur la droite tangente
#  tangent_return <- seq(fig_min_point$Return,
#                        fig_max_point$Return,
#                        length.out = 100)
#
#  # Calcul de l'ordonnée à l'origine (b)
#  b <- fig_min_point$Return - slope_return * fig_min_point$Volatility
#
#  # Calcul des coordonnées y pour chaque x de tangent_return
#  tangent_volatility <- (tangent_return - b) / slope_return
#
#  # Affichage des résultats
#  tangent_points <- data.frame(Return = tangent_return, Volatility = tangent_volatility)
#  # print(tangent_points)
```

```
#
#  zero_rt_vol_df <- tangent_points%>%dplyr::filter(Return == 0)
#
#  if(nrow(zero_rt_vol_df)== 0){
#   zero_rt_vol <- (0 - b) / slope_return
#   zero_rt_vol_df <- data.frame(Return = 0, Volatility = zero_rt_vol)
#
#   tangent_points <- rbind(tangent_points, zero_rt_vol_df)
#  }
#
#  # Get the diff between reference_point volatility and zero_rt_vol_df volatility
#  vol_diff = abs(reference_point$Volatility - zero_rt_vol_df$Volatility)
#
#  tg_line_ret <- round(tangent_points$Return, 2)
#
#  tg_line_vol <- round((tangent_points$Volatility - vol_diff), 4)
#
#  tangent_line <- data.frame(Return = tg_line_ret, Volatility = tg_line_vol)
#
#  tangent_line <- tangent_line%>%arrange(Return)
#
#  new_grid_return <- unique(tangent_line$Return)
#
#  # new
#  new_grid_volatility <- unique(tangent_line$Volatility)
#
#
#  # Step 10: Create the tangent line surface
#  tg_surface <- lapply(test_esg, function(esg_val) {
```

```
#    tangent_line %>%
#      mutate(ESG = esg_val)
#  })
#
#  tg_surface_df <- bind_rows(tg_surface)
#
#        grid_combinations_tg <- expand.grid(Return = unique(tg_surface_df$Return), ESG = unique(tg_surface_df$ESG))
#
#  tg_volatility_matrix <- tg_surface_df %>%
#    dplyr::right_join(grid_combinations_tg, by = c("Return", "ESG")) %>%
#    dplyr::group_by(Return, ESG) %>%
#    dplyr::summarise(Volatility = mean(Volatility, na.rm = TRUE), .groups = "drop") %>%
#    tidyr::pivot_wider(names_from = ESG, values_from = Volatility) %>%
#    dplyr::select(-Return) %>%
#    as.matrix()
#
#  tg_volatility_matrix[is.na(tg_volatility_matrix)] <- NA
#
#  ############ old v. ################
#  # fig <- fig %>%
#  #  add_surface(
#  #    z = ~tg_volatility_matrix,
#  #    y = ~new_grid_return,
#  #    x = ~grid_esg,
#  #    colorscale = list(c(0, 1), c("cyan", "cyan")),
#  #    opacity = 0.1,
#  #    showscale = FALSE,
#  #    name = 'Tangent Surface'
```

```
#  #  )
#  ##############################
#
#  ########### new v. ###############
#  fig <- fig %>%
#    add_surface(
#      z = ~tg_volatility_matrix,
#      x = ~new_grid_volatility,  # Risque (Volatility)
#      y = ~new_grid_return,     # Rendement (Return)
#      colorscale = list(c(0, 1), c("cyan", "cyan")),
#      opacity = 0.1,
#      showscale = FALSE,
#      name = 'Tangent Surface'
#    )
#  ##############################
#
#
#  # Le nouveau surplus
#  # # Ajout d'une disposition fixe des axes dans la mise en page
#  # fig <- fig %>%
#  #  layout(
#  #    title = "3D Portfolio Performance with Efficient Frontier",
#  #    scene = list(
#  #      xaxis = list(
#  #        title = list(text = "Risk"),
#  #        autorange = TRUE,   # Ajuste automatiquement l'échelle si nécessaire
#  #        showgrid = TRUE,    # Affiche une grille pour une meilleure lisibilité
#  #        zeroline = FALSE    # Supprime la ligne de zéro (pas pertinent ici)
#  #      ),
```

```
#  #     yaxis = list(
#  #       title = list(text = "Return"),
#  #       autorange = TRUE,
#  #       showgrid = TRUE,
#  #       zeroline = TRUE
#  #     ),
#  #     zaxis = list(
#  #       title = list(text = "ESG"),
#  #       autorange = TRUE,
#  #       showgrid = TRUE,
#  #       zeroline = FALSE
#  #     )
#  #   ),
#  #   legend = list(
#  #     title = list(text = "Return type"),
#  #     itemsizing = "constant"
#  #   )
#  # )
#
#
#  return(fig)
# }

################## FIN NOUVELLE VERSION ######################



# Function to calculate Sharpe ratio per observation
calc_sharpe_ratio <- function(return, rf, annualization_factor) {
```

```
 excess_return <- return - rf
 annualized_return <- excess_return * annualization_factor
 sharpe_ratio <- annualized_return / (sd(excess_return) * sqrt(annualization_factor))
 return(sharpe_ratio)
}

# Function to calculate modified Sharpe ratio per observation
# calc_modified_sresg <- function(returns, rf, esg, sc, annualization_factor) {
#   sharpe_ratio <- calc_sharpe_ratio(returns, rf, annualization_factor)
#   ifelse(sharpe_ratio > 0,
#          sharpe_ratio * (1 + esg)^sc,
#          sharpe_ratio * (1 + esg)^(-sc))
# }
calc_modified_sresg <- function(return, rf, esg, sc, annualization_factor) {
 sharpe_ratio <- calc_sharpe_ratio(return, rf, annualization_factor)
 modified_sr <- ifelse(sharpe_ratio > 0,
                 sharpe_ratio * (1 + esg)^sc,
                 sharpe_ratio * (1 + esg)^(-sc))
 return(modified_sr)
}



# calculate_modified_sharpe

# calc_rolling_sresg <- function(df, sc, rf, esg_col, type_return = "daily") {
#   annualization_factor <- switch(type_return,
#                          "daily" = 252,
#                          "weekly" = 52,
#                          "monthly" = 12,
```

```
#                         "yearly" = 1)
#
#   # Calculate the rolling mean and standard deviation
#   df <- df %>%
#     mutate(
#       rolling_mean_ticker = rollapply(Ticker_return, width = 252, FUN = mean, fill = NA, align = 'right'),
#       rolling_sd_ticker = rollapply(Ticker_return, width = 252, FUN = sd, fill = NA, align = 'right'),
#       rolling_mean_rf = rollapply(rf, width = 252, FUN = mean, fill = NA, align = 'right')
#     ) %>%
#     dplyr::filter(!is.na(rolling_mean_ticker), !is.na(rolling_sd_ticker), !is.na(rolling_mean_rf)) %>%
#     mutate(
#       annualized_return = rolling_mean_ticker * annualization_factor,
#       annualized_rf = rolling_mean_rf * annualization_factor,
#       annualized_volatility = rolling_sd_ticker * sqrt(annualization_factor),
#       Sharpe_Ratio = (annualized_return - annualized_rf) / annualized_volatility,
#       Modified_SR = ifelse(Sharpe_Ratio > 0,
#                            Sharpe_Ratio * (1 + !!sym(esg_col))^sc,
#                            Sharpe_Ratio * (1 + !!sym(esg_col))^(-sc))
#     )
#
#   return(df)
# }



my_color <-c("blue", "red", "yellow", "pink", "royalblue", "darkgreen")
options(spinner.color="#ffffff", spinner.color.background ="#ffffff", spinner.size = 2)

# Fonctionne bien pour obtenir les données des esg
get_historic_esg <- function(ticker, max_retries = 5, retry_delay = 1) {
```

```
 headers <- c('User-Agent' = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36')
 url <- 'https://query2.finance.yahoo.com/v1/finance/esgChart'
 params <- list(symbol = ticker)

 for (retry in 1:max_retries) {
  tryCatch({
   response <- httr::GET(url, query = params, httr::add_headers(headers))
   content <- httr::content(response, "text")
   json <- jsonlite::fromJSON(content #, flatten = TRUE
   )
   df <- data.frame(json$esgChart$result$symbolSeries)
   # df$Date <- as.POSIXct(df$timestamp[[1]], origin="1970-01-01")

   df <- data.frame(
    timestamp = unlist(df$timestamp),
    esgScore = unlist(df$esgScore),
    governanceScore = unlist(df$governanceScore),
    environmentScore = unlist(df$environmentScore),
    socialScore = unlist(df$socialScore)
   )

   df$Date <- as.Date(as.POSIXct(df$timestamp, origin = "1970-01-01"))

   names(df) <- c("timestamp", "Total-Score", "E-Score", "S-Score", "G-Score", "Date")
   df <- dplyr::select(df, Date, `Total-Score`, `E-Score`, `S-Score`, `G-Score`)
   # rownames(df) <- df$Date
   # df$Date <- NULL
   return(df)
```

```
  }, error = function(e) {
    if (retry < max_retries) {
      Sys.sleep(retry_delay)
    } else {
      print('Max retries reached. Please check the ticker symbol or try again later.')
      # stop("Failed to retrieve data after multiple retries. Please check your network connection and try again.")
      return(NULL)
    }
  }, warning = function(w) {
    if (retry < max_retries) {
      Sys.sleep(retry_delay)
    } else {
      warning("Warnings encountered and data could not be retrieved. Please check your network connection and try again.")
    }
  })
 }


}


# Fonctionne aussi
# en utilisant l'option html_read
# get_historic_esg <- function(ticker, max_retries = 5, retry_delay = 1) {
#   headers <- c('User-Agent' = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36')
#   # url <- 'https://query2.finance.yahoo.com/v1/finance/esgChart'
#   # params <- list(symbol = ticker)
#
#   url <- glue::glue("https://query2.finance.yahoo.com/v1/finance/esgChart?symbol={ticker}")
```

```
#
#  for (retry in 1:max_retries) {
#   tryCatch({
#    # response <- httr::GET(url, query = params, add_headers(headers))
#    # content <- httr::content(response, "text")
#    # json <- jsonlite::fromJSON(content #, flatten = TRUE
#    # )
#    response <- read_html(url) %>% html_node("body") %>% rvest::html_text()
#    # content <- content(response, "text")
#
#    json <- fromJSON(response #, flatten = TRUE
#    )
#    df <- data.frame(json$esgChart$result$symbolSeries)
#    # df$Date <- as.POSIXct(df$timestamp[[1]], origin="1970-01-01")
#
#    df <- data.frame(
#     timestamp = unlist(df$timestamp),
#     esgScore = unlist(df$esgScore),
#     governanceScore = unlist(df$governanceScore),
#     environmentScore = unlist(df$environmentScore),
#     socialScore = unlist(df$socialScore)
#    )
#
#    df$Date <- as.Date(as.POSIXct(df$timestamp, origin = "1970-01-01"))
#
#    names(df) <- c("timestamp", "Total-Score", "E-Score", "S-Score", "G-Score", "Date")
#    df <- dplyr::select(df, Date, `Total-Score`, `E-Score`, `S-Score`, `G-Score`)
#    # rownames(df) <- df$Date
#    # df$Date <- NULL
```

```
#      return(df)
#    }, error = function(e) {
#      if (retry < max_retries) {
#        Sys.sleep(retry_delay)
#      } else {
#        print('Max retries reached. Please check the ticker symbol or try again later.')
#       # stop("Failed to retrieve data after multiple retries. Please check your network connection and try again.")
#        return(NULL)
#      }
#    }, warning = function(w) {
#      if (retry < max_retries) {
#        Sys.sleep(retry_delay)
#      } else {
#         warning("Warnings encountered and data could not be retrieved. Please check your network connection and try again.")
#      }
#    })
#  }
#
# }


get_esg_data <- function(ticker, full = TRUE) {


 ticker = toupper(ticker)


 if(all(ticker == "")){
   # if(length(ticker) == 1  && ticker == ''){
   return(NULL)
```

```
}

 httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))

 # Fonction pour obtenir les données ESG pour un seul ticker
 get_esg_for_ticker <- function(ticker, full = TRUE) {
  # httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))

  UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"

  url = glue::glue("https://finance.yahoo.com/quote/{ticker}/sustainability/")

  # print("--url of get esg ticker--")
  # print(url)
  # url = paste0("https://finance.yahoo.com/quote/", ticker, "/sustainability/")
  # url <- "https://finance.yahoo.com/quote/RELIANCE.NS/sustainability/"

  cookies = c(
   GUCS = "AXAaNFgz",
   GUC = "AQABCAFnLLFnXEIeOgSl&s=AQAAAD8j2FLC&g=ZytikA",
   EuConsent = "CQFm5cAQFm5cAAOACKFRBOFgAAAAAAAAACiQAAAAAAAA",
   A1 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAGxLGdcZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAq5W1hyNRjdnqqQ3_Y1KWsg",
   A3 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAGxLGdcZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAq5W1hyNRjdnqqQ3_Y1KWsg",
   A1S = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAGxLGdcZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAq5W1hyNRjdnqqQ3_Y1KWsg",
   cmp = "t=1730896521&j=1&u=1---&v=51",
```

```
    PRF =
"t=PRU%2BETGLX%2BKO%2BMETA%2B0P000023MW.L%2BCPRI%2BAAA%2B0P0000XPCJ.F%2B0P0001FG1E.F%2BMAMI.F%2B8AECQ.MI%2BZPDX.MU%2BZPDX.BE%2BZPDX.DE%2BZPDXD.XD&newChartbetateaser=0%2C1719358224417"
  )


  # Faire la requête GET avec les en-têtes spécifiés
  response <- httr::GET(url,
                , httr::set_cookies(.cookies = cookies),
                add_headers(`Connection` = "keep-alive", `User-Agent` = UA))


  # response <- GET(
  #   url,
  #   add_headers(
  #     `accept` =
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
  #     `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",
  #     `cache-control` = "no-cache",
  #     `pragma` = "no-cache",
  #     `priority` = "u=0, i",
  #     `sec-ch-ua` = "\"Google Chrome\";v=\"125\", \"Chromium\";v=\"125\",
\"Not.A/Brand\";v=\"24\"",
  #     `sec-ch-ua-mobile` = "?0",
  #     `sec-ch-ua-platform` = "\"Windows\"",
  #     `sec-fetch-dest` = "document",
  #     `sec-fetch-mode` = "navigate",
  #     `sec-fetch-site` = "none",
  #     `sec-fetch-user` = "?1",
  #     `upgrade-insecure-requests` = "1"
  #   )
```

```
  # )
  esg_not_available = paste0("https://finance.yahoo.com/quote/", ticker, "/")

  if(response$url == esg_not_available){

   esg_prod_envol = structure(list(Info = c("Alcoholic Beverages", "Adult Entertainment",
                                            "Gambling", "Tobacco Products", "Animal Testing", "Fur and Specialty Leather",
                                            "Controversial Weapons", "Small Arms", "Catholic Values", "GMO",
                                            "Military Contracting", "Pesticides", "Thermal Coal", "Palm Oil"
   ), Detail = "-"), row.names = c(NA, -14L), class = c("tbl_df",
                                                        "tbl", "data.frame"))

   df = structure(list(Info = c("Ticker",
                                "Total ESG Risk Score",
                                "Environmental Risk Score",
                                "Social Risk Score",
                                "Governance Risk Score",
                                "Actual performance",
                                "Peer average score",
                                "Controversy score",
                                "Controversy level",
                                "Last update"), Detail = c(ticker, rep('-',9))),
                  row.names = c(NA, -10L), class = c("tbl_df",
                                                     "tbl", "data.frame"))


  }else{
   content <- content(response, "text")
```

```
    html_parsed <- rvest::read_html(response)

    tables = html_parsed %>%
      rvest::html_nodes('table') %>%
      rvest::html_table()

    if (length(tables) %in% c(1, 2)){
      # if (length(tables) >= 1 && length(tables) <= 2){
      if(length(tables) == 1){
        esg_for_peers = tables[[1]]

        esg_prod_envol = structure(list(Info = c("Alcoholic Beverages", "Adult Entertainment",
                                                 "Gambling", "Tobacco Products", "Animal Testing", "Fur and Specialty Leather",
                                                 "Controversial Weapons", "Small Arms", "Catholic Values", "GMO",
                                                 "Military Contracting", "Pesticides", "Thermal Coal", "Palm Oil"
        ), Detail = "-"), row.names = c(NA, -14L), class = c("tbl_df",
                                                             "tbl", "data.frame"))
        the_act_type = "-"
        controversy_score = "-"
        peer_avg_score = "-"
        controversy_level_text = "-"
        Last_update = "-"

      }else{
        esg_for_peers = tables[[1]]
        esg_prod_envol = tables[[2]]
        # names(esg_prod_envol) = c('Info', 'Detail')
```

```
names(esg_prod_envol) = c("Products and Activities", "Significant Involvement")


# controversy_score = html_parsed %>%rvest::html_nodes('.controversy-score.svelte-ye6fz0') %>%

#   rvest::html_text()


# Type of Risk

# the_act_type = html_parsed %>%rvest::html_nodes('.perf.svelte-y3c2sq') %>%

#   rvest::html_text()


the_act_type = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[1]/div/span') %>%

 rvest::html_text()


# Actual performance

# actual_perf =  html_parsed %>%rvest::html_nodes('.scoreRank.svelte-y3c2sq') %>%

#   rvest::html_text()



controversy_score = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[3]/section[1]/div/div[2]/span[1]') %>%

 rvest::html_text()


# peer_avg_score = html_parsed %>%rvest::html_nodes('.peer-score.svelte-ye6fz0') %>%

#   rvest::html_text()


peer_avg_score = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[3]/section[1]/div/span[2]/div[2]/div[2]/div[2]/span') %>%

 rvest::html_text()
```

```
# controversy_level_text = (html_parsed %>%
#                     rvest::html_nodes('.val.svelte-ye6fz0 span') %>%
#                     rvest::html_text())[3]

controversy_level_text = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[3]/section[1]/div/div[2]/span[3]') %>%
 rvest::html_text()

# Ou
# controversy_level <- rvest::read_html(response)%>%
#   html_nodes('.val.svelte-ye6fz0') %>%
#   html_text()

# controversy_level_text <- str_replace_all(controversy_level, "\\d+", "") %>%
#   str_trim()

# updating_info = (html_parsed %>%rvest::html_nodes('.content.svelte-ye6fz0 span') %>%
#                rvest::html_text())[1]

updating_info = html_parsed %>%rvest::html_nodes('#nimbus-app > section > section > section > article > section:nth-child(3) > section:nth-child(1) > div > span:nth-child(1)') %>%
 rvest::html_text()

date_pattern = "\\d+/\\d+"

Last_update = str_extract(updating_info, date_pattern)

Last_update = as.Date(paste0(Last_update, "/01"), format="%m/%Y/%d")
```

```
 Last_update = format(Last_update, "%m/%Y")
}



# Obtenir
# yf_tot_esg_score = html_parsed %>%rvest::html_nodes('.border.svelte-y3c2sq') %>%
#  rvest::html_text()

yf_tot_esg_score = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[1]/div/div/h4') %>%
 rvest::html_text()

# E_grade = as.numeric(gsub(" ", "", actual_perf[2]))

# S_grade = as.numeric(gsub(" ", "", actual_perf[3]))

# G_grade = as.numeric(gsub(" ", "", actual_perf[4]))

E_grade = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[2]/div/div/h4') %>%
 rvest::html_text()%>%as.numeric()

S_grade = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[3]/div/div/h4') %>%
 rvest::html_text()%>%as.numeric()

G_grade = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[4]/div/div/h4') %>%
 rvest::html_text()%>%as.numeric()
```

```
# partial_sever = rvest::read_html(response)%>%
#   html_nodes('.partial.svelte-he3t4n') %>%
#   html_text()

# Ici on a trois éléments
# "None"   "4"     "Severe"
# list_range_sever = rvest::read_html(response)%>%
#   html_nodes('.scale.svelte-he3t4n span') %>%
#   html_text()

df = as.data.frame(matrix(NA, ncol = 2, nrow = 10))
names(df) = c('Info', 'Detail')

df$Info = c("Ticker",
        "Total ESG Risk Score",
        "Environmental Risk Score",
        "Social Risk Score",
        "Governance Risk Score",
        "Actual performance",
        "Peer average score",
        "Controversy score",
        "Controversy level",
        "Last update")

df$Detail = c(ticker,
         yf_tot_esg_score,
         E_grade,
         S_grade,
         G_grade,
```

```
            the_act_type,
            peer_avg_score,
            controversy_score,
            controversy_level_text,
            Last_update)

    }else{
     esg_prod_envol = structure(list(Info = c("Alcoholic Beverages", "Adult Entertainment",
                                              "Gambling", "Tobacco Products", "Animal Testing", "Fur and Specialty Leather",
                                              "Controversial Weapons", "Small Arms", "Catholic Values", "GMO",
                                              "Military Contracting", "Pesticides", "Thermal Coal", "Palm Oil"
     ), Detail = "-"), row.names = c(NA, -14L), class = c("tbl_df",
                                                          "tbl", "data.frame"))


     df = structure(list(Info = c("Ticker",
                                  "Total ESG Risk Score",
                                  "Environmental Risk Score",
                                  "Social Risk Score",
                                  "Governance Risk Score",
                                  "Actual performance",
                                  "Peer average score",
                                  "Controversy score",
                                  "Controversy level",
                                  "Last update"), Detail = c(ticker, rep('-',9))),
                    row.names = c(NA, -10L), class = c("tbl_df",
                                                       "tbl", "data.frame"))

    }
```

```
 } #in case sustainability exist



 if(full){

  if(!is.null(esg_prod_envol)){
   names(esg_prod_envol) = c('Info', 'Detail')

   final_df = rbind(df, esg_prod_envol)

   # print("--final df--")
   # print(final_df)

   return(final_df)

  }else{
   return(df)
  }

 } else{

  return(df)
 }

}

for (i in 1:length(ticker)) {
 if(i == 1){
```

```
   final_data = get_esg_for_ticker(ticker[i], full= full)

 }else{

   final_dt = get_esg_for_ticker(ticker[i], full= full)

   final_data = cbind(final_data, final_dt[, 'Detail'])

 }


}

# Rename final data
names(final_data)[2:length(final_data)] = final_data[1, 2:length(final_data)]
# remoove first row
final_data = final_data[-1,]

# print("--Controversy Level--")
# print(final_data)

# print("--final_data[8, ]--")
# print(final_data[8, ])

# print("--test sur contro--")
# print(gsub("Controversy Level", "", final_data[8, ]))

# if(final_data[8, 2][1] != "-"){
#   final_data[8, ] = gsub("Controversy Level", "", final_data[8, ])
```

```
 #   print("--rien à faire--")
 # }



 # set first colomn to rownname
 # rownames(final_data) = final_data[,1]
 #
 # final_data = final_data[,-1]

 final_data = final_data %>%
   # tibble::as_tibble()%>%
   # tibble::as.tibble()%>%
   as_tibble()%>%
   column_to_rownames(var = "Info")

 return(final_data)

}

# Comparaison entre les ESG
# Version améliorée
# get_esg_data <- function(ticker, full = TRUE) {
#
#   ticker = toupper(ticker)
#
#   if(all(ticker == "")){
#   # if(length(ticker) == 1  && ticker == ''){
#     return(NULL)
#   }
```

```
#
#         # httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))
#
#  # Fonction pour obtenir les données ESG pour un seul ticker
#  get_esg_for_ticker <- function(ticker, full = TRUE) {
#             # httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))
#
#    UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"
#
#    # url = glue::glue("https://finance.yahoo.com/quote/{ticker}/sustainability/")
#    url = paste0("https://finance.yahoo.com/quote/", ticker, "/sustainability/")
#
#
#    # url = paste0("https://finance.yahoo.com/quote/", ticker, "/sustainability/")
#    # url <- "https://finance.yahoo.com/quote/RELIANCE.NS/sustainability/"
#
#    cookies = c(
#      GUCS = "AXAaNFgz",
#      GUC = "AQABCAFnLLFnXEIeOgSl&s=AQAAAD8j2FLC&g=ZytikA",
#      EuConsent = "CQFm5cAQFm5cAAOACKFRBOFgAAAAAAAAACiQAAAAAAAA",
#                    A1    =    "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAGxLGdcZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAq5W1hyNRjdnqqQ3_Y1KWsg",
#                    A3    =    "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAGxLGdcZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAq5W1hyNRjdnqqQ3_Y1KWsg",
#                   A1S    =    "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAGxLGdcZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAq5W1hyNRjdnqqQ3_Y1KWsg",
#      cmp = "t=1730896521&j=1&u=1---&v=51",
#                                                                                PRF    =    "t=PRU%2BETGLX%2BKO%2BMETA%2B0P000023MW.L%2BCPRI%2BAAA%2B0P0000XPCJ.F%2B
```

```
0P0001FG1E.F%2BMAMI.F%2B8AECQ.MI%2BZPDX.MU%2BZPDX.BE%2BZPDX.DE%2BZPDXD.XD
&newChartbetateaser=0%2C1719358224417"

#   )

#

#   # Faire la requête GET avec les en-têtes spécifiés

#   response <- httr::GET(url,

#                 , httr::set_cookies(.cookies = cookies),

#                 add_headers(`Connection` = "keep-alive", `User-Agent` = UA))

#

#   # response <- httr::GET(

#   #  url,

#   #  add_headers(

#                                                 #                                                 `accept`          =
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.7",

#   #    `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",

#   #    `cache-control` = "no-cache",

#   #    `pragma` = "no-cache",

#   #    `priority` = "u=0, i",

#           #           `sec-ch-ua`   =   "\"Google   Chrome\";v=\"125\",   \"Chromium\";v=\"125\",
\"Not.A/Brand\";v=\"24\"",

#   #    `sec-ch-ua-mobile` = "?0",

#   #    `sec-ch-ua-platform` = "\"Windows\"",

#   #    `sec-fetch-dest` = "document",

#   #    `sec-fetch-mode` = "navigate",

#   #    `sec-fetch-site` = "none",

#   #    `sec-fetch-user` = "?1",

#   #    `upgrade-insecure-requests` = "1"

#   #  )

#   # )
```

```
#
#
#   content <- content(response, "text")
#
#   html_parsed <- read_html(response)
#
#   tables = html_parsed %>%
#     rvest::html_nodes('table') %>%
#     rvest::html_table()
#
#
#   if (base::length(tables) == 2){
#     esg_for_peers = tables[[1]]
#     esg_prod_envol = tables[[2]]
#     # names(esg_prod_envol) = c('Info', 'Detail')
#     names(esg_prod_envol) = c("Products and Activities", "Significant Involvement")
#
#     # Obtenir
#     # yf_tot_esg_score = html_parsed %>%rvest::html_nodes('.border.svelte-y3c2sq') %>%
#     #   rvest::html_text()
#
#               yf_tot_esg_score = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[1]/div/div/h4') %>%
#       rvest::html_text()
#
#     # Type of Risk
#     # the_act_type = html_parsed %>%rvest::html_nodes('.perf.svelte-y3c2sq') %>%
#     #   rvest::html_text()
#
```

```
#                the_act_type = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[1]/div/span') %>%
#       rvest::html_text()
#
#     # Actual performance
#     # actual_perf =  html_parsed %>%rvest::html_nodes('.scoreRank.svelte-y3c2sq') %>%
#     #   rvest::html_text()
#
#     # E_grade = as.numeric(gsub(" ", "", actual_perf[2]))
#
#     # S_grade = as.numeric(gsub(" ", "", actual_perf[3]))
#
#     # G_grade = as.numeric(gsub(" ", "", actual_perf[4]))
#
#                 E_grade = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[2]/div/div/h4') %>%
#       rvest::html_text()%>%as.numeric()
#
#                 S_grade = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[3]/div/div/h4') %>%
#       rvest::html_text()%>%as.numeric()
#
#                 G_grade = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[2]/section[1]/div/section[4]/div/div/h4') %>%
#       rvest::html_text()%>%as.numeric()
#
#     # controversy_score = html_parsed %>%rvest::html_nodes('.controversy-score.svelte-ye6fz0') %>%
#     #   rvest::html_text()
#
```

```
#
#            controversy_score = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[3]/section[1]/div/div[2]/span[1]') %>%
#      rvest::html_text()
#
#
#     # peer_avg_score = html_parsed %>%rvest::html_nodes('.peer-score.svelte-ye6fz0') %>%
#     #  rvest::html_text()
#
#              peer_avg_score = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[3]/section[1]/div/span[2]/div[2]/div[2]/div[2]/span') %>%
#      rvest::html_text()
#
#     # partial_sever = rvest::read_html(response)%>%
#     #  rvest::html_nodes('.partial.svelte-he3t4n') %>%
#     #  rvest::html_text()
#
#     # Ici on a trois éléments
#     # "None"   "4"     "Severe"
#     # list_range_sever = rvest::read_html(response)%>%
#     #  rvest::html_nodes('.scale.svelte-he3t4n span') %>%
#     #  rvest::html_text()
#
#     # controversy_level_text = (html_parsed %>%
#     #                   rvest::html_nodes('.val.svelte-ye6fz0 span') %>%
#     #                   rvest::html_text())[3]
#
#          controversy_level_text = html_parsed %>%rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[3]/section[1]/div/div[2]/span[3]') %>%
```

```
#       rvest::html_text()
#
#     # Ou
#     # controversy_level <- rvest::read_html(response)%>%
#     #   rvest::html_nodes('.val.svelte-ye6fz0') %>%
#     #   rvest::html_text()
#
#     # controversy_level_text <- str_replace_all(controversy_level, "\\d+", "") %>%
#     #   str_trim()
#
#     # updating_info = (html_parsed %>%rvest::html_nodes('.content.svelte-ye6fz0 span') %>%
#     #                  rvest::html_text())[1]
#
#    updating_info = html_parsed %>%rvest::html_nodes('#nimbus-app > section > section > section
> article > section:nth-child(3) > section:nth-child(1) > div > span:nth-child(1)') %>%
#       rvest::html_text()
#
#     date_pattern = "\\d+/\\d+"
#
#     Last_update = str_extract(updating_info, date_pattern)
#
#     Last_update = as.Date(paste0(Last_update, "/01"), format="%m/%Y/%d")
#     Last_update = format(Last_update, "%m/%Y")
#
#     df = as.data.frame(matrix(NA, ncol = 2, nrow = 10))
#     names(df) = c('Info', 'Detail')
#
#     df$Info = c("Ticker",
#                 "Total ESG Risk Score",
```

```
#              "Environmental Risk Score",
#              "Social Risk Score",
#              "Governance Risk Score",
#              "Actual performance",
#              "Peer average score",
#              "Controversy score",
#              "Controversy level",
#              "Last update")
#
#     df$Detail = c(ticker,
#                   yf_tot_esg_score,
#                   E_grade,
#                   S_grade,
#                   G_grade,
#                   the_act_type,
#                   peer_avg_score,
#                   controversy_score,
#                   controversy_level_text,
#                   Last_update)
#
#    }else{
#     esg_prod_envol = structure(list(Info = c("Alcoholic Beverages", "Adult Entertainment",
#                                              "Gambling", "Tobacco Products", "Animal Testing", "Fur and Specialty
Leather",
#                                    "Controversial Weapons", "Small Arms", "Catholic Values", "GMO",
#                                    "Military Contracting", "Pesticides", "Thermal Coal", "Palm Oil"
#     ), Detail = "-"), row.names = c(NA, -14L), class = c("tbl_df",
#                                                          "tbl", "data.frame"))
#
```

```
#     df = structure(list(Info = c("Ticker",
#                                  "Total ESG Risk Score",
#                                  "Environmental Risk Score",
#                                  "Social Risk Score",
#                                  "Governance Risk Score",
#                                  "Actual performance",
#                                  "Peer average score",
#                                  "Controversy score",
#                                  "Controversy level",
#                                  "Last update"), Detail = c(ticker, rep('-',9))),
#                row.names = c(NA, -10L), class = c("tbl_df",
#                                                   "tbl", "data.frame"))
#
#   }
#
#   if(full){
#
#     if(!is.null(esg_prod_envol)){
#       names(esg_prod_envol) = c('Info', 'Detail')
#       esg_prod_envol = as.data.frame(esg_prod_envol)
#       print("--Ncol df--")
#       print(NCOL(df))
#       print("--Ncol esg prod--")
#       print(NCOL(esg_prod_envol))
#       # df = as.data.frame(df)
#
#       final_df = rbind(df, esg_prod_envol)
#
#       # final_df = rbind(df[, c(1,2)], esg_prod_envol[, c(1,2)])
```

```
#
#       return(final_df)
#
#     }else{
#       return(df)
#     }
#
#   } else{
#
#     return(df)
#   }
#
# }
#
#  for (i in 1:length(ticker)) {
#   if(i == 1){
#
#     final_data = get_esg_for_ticker(ticker[i], full= full)
#
#   }else{
#
#     final_dt = get_esg_for_ticker(ticker[i], full= full)
#
#     final_data = cbind(final_data, final_dt[, 'Detail'])
#
#   }
#
#
# }
```

```
#
#   # Rename final data
#   names(final_data)[2:length(final_data)] = final_data[1, 2:length(final_data)]
#   # remoove first row
#   final_data = final_data[-1,]
#
#   final_data[8, ] = gsub("Controversy Level", "", final_data[8, ])
#   # set first colomn to rownname
#   # rownames(final_data) = final_data[,1]
#   #
#   # final_data = final_data[,-1]
#
#   final_data = final_data %>%
#     # tibble::as_tibble()%>%
#     # tibble::as.tibble()%>%
#     as_tibble()%>%
#     column_to_rownames(var = "Info")
#
#   return(final_data)
#
# }


# Ancienne version
# get_esg_data = function(ticker, full = TRUE) {
#
#  httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))
#
#   # Fonction pour obtenir les données ESG pour un seul ticker
```

```
#  get_esg_for_ticker <- function(ticker, full = TRUE) {
#                 httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))
#
#   url = glue::glue("https://finance.yahoo.com/quote/{ticker}/sustainability/")
#   # url = paste0("https://finance.yahoo.com/quote/", ticker, "/sustainability/")
#   # url <- "https://finance.yahoo.com/quote/RELIANCE.NS/sustainability/"
#
#   # Faire la requête GET avec les en-têtes spécifiés
#   response <- httr::GET(
#     url,
#     add_headers(
#                                                                 `accept` = "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
#       `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",
#       `cache-control` = "no-cache",
#       `pragma` = "no-cache",
#       `priority` = "u=0, i",
#                       `sec-ch-ua` = "\"Google Chrome\";v=\"125\", \"Chromium\";v=\"125\", \"Not.A/Brand\";v=\"24\"",
#       `sec-ch-ua-mobile` = "?0",
#       `sec-ch-ua-platform` = "\"Windows\"",
#       `sec-fetch-dest` = "document",
#       `sec-fetch-mode` = "navigate",
#       `sec-fetch-site` = "none",
#       `sec-fetch-user` = "?1",
#       `upgrade-insecure-requests` = "1"
#     )
#   )
```

```
#
#   content <- content(response, "text")
#
#   html_parsed <- read_html(response)
#
#   tables = html_parsed %>%
#     rvest::html_nodes('table') %>%
#     rvest::html_table()
#
#
#   if (length(tables) == 2){
#     esg_for_peers = tables[[1]]
#     esg_prod_envol = tables[[2]]
#     # names(esg_prod_envol) = c('Info', 'Detail')
#     names(esg_prod_envol) = c("Products and Activities", "Significant Involvement")
#   }
#
#   # Obtenir
#   yf_tot_esg_score = html_parsed %>%rvest::html_nodes('.border.svelte-y3c2sq') %>%
#     rvest::html_text()
#
#   # Type of Risk
#   the_act_type = html_parsed %>%rvest::html_nodes('.perf.svelte-y3c2sq') %>%
#     rvest::html_text()
#
#   # Actual performance
#   actual_perf =  html_parsed %>%rvest::html_nodes('.scoreRank.svelte-y3c2sq') %>%
#     rvest::html_text()
#
```

```
#    E_grade = as.numeric(gsub(" ", "", actual_perf[2]))
#
#    S_grade = as.numeric(gsub(" ", "", actual_perf[3]))
#
#    G_grade = as.numeric(gsub(" ", "", actual_perf[4]))
#
#      controversy_score = html_parsed %>%rvest::html_nodes('.controversy-score.svelte-ye6fz0')
%>%
#     rvest::html_text()
#
#    peer_avg_score = html_parsed %>%rvest::html_nodes('.peer-score.svelte-ye6fz0') %>%
#     rvest::html_text()
#
#
#    partial_sever = rvest::read_html(response)%>%
#     rvest::html_nodes('.partial.svelte-he3t4n') %>%
#     rvest::html_text()
#
#    # Ici on a trois éléments
#    # "None"   "4"     "Severe"
#    list_range_sever = rvest::read_html(response)%>%
#     rvest::html_nodes('.scale.svelte-he3t4n span') %>%
#     rvest::html_text()
#
#    controversy_level_text = (html_parsed %>%
#                         rvest::html_nodes('.val.svelte-ye6fz0 span') %>%
#                         rvest::html_text())[3]
#
#    # Ou
```

```
#   # controversy_level <- rvest::read_html(response)%>%
#   #   rvest::html_nodes('.val.svelte-ye6fz0') %>%
#   #   rvest::html_text()
#
#   # controversy_level_text <- str_replace_all(controversy_level, "\\d+", "") %>%
#   #   str_trim()
#
#   updating_info = (html_parsed %>%rvest::html_nodes('.content.svelte-ye6fz0 span') %>%
#                      rvest::html_text())[1]
#
#   date_pattern = "\\d+/\\d+"
#
#   Last_update = str_extract(updating_info, date_pattern)
#
#   Last_update = as.Date(paste0(Last_update, "/01"), format="%m/%Y/%d")
#   Last_update = format(Last_update, "%m/%Y")
#
#   df = as.data.frame(matrix(NA, ncol = 2, nrow = 10))
#   names(df) = c('Info', 'Detail')
#
#   df$Info = c("Ticker",
#               "Total ESG Risk Score",
#               "Environmental Risk Score",
#               "Social Risk Score",
#               "Governance Risk Score",
#               "Actual performance",
#               "Peer average score",
#               "Controversy score",
#               "Controversy level",
```

```
#             "Last update")
#
#   df$Detail = c(ticker,
#             yf_tot_esg_score,
#             E_grade,
#             S_grade,
#             G_grade,
#             the_act_type,
#             peer_avg_score,
#             controversy_score,
#             controversy_level_text,
#             Last_update)
#
#   if(full){
#
#    if(!is.null(esg_prod_envol)){
#      names(esg_prod_envol) = c('Info', 'Detail')
#
#      final_df = rbind(df, esg_prod_envol)
#
#      return(final_df)
#
#    }else{
#      return(df)
#    }
#
#   } else{
#
#    return(df)
```

```
#   }
#
#  }
#
#  for (i in 1:length(ticker)) {
#   if(i == 1){
#
#    final_data = get_esg_for_ticker(ticker[i], full= full)
#
#   }else{
#
#    final_dt = get_esg_for_ticker(ticker[i], full= full)
#
#    final_data = cbind(final_data, final_dt[, 'Detail'])
#
#   }
#
#
#  }
#
#  # Rename final data
#  names(final_data)[2:length(final_data)] = final_data[1, 2:length(final_data)]
#  # remoove first row
#  final_data = final_data[-1,]
#
#  final_data[8, ] = gsub("Controversy Level", "", final_data[8, ])
#  # set first colomn to rownname
#  # rownames(final_data) = final_data[,1]
#  #
```

```
#   # final_data = final_data[,-1]
#   final_data = final_data %>%
#     as_tibble()%>%
#     column_to_rownames(var = "Info")
#
#   return(final_data)
#
# }

# get_esg_data(c("KO", "AAPL", "PRU"))

# get_esg_data(c("KO", "AAPL", "PRU"), full = FALSE)



# Version améliorée yf_get_keys.stats
# yf_get_keys.stats <- function(ticker, full = TRUE) {
#
#   if(all(ticker == "")){
#     # if(length(ticker) == 1  && ticker == ''){
#     return(NULL)
#   }
#
#  httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))
#
#   # Fonction pour obtenir les données ESG pour un seul ticker
#   get_keys_for_ticker <- function(ticker, full = TRUE) {
#                 httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))
```

```
#
#   url = glue::glue("https://finance.yahoo.com/quote/{ticker}/key-statistics/")
#   # url = paste0("https://finance.yahoo.com/quote/", ticker, "/key-statistics/")
#   # url <- "https://finance.yahoo.com/quote/KO/key-statistics/"
#
#   # Faire la requête GET avec les en-têtes spécifiés
#   response <- httr::GET(
#     url,
#     add_headers(
#                                                                                 `accept` =
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.7",
#       `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",
#       `cache-control` = "no-cache",
#       `pragma` = "no-cache",
#       `priority` = "u=0, i",
#                         `sec-ch-ua` = "\"Google Chrome\";v=\"125\", \"Chromium\";v=\"125\",
\"Not.A/Brand\";v=\"24\"",
#       `sec-ch-ua-mobile` = "?0",
#       `sec-ch-ua-platform` = "\"Windows\"",
#       `sec-fetch-dest` = "document",
#       `sec-fetch-mode` = "navigate",
#       `sec-fetch-site` = "none",
#       `sec-fetch-user` = "?1",
#       `upgrade-insecure-requests` = "1"
#     )
#   )
#
#   content <- content(response, "text")
#
```

```
#    html_parsed <- read_html(response)
#
#    tables = html_parsed %>%
#      rvest::html_nodes('table') %>%
#      rvest::html_table()
#
#
#    if (length(tables) == 10){
#      # After the rbind
#      # names(finacial_h_table) = c('Info', 'Detail')
#
#      ## Valuation Measures
#      ### 9 rows
#      val_measure = tables[[1]]
#      val_measure = val_measure[, c(1,2)]
#      # names(val_measure) = c("Info", "Current")
#      names(val_measure) = c("X1", "X2")
#
#      # Financial Highlights tab
#      ## Fiscal Year
#      ### 2 rows
#      Fiscal_Year = tables[[2]]
#
#      ## Profitability
#      ### 2 rows
#      Profitability = tables[[3]]
#
#      ## Management Effectiveness
#      ### 2 rows
```

```
#     Management_Effectiveness = tables[[4]]
#
#     ## Income Statement
#     ### 8 rows
#     INC.stat = tables[[5]]
#
#     ## Balance Sheet
#     ### 6 rows
#     BS = tables[[6]]
#
#     ## Cash Flow Statement
#     ### 2 rows
#     CF = tables[[7]]
#
#     ## Trading Information tab
#     ## Stock Price History
#     # Enlever les derniers chiffres en fin de texte
#     ### 7 rows
#     St_price_h = tables[[8]]
#
#     # Share Statistics
#     ### 12 rows
#     Share_stats = tables[[9]]
#
#     ## Dividends & Splits
#     ### 10 rows
#     Div_split = tables[[10]]
#
#     df_init = as.data.frame(matrix(NA, ncol = 2, nrow = 1))
```

```
#     names(df_init) = c('X1', 'X2')
#     df_init$X1 = c("Ticker")
#     df_init$X2 = ticker
#
#     df = rbind(df_init,
#             val_measure, Fiscal_Year, Profitability, Management_Effectiveness,
#             INC.stat, BS, CF, St_price_h, Share_stats, Div_split)
#
#     names(df) = c('Info', 'Detail')
#
#     df_val_measure = rbind(df_init,
#                     val_measure)
#     names(df_val_measure) = c('Info', 'Detail')
#
#     df_f.Highlights = rbind(df_init, Fiscal_Year, Profitability, Management_Effectiveness,
#                      INC.stat, BS, CF)
#     names(df_f.Highlights) = c('Info', 'Detail')
#
#     df_t.Information = rbind(df_init, St_price_h, Share_stats, Div_split)
#     names(df_t.Information) = c('Info', 'Detail')
#
#   }else{
#
#     df = structure(list(Info = c("Ticker", "Market Cap", "Enterprise Value",
#                         "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
#                         "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA",
#                         "Fiscal Year Ends", "Most Recent Quarter  (mrq)", "Profit Margin",
#                         "Operating Margin  (ttm)", "Return on Assets  (ttm)", "Return on Equity  (ttm)",
#                         "Revenue  (ttm)", "Revenue Per Share  (ttm)", "Quarterly Revenue Growth  (yoy)",
```

```
#                       "Gross Profit  (ttm)", "EBITDA", "Net Income Avi to Common  (ttm)",
#                       "Diluted EPS  (ttm)", "Quarterly Earnings Growth  (yoy)", "Total Cash  (mrq)",
#                       "Total Cash Per Share  (mrq)", "Total Debt  (mrq)", "Total Debt/Equity  (mrq)",
#                      "Current Ratio  (mrq)", "Book Value Per Share  (mrq)", "Operating Cash Flow  (ttm)",
#                       "Levered Free Cash Flow  (ttm)", "Beta (5Y Monthly)", "52 Week Range 3",
#                       "S&P 500 52-Week Change 3", "52 Week High 3", "52 Week Low 3",
#                       "50-Day Moving Average 3", "200-Day Moving Average 3", "Avg Vol (3 month) 3",
#                       "Avg Vol (10 day) 3", "Shares Outstanding 5", "Implied Shares Outstanding 6",
#                       "Float 8", "% Held by Insiders 1", "% Held by Institutions 1",
#                            "Shares Short (6/14/2024) 4", "Short Ratio (6/14/2024) 4", "Short % of Float
(6/14/2024) 4",
#                            "Short % of Shares Outstanding (6/14/2024) 4", "Shares Short (prior month
5/15/2024) 4",
#                       "Forward Annual Dividend Rate 4", "Forward Annual Dividend Yield 4",
#                       "Trailing Annual Dividend Rate 3", "Trailing Annual Dividend Yield 3",
#                       "5 Year Average Dividend Yield 4", "Payout Ratio 4", "Dividend Date 3",
#                       "Ex-Dividend Date 4", "Last Split Factor 2", "Last Split Date 3"
#     ), Detail = c(ticker, rep('-',60))),
#     row.names = c(NA, -61L), class = c("tbl_df",
#                              "tbl", "data.frame"))
#
#     df_val_measure = structure(list(Info = c("Ticker", "Market Cap", "Enterprise Value",
#                                "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
#                                "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA"),
#                           Detail = c(ticker, rep('-',9))),
#                       row.names = c(NA, -10L),
#                       class =  c("tbl_df","tbl", "data.frame"))
#
#    df_f.Highlights = structure(list(Info = c("Ticker", "Fiscal Year Ends", "Most Recent Quarter  (mrq)",
```

```
#                                 "Profit Margin", "Operating Margin  (ttm)", "Return on Assets  (ttm)",
#                                 "Return on Equity  (ttm)", "Revenue  (ttm)", "Revenue Per Share  (ttm)",
#                                 "Quarterly Revenue Growth  (yoy)", "Gross Profit  (ttm)", "EBITDA",
#                                "Net Income Avi to Common  (ttm)", "Diluted EPS  (ttm)", "Quarterly Earnings
Growth  (yoy)",
#                                 "Total Cash  (mrq)", "Total Cash Per Share  (mrq)", "Total Debt  (mrq)",
#                                    "Total Debt/Equity  (mrq)", "Current Ratio  (mrq)", "Book Value Per Share
(mrq)",
#                                 "Operating Cash Flow  (ttm)", "Levered Free Cash Flow  (ttm)"),
#                            Detail = c(ticker, rep('-',22))),
#                         row.names = c(NA, -23L),
#                         class =  c("tbl_df","tbl", "data.frame"))
#
#     df_t.Information = structure(list(Info = c("Ticker", "Beta (5Y Monthly)", "52 Week Range 3",
#                                  "S&P 500 52-Week Change 3", "52 Week High 3", "52 Week Low 3",
#                                 "50-Day Moving Average 3", "200-Day Moving Average 3", "Avg Vol (3 month)
3",
#                                  "Avg Vol (10 day) 3", "Shares Outstanding 5", "Implied Shares Outstanding
6",
#                                 "Float 8", "% Held by Insiders 1", "% Held by Institutions 1",
#                                 "Shares Short (6/14/2024) 4", "Short Ratio (6/14/2024) 4", "Short % of Float
(6/14/2024) 4",
#                                 "Short % of Shares Outstanding (6/14/2024) 4", "Shares Short (prior month
5/15/2024) 4",
#                                 "Forward Annual Dividend Rate 4", "Forward Annual Dividend Yield 4",
#                                 "Trailing Annual Dividend Rate 3", "Trailing Annual Dividend Yield 3",
#                                 "5 Year Average Dividend Yield 4", "Payout Ratio 4", "Dividend Date 3",
#                                 "Ex-Dividend Date 4", "Last Split Factor 2", "Last Split Date 3"),
#                            Detail = c(ticker, rep('-',29))),
#                         row.names = c(NA, -30L),
#                         class =  c("tbl_df","tbl", "data.frame"))
```

```
#
#   }
#
#   if(full){
#
#    if(!is.null(tables)){
#
#     final_df = df
#
#     return(final_df)
#
#     # return(list(final_df = df,
#     #          val_measure = df_val_measure,
#     #          f.Highlights = df_f.Highlights,
#     #          t.Information = df_t.Information))
#
#    }else{
#     names(val_measure) = c('Info', 'Detail')
#
#     return(val_measure)
#    }
#
#   } else{
#
#
#    return(list(val_measure = df_val_measure,
#             f.Highlights = df_f.Highlights,
#             t.Information = df_t.Information))
#   }
```

```
#
#  }
#
#  for (i in 1:length(ticker)) {
#   if(isTRUE(full)){
#    if(i == 1){
#
#      final_data = get_keys_for_ticker(ticker[i], full= full)
#    }else{
#
#      final_dt = get_keys_for_ticker(ticker[i], full= full)
#
#      final_data = cbind(final_data, final_dt[, 'Detail'])
#
#    }
#   } else{ #Losque false
#
#    if(i == 1){
#
#      final_data = get_keys_for_ticker(ticker[i], full= full)
#      final_data1 = final_data$val_measure
#      final_data2 = final_data$f.Highlights
#      final_data3 = final_data$t.Information
#
#    }else{
#
#      final_dt = get_keys_for_ticker(ticker[i], full= full)
#      final_dt1 = final_dt$val_measure
#      final_dt2 = final_dt$f.Highlights
```

```
#       final_dt3 = final_dt$t.Information
#
#       final_data1 = cbind(final_data1, final_dt1[, 'Detail'])
#       final_data2 = cbind(final_data2, final_dt2[, 'Detail'])
#       final_data3 = cbind(final_data3, final_dt3[, 'Detail'])
#
#
#     }
#
#
#
#   }
#
#
# }
#
#  if(isTRUE(full)){
#    # Rename final data
#    names(final_data)[2:length(final_data)] = final_data[1, 2:length(final_data)]
#    # remoove first row
#    final_data = final_data[-1,]
#
#    final_data = final_data %>%
#      as_tibble()%>%
#      column_to_rownames(var = "Info")
#
#    return(final_data)
#
#  } else{
```

```
#   # Rename final data1
#   names(final_data1)[2:length(final_data1)] = final_data1[1, 2:length(final_data1)]
#   # remoove first row
#   final_data1 = final_data1[-1,]
#
#   final_data1 = final_data1 %>%
#    as_tibble()%>%
#    column_to_rownames(var = "Info")
#
#   # Rename final data2
#   names(final_data2)[2:length(final_data2)] = final_data2[1, 2:length(final_data2)]
#   # remoove first row
#   final_data2 = final_data2[-1,]
#
#   final_data2 = final_data2 %>%
#    as_tibble()%>%
#    column_to_rownames(var = "Info")
#
#   # Rename final data3
#   names(final_data3)[2:length(final_data3)] = final_data3[1, 2:length(final_data3)]
#   # remoove first row
#   final_data3 = final_data3[-1,]
#
#   final_data3 = final_data3 %>%
#    as_tibble()%>%
#    column_to_rownames(var = "Info")
#
#   return(list(val_measure = final_data1,
#               f.Highlights = final_data2,
```

```
#             t.Information = final_data3))
#
#  }
#
# }
```

```
###############
# METRICS
###############

# Code super amélioré
# Define the function
calculate_portfolio_metrics <- function(tickers,
                                        benchmark,
                                        level_alpha = 0.05, #seuil
                                        MAR = 0, # Minimum acceptable Return
                                        MA_ESG = 0, # Minimum acceptable ESG
                                        rf_return = 0,
                                        sc = 0.05,
                                        start_date = Sys.Date() - 365*2,
                                        end_date = Sys.Date(),
                                        type_return = 'daily',
                                        return_method = 'log',
                                        user_weights = NULL) {
 # Get stock prices
 price_data <- tq_get(tickers, from = start_date, to = end_date, get = 'stock.prices')

 MAR = as.numeric(MAR)
```

```
MA_ESG = as.numeric(MA_ESG)
# price_data <- try(tq_get(ticker,
#                  get = "stock.prices",
#                  from = as.Date(start_date),
#                  to = as.Date(end_date) + 1),
#              silent = TRUE)

# print("--Testons la dataframe des prices--")
# print(head(price_data))
# print(tail(price_data))

# print("-- Head Price data --")
# print(head(price_data))

if(is.null(benchmark) || all(nchar(benchmark) == 0)){
 return(NULL)
} else{

 print(benchmark)

 # benchmark <- benchmark$benchmark

 benchmark_ret <- benchmark$df

 # benchmark <- benchmark[['benchmark']]

 # benchmark_ret <- benchmark[['df']]

 print("test sur la df")
```

```
  # print(head(benchmark_ret))
  print('str benchmark_ret')
  # print(str(benchmark_ret))
  print("Nom du bench")
  # print(benchmark)

  # risk_free_rate <- c("FR1YT=RR", 'US1YT=X', 'DE1YT=RR', "FR5YT=RR", 'US5YT=X', 'DE5YT=RR',
"FR10YT=RR", 'DE10YT=RR', 'US10YT=X')

  # bench_list <- benchmark_list$Name

  # BNC_final_list <- c(risk_free_rate, bench_list)

  # if(benchmark %in% BNC_final_list){
  #   benchmark_ret <- benchmark_ret
  #
  # } else{
  #
  #   benchmark_data <- tq_get(benchmark, from = start_date, to = end_date, get = 'stock.prices')
  #
  #   benchmark_ret <- benchmark_data %>%
  #     tq_transmute(select = adjusted,
  #                  mutate_fun = periodReturn,
  #                  period = type_return,
  #                  col_rename = 'benchmark_ret',
  #                  type = return_method) %>%
  #     tk_xts(date_var = "date", silent = TRUE)
  #
  #
```

```
  # }

  neg_price_val = which(price_data$adjusted<0)

  if(!is.null(neg_price_val)){
   price_data$adjusted[neg_price_val] = (price_data$adjusted)[which(price_data$adjusted<0)] *(-
1)
  }

  # Calculate returns
  log_ret_tidy <- price_data %>%
   group_by(symbol) %>%
   tq_transmute(select = adjusted,
            mutate_fun = periodReturn,
            period = type_return,
            col_rename = 'ret',
            type = return_method)

  # print("-- Head log_ret_tidy --")
  # print(head(log_ret_tidy))

  # Get ESG Data
  df_ = log_ret_tidy
  df_with_ESG <- list()

  for (ticker in unique(df_$symbol)) {
   dt_ <- df_ %>%
    dplyr::filter(symbol == ticker)
   # print(head(dt_))
```

```
dt_ESG = get_historic_esg(ticker)
if(!is.null(dt_ESG)){
 dt_ESG = na.omit(dt_ESG)
 names(dt_ESG) = c("date", "ESG", "E_Score", "S_Score", "G_Score")
 dt_ESG = dt_ESG[, c("date", "ESG")]

 start_date <- as.Date(start_date)
 end_date <- as.Date(end_date)

 # Générez une séquence de dates entre start_date et end_date (inclusivement) pour les jours ouvrables (business days)
 # date_range <- seq(from = start_date, to = end_date, by = "days")
 # workday_dates <- date_range[weekdays(date_range) %in% c("lundi", "mardi", "mercredi", "jeudi", "vendredi")]

 date_range <- base::seq(from = start_date, to = end_date, by = "days")
 jours_semaine <- c("lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche")

 if (any(weekdays(date_range, abbreviate = FALSE) %in% jours_semaine)){
  workday_dates <- date_range[weekdays(date_range) %in% c("lundi", "mardi", "mercredi", "jeudi", "vendredi")]

 } else{
  jours_semaine_en <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
  workday_dates <- date_range[weekdays(date_range, abbreviate = FALSE) %in% jours_semaine_en]
 }
```

```
  # Créez une dataframe avec les dates
  df <- data.frame(date = workday_dates)

  # Fusionnez les dataframes en utilisant la colonne "Date"
  merged_df <- merge(df, dt_ESG, by = "date", all.x = TRUE)

  df = merged_df%>%
   tidyr::fill(ESG)

  na_elm = which(is.na(df[,"ESG"]))
  non_nul_elm = which(!is.na(df[,"ESG"]))

  # Si les premiers elements sont encore des NA
  if(length(na_elm)!=0){
   if(length(non_nul_elm)!=0){
     df[na_elm,"ESG"] = 0

   }

  }

  # Fusionnez les dataframes en utilisant la colonne "Date"
  df_final <- merge(dt_, df, by = "date", all.x = TRUE)

  df_with_ESG[[ticker]] <- df_final%>%
   tidyr::fill(ESG)


}else{
```

```
    dt_$ESG <- 0
    df_with_ESG[[ticker]] <- dt_

   }

  }

  if(length(df_with_ESG) >= 1){
   # df_with_ESG <- do.call(rbind, df_with_ESG)
   df_with_ESG = dplyr::bind_rows(df_with_ESG)
  }

  log_ret_xts <- log_ret_tidy %>%
   spread(symbol, value = ret) %>%
   tk_xts(date_var = "date", silent = TRUE)

  # benchmark_ret <- benchmark_data %>%
  #   tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename
= 'benchmark_ret', type = return_method) %>%
  #   tk_xts()

  mean_ret <- colMeans(na.omit(log_ret_xts))

  annualization_factor <- switch(type_return,
                "daily" = 252,
                "weekly" = 52,
                "monthly" = 12,
                "yearly" = 1,
                stop("type_return doit être 'daily', 'weekly', 'monthly' ou 'yearly'"))
```

```
cov_mat <- cov(na.omit(log_ret_xts)) * annualization_factor

if(is.character(user_weights)){
  stop("User weights should be a numeric vector!")
}

if (!is.null(user_weights)) {

  if(sum(user_weights) < 0 ){
    stop("User weights should be positive!")
  }

  if(sum(user_weights) == 0 ){
    l_ticker = length(tickers)
    user_weights = rep((1/l_ticker), l_ticker)
  } else if(sum(user_weights) > 0 && sum(user_weights) <= 1 ){
    user_weights = user_weights
  } else if(sum(user_weights) > 1 && sum(user_weights) <= 100 ){
    user_weights = user_weights/100
  } else{
    user_weights <- user_weights / sum(user_weights)
  }

} else{

  user_weights <- rep(1 / length(tickers), length(tickers))

}
```

```
  # if (is.null(user_weights)) {
  #   user_weights <- rep(1 / length(tickers), length(tickers))
  # }

  portfolio_ret <- sum(user_weights * mean_ret)
  portfolio_ret <- ((portfolio_ret + 1)^annualization_factor) - 1
  portfolio_risk <- sqrt(t(user_weights) %*% (cov_mat %*% user_weights))

  weights <- matrix(user_weights, ncol = 1)

  portfolio_var <- as.numeric(t(weights) %*% var(na.omit(log_ret_xts)) %*% weights)

  pf_ret <- Return.portfolio(na.omit(log_ret_xts), weights = user_weights)

  log_ret_xts = na.omit(log_ret_xts)

  log_ret_xts$portfolio.returns <- pf_ret$portfolio.returns

  if (is.numeric(rf_return) && length(rf_return) == 1) {
    rf_return <- rf_return / annualization_factor
  } else if (is.character(rf_return) && length(rf_return) == 1) {
    rf_data <- tq_get(rf_return, from = start_date, to = end_date, get = 'stock.prices')
    rf_log_ret_tidy <- rf_data %>%
      tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename =
'rf_return', type = return_method)
    rf_return <- mean(rf_log_ret_tidy$rf_return)
  } else {
    stop("rf doit être un taux numérique ou un ticker valide.")
```

```
}

sharpe_ratio <- (portfolio_ret - rf_return) / portfolio_risk

benchmark_mean_ret <- mean(na.omit(benchmark_ret))

tracking_error <- sqrt(mean((rowMeans(log_ret_xts) - benchmark_mean_ret)^2)) * sqrt(annualization_factor)

# Align dates of log_ret_xts and benchmark_ret
common_dates <- index(na.omit(log_ret_xts)) %in% index(na.omit(benchmark_ret))
log_ret_xts_aligned <- log_ret_xts[common_dates]
benchmark_ret_aligned <- benchmark_ret[common_dates]

# print("-- Dim log_ret_xts --")
# print(dim(log_ret_xts))
# print(head(log_ret_xts,3))

# print("-Dim benchmark_ret")
# print(dim(benchmark_ret))
# print(head(benchmark_ret,3))

beta <- cov(as.numeric(rowMeans(log_ret_xts_aligned)), as.numeric(benchmark_ret_aligned)) / var(as.numeric(benchmark_ret_aligned))

beta <- beta * annualization_factor
# beta <- cov(as.numeric(rowMeans(log_ret_xts)), as.numeric(benchmark_ret)) / var(as.numeric(benchmark_ret))

print("-- log_ret_xts_aligned --")
```

```
# print(log_ret_xts_aligned)

pf_log_ret_xts_al = log_ret_xts_aligned[, "portfolio.returns"]

# Calculate beta bull and beta bear
beta_bull <- cov(as.numeric(rowMeans(pf_log_ret_xts_al[benchmark_ret_aligned > 0])),
            as.numeric(benchmark_ret_aligned[benchmark_ret_aligned > 0])) /
 var(as.numeric(benchmark_ret_aligned[benchmark_ret_aligned > 0]))

beta_bear <- cov(as.numeric(rowMeans(pf_log_ret_xts_al[benchmark_ret_aligned < 0])),
            as.numeric(benchmark_ret_aligned[benchmark_ret_aligned < 0])) /
 var(as.numeric(benchmark_ret_aligned[benchmark_ret_aligned < 0]))

# Annualization of beta_bull and bear
beta_bull <- beta_bull * annualization_factor
beta_bear <- beta_bear * annualization_factor

# Calculate beta bull and beta bear
# beta_bull <- cov(as.numeric(rowMeans(log_ret_xts_aligned[benchmark_ret_aligned > 0])),
#             as.numeric(benchmark_ret_aligned[benchmark_ret_aligned > 0])) /
#  var(as.numeric(benchmark_ret_aligned[benchmark_ret_aligned > 0]))

# beta_bear <- cov(as.numeric(rowMeans(log_ret_xts_aligned[benchmark_ret_aligned < 0])),
#             as.numeric(benchmark_ret_aligned[benchmark_ret_aligned < 0])) /
#  var(as.numeric(benchmark_ret_aligned[benchmark_ret_aligned < 0]))

alpha <- jenson.alpha(log_ret_xts$portfolio.returns, benchmark_ret, rf = rf_return)

# Alpha de Jensen modifié
```

```
  modified_jensen_alpha <- portfolio_ret - (rf_return + beta * (benchmark_mean_ret - rf_return))

  downside_deviation <- sqrt(mean(ifelse(rowMeans(log_ret_xts) < MAR, (rowMeans(log_ret_xts) - MAR)^2, 0))) * sqrt(annualization_factor)
  sortino_ratio <- (portfolio_ret - MAR) / downside_deviation

  # info_ratio <- ratio.information(log_ret_xts$portfolio.returns, benchmark_ret)
  # print("--Test head log_ret_xts_aligned$portfolio.returns --")
  # print(head(log_ret_xts_aligned$portfolio.returns))

  # print('-- Head benchmark_ret_aligned --')
  # print(head(benchmark_ret_aligned, 3))

  info_ratio <- ratio.information(log_ret_xts_aligned$portfolio.returns, benchmark_ret_aligned)

  modified_info_ratio <- (portfolio_ret - benchmark_mean_ret) / tracking_error

  # Calcul du skewness
  skewness <- skewness(log_ret_xts_aligned$portfolio.returns)

  # Calcul du kurtosis
  kurtosis <- kurtosis(log_ret_xts_aligned$portfolio.returns)

  jb_test_ = normalTest(log_ret_xts_aligned$portfolio.returns, method = c("jb"), na.rm = TRUE)

  jb_test_ = jb_test_@test$p.value[[1]]

  jb_normality <- ifelse(jb_test_ > as.numeric(level_alpha), "Normal", "Not Normal")
```

```
  # Calculate metrics by ticker
  metrics_by_ticker <- sapply(tickers, function(ticker) {
    ticker_returns <- log_ret_xts[, ticker]
    common_dates <- index(na.omit(ticker_returns)) %in% index(na.omit(benchmark_ret))
    ticker_returns_aligned <- ticker_returns[common_dates]
    benchmark_ret_aligned <- benchmark_ret[common_dates]



    ticker_return_annualized <- ((mean(ticker_returns) + 1)^annualization_factor) - 1
    ticker_risk_annualized <- sd(ticker_returns) * sqrt(annualization_factor)
    # ticker_beta <- cov(ticker_returns, benchmark_ret) / var(benchmark_ret)
    ticker_beta <- cov(ticker_returns_aligned, benchmark_ret_aligned) / var(benchmark_ret_aligned)
    ticker_beta <- ticker_beta * annualization_factor


    ticker_alpha <- ticker_return_annualized - (rf_return + ticker_beta * (benchmark_mean_ret - rf_return))
    ticker_sharpe_ratio <- (ticker_return_annualized - rf_return) / ticker_risk_annualized
    ticker_ESG <- (df_with_ESG[df_with_ESG$symbol == ticker,4])[length(df_with_ESG[df_with_ESG$symbol == ticker,4])]
    ticker_modified_SR <- ifelse(ticker_sharpe_ratio > 0,
                                 ticker_sharpe_ratio * (1 + ticker_ESG)^sc,
                                 ticker_sharpe_ratio * (1 + ticker_ESG)^(-sc)
    )
    ticker_sortino_ratio <- (ticker_return_annualized - MAR) / (sd(ticker_returns[ticker_returns < MAR]) * sqrt(annualization_factor))
    ticker_tracking_error <- sqrt(mean((ticker_returns - benchmark_ret)^2)) * sqrt(annualization_factor)
    ticker_info_ratio <- (ticker_return_annualized - benchmark_mean_ret) / ticker_tracking_error
    modified_jensen_alpha_ticker <- ticker_return_annualized - (rf_return + ticker_beta * (benchmark_mean_ret - rf_return))
```

```
modified_info_ratio_ticker <- (ticker_return_annualized - benchmark_mean_ret) / ticker_tracking_error



# Calcul du skewness
ticker_skewness <- skewness(ticker_returns)
ticker_skewness <- ticker_skewness[[1]]

# Calcul du kurtosis
ticker_kurtosis <- kurtosis(ticker_returns)
ticker_kurtosis <- ticker_kurtosis[[1]]

# Test de Jarque-Bera
ticker_jb_test <- jarque.bera.test(na.omit(ticker_returns))
ticker_jb_p_value <- ticker_jb_test$p.value
ticker_jb_normality <- ifelse(ticker_jb_p_value > level_alpha, "Normal", "Not Normal")



c(annual_return = ticker_return_annualized,
  annual_risk = ticker_risk_annualized,
  beta = ticker_beta,
  tracking_error = ticker_tracking_error,
  sharpe_ratio = ticker_sharpe_ratio,
  modified_SR = ticker_modified_SR,
  info_ratio = ticker_info_ratio,
  modified_info_ratio = modified_info_ratio_ticker,
  jensen_alpha = ticker_alpha,
  modified_jensen_alpha = modified_jensen_alpha_ticker,
  sortino_ratio = ticker_sortino_ratio, #11
```

```
    ESG = ticker_ESG,#12 avant amélioration
    Skewness = ticker_skewness,
    Kurtosis = ticker_kurtosis,
    Jb.P_value = ticker_jb_p_value
    # ,
    # Jb_test_decision = ticker_jb_normality #16
   )

  })

  print("--premier test --")
  # print(metrics_by_ticker)

  esg_line <- which(row.names(metrics_by_ticker) == 'ESG')

  print("--ESG line--")
  # print(esg_line)

  print("-Class user weight-")
  # print(class(user_weights))
  pf_ESG = sum(as.numeric(metrics_by_ticker[esg_line,]) * user_weights)

  print("--pf_ESG--")
  print(pf_ESG)

  # pf_ESG = sum(metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),] *
user_weights)

  # pf_ESG = c(pf_ESG)
```

```
  # Modified Sharpe ratio #is row 12
  Modified_SR <- ifelse(as.numeric(sharpe_ratio) > 0,
                        as.numeric(sharpe_ratio)* (1 + pf_ESG)^sc,
                        as.numeric(sharpe_ratio)* (1 + pf_ESG)^(-sc)
  )

  mean_esg <- sum(as.numeric(metrics_by_ticker[esg_line,])) / as.numeric(ncol(metrics_by_ticker))
  # mean_esg <- sum(metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) / ncol(metrics_by_ticker)

  # print("-- mean_esg --")
  # print(mean_esg)

  # metrics_by_ticker["Relative ESG", ] <- (metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) - mean_esg
  # Relative_ESG <- (metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) - mean_esg
  Relative_ESG <- as.numeric(metrics_by_ticker[esg_line, ]) - as.numeric(mean_esg)

  # Stresser le relative ESG
  # Chercher à les convertir tous en valeur positive
  Relative_ESG <- (Relative_ESG^(-3)) / 3

  metrics_by_ticker <- rbind(metrics_by_ticker, Relative_ESG)

  rownames(metrics_by_ticker)[nrow(metrics_by_ticker)] <- "Relative_ESG"

  # metrics_by_ticker[13,] is Relative_ESG
```

```
  the_num <- which(row.names(metrics_by_ticker) == 'Relative_ESG')
  print("-- Test sur la ligne Relative_ESG --")
  # print(the_num)

  metrics_by_ticker[the_num,] <- ifelse(as.numeric(metrics_by_ticker[5,]) > 0,
                        as.numeric(metrics_by_ticker[5,])  *  (1  +
as.numeric(metrics_by_ticker[the_num,]) )^sc,
                        as.numeric(metrics_by_ticker[5,])  *  (1  +
as.numeric(metrics_by_ticker[the_num,]) )^(-sc))

  # Avant l'amélioration du code
  # metrics_by_ticker[13,] <- ifelse(metrics_by_ticker[5,] > 0,
  #                       metrics_by_ticker[5,]  * (1 + metrics_by_ticker[13,] )^sc,
  #                       metrics_by_ticker[5,]  * (1 + metrics_by_ticker[13,] )^(-sc))



  SR_with_MAESG <- NA #
  metrics_by_ticker = rbind(metrics_by_ticker, SR_with_MAESG)

  tot_row = nrow(metrics_by_ticker)

  if(MA_ESG != 0){
   metrics_by_ticker[tot_row,] <- (as.numeric(metrics_by_ticker[esg_line,]) - MA_ESG^(-3))/3
   # metrics_by_ticker[tot_row,] <- (metrics_by_ticker["ESG",] - MA_ESG^(-3))/3
  }else{
   metrics_by_ticker[tot_row,] <- NA
  }
```

```
  # metrics_by_ticker["Relative_ESG", ] <- ifelse(metrics_by_ticker["sharpe_ratio", ] > 0,
  #                         metrics_by_ticker["sharpe_ratio", ] * (1 + metrics_by_ticker["Relative_ESG",
])^sc,
  #                         metrics_by_ticker["sharpe_ratio", ] * (1 + metrics_by_ticker["Relative_ESG",
])^(-sc)
  # )


  rownames(metrics_by_ticker) <- c("Annual return", "Annual risk", "Beta", "Tracking error",
                    "Sharpe ratio", "Modified Sharpe ratio", "Information ratio", "Modified information
ratio",
                    "Jensen alpha", "Modified Jensen alpha",
                    "Sortino ratio", "ESG",
                    "Skewness", "Kurtosis", "JB P Value",
                    # "JB test Decision",
                    "Relative Sharpe ratio", "Sharpe ratio with MAESG")


  metrics_by_ticker <- metrics_by_ticker[c("ESG","Annual return", "Annual risk", "Beta", "Tracking
error",
                       "Sharpe ratio", "Modified Sharpe ratio", "Relative Sharpe ratio", "Sharpe ratio
with MAESG",
                       "Information ratio", "Modified information ratio",
                       "Jensen alpha", "Modified Jensen alpha",
                       "Sortino ratio",
                       "Skewness", "Kurtosis", "JB P Value"
                       # , "JB test Decision"
  ),]
```

```
the_num1 <- which(row.names(metrics_by_ticker) == 'JB P Value')
# which(rownames(metrics_by_ticker) == "JB test Decision")

nbr = 1:17
# Ne pas arrondir JB Pvalue
remaining_rows <- nbr[nbr != the_num1]

metrics_by_ticker[remaining_rows,] <- round(as.numeric(metrics_by_ticker[remaining_rows,]), 6)

result <- list(
  log_ret_tidy = log_ret_tidy,
  df_with_ESG = df_with_ESG,
  return_table = log_ret_xts,
  pf_ESG = pf_ESG,
  pf_annual_return = portfolio_ret,
  pf_annual_risk = portfolio_risk[1],
  pf_var = portfolio_var,
  pf_tracking_error = tracking_error,
  pf_beta = beta,
  pf_beta_bull = beta_bull,
  pf_beta_bear = beta_bear,
  pf_sharpe_ratio = sharpe_ratio[1],
  pf_Modified_SR = Modified_SR,
  pf_info_ratio = info_ratio,
  pf_modified_info_ratio = modified_info_ratio,
  pf_jensen_alpha = alpha,
  pf_modified_jensen_alpha = modified_jensen_alpha,
  pf_sortino_ratio = sortino_ratio,
```

```
    pf_skewness = skewness[[1]],
    pf_kurtosis = kurtosis[[1]],
    pf_jb.pvalue = jb_test_,
    pf_jb_test.decision = jb_normality,
    # metrics_by_ticker = as.data.frame(t(metrics_by_ticker))
    metrics_by_ticker = as.data.frame(metrics_by_ticker)
   )

  return(result)

 }



}

# # New version à améliorer
# calculate_portfolio_metrics <- function(tickers,
#                         benchmark,
#                         level_alpha = 0.05, #seuil
#                         MAR = 0, # Minimum acceptable Return
#                         MA_ESG = 0, # Minimum acceptable ESG
#                         rf_return = 0,
#                         sc = 0.05,
#                         start_date = Sys.Date() - 365*2,
#                         end_date = Sys.Date(),
#                         type_return = 'daily',
#                         return_method = 'log',
#                         user_weights = NULL) {
#   # Get stock prices
```

```
#   price_data <- tq_get(tickers, from = start_date, to = end_date, get = 'stock.prices')
#
#   MAR = as.numeric(MAR)
#   MA_ESG = as.numeric(MA_ESG)
#   # price_data <- try(tq_get(ticker,
#   #                  get = "stock.prices",
#   #                  from = as.Date(start_date),
#   #                  to = as.Date(end_date) + 1),
#   #              silent = TRUE)
#
#   # print("--Testons la dataframe des prices--")
#   # print(head(price_data))
#   # print(tail(price_data))
#
#   # print("-- Head Price data --")
#   # print(head(price_data))
#
#   if(is.null(benchmark) || all(nchar(benchmark) == 0)){
#   # if(is.null(benchmark) || nchar(benchmark) == 0){
#     return(NULL)
#   } else{
#       risk_free_rate <- c("FR1YT=RR", 'US1YT=X', 'DE1YT=RR', "FR5YT=RR", 'US5YT=X', 'DE5YT=RR',
"FR10YT=RR", 'DE10YT=RR', 'US10YT=X')
#
#     bench_list <- benchmark_list$Name
#
#     if(benchmark %in% bench_list){
#
#       benchmark_data  <- Get_bench_data(Name = benchmark,
```

```
#                           # from = as.Date(start_date),
#                           from = start_date,
#                           na_method = "keep",
#                           # na_method = "previous",
#                           to = end_date
#                           # to = as.Date(end_date)
#                           )
#
#     # Récupération du benchmark (BNC_return ou RF_Data_return)
#     benchmark_data <- if (!is.null(benchmark_data) && nrow(benchmark_data) > 1) {
#
#       colnames(benchmark_data)[2] <- 'adjusted' # Renommer par close la seconde colonne
#
#       benchmark_ret <- benchmark_data %>%
#         tq_transmute(select = adjusted,
#                      mutate_fun = periodReturn,
#                      period = type_return,
#                      col_rename = 'benchmark_ret',
#                      type = return_method) %>%
#         tk_xts(date_var = "date", silent = TRUE)
#
#     }
#
#   } else if(benchmark %in% risk_free_rate){
#
#     # benchmark_data <-  ready_hcDt_new(ticker_info = benchmark,
#     #                     "D", start_date, end_date)
#     #
#     # benchmark_data$Date <- as.Date(benchmark_data$Date)
```

```
#
#
#    # benchmark_data <-  fetch_inv_prices(ticker_info = benchmark,
#    #                        time_frame = 'Daily',
#    #                        from = start_date,
#    #                        to = end_date)
#
#    RF_id_new = switch(benchmark,
#                "FR1YT=RR" = 23769,
#                'DE1YT=RR' = 23684,
#                'US1YT=X' = 23700,
#                "FR5YT=RR" = 23773,
#                'DE5YT=RR' = 23688,
#                'US5YT=X' = 23703,
#                "FR10YT=RR" = 23778,
#                'DE10YT=RR' = 23693,
#                'US10YT=X' = 23705 )
#
#    benchmark_data <- ready_hcDt(ticker_id = RF_id_new, max_retries = 30)
#    benchmark_data$Date <- as.Date(benchmark_data$Date)
#
#    benchmark_data <- dplyr::as_tibble(benchmark_data) %>%
#     dplyr::filter(Date >= as.Date(start_date)) %>%
#     dplyr::filter(Date <= as.Date(end_date))
#
#    benchmark_ret <- benchmark_data %>%
#     tq_transmute(select = Close,
#                mutate_fun = periodReturn,
#                period = type_return,
```

```
#                 col_rename = 'benchmark_ret',
#                 type = return_method) %>%
#     tk_xts(date_var = "Date", silent = TRUE)
#
#   } else{
#
#
#     benchmark_data <- tq_get(benchmark, from = start_date, to = end_date, get = 'stock.prices')
#
#     benchmark_ret <- benchmark_data %>%
#       tq_transmute(select = adjusted,
#                 mutate_fun = periodReturn,
#                 period = type_return,
#                 col_rename = 'benchmark_ret',
#                 type = return_method) %>%
#       tk_xts(date_var = "date", silent = TRUE)
#
#
#   }
#
#   neg_price_val = which(price_data$adjusted<0)
#
#   if(!is.null(neg_price_val)){
#    price_data$adjusted[neg_price_val] = (price_data$adjusted)[which(price_data$adjusted<0)] *(-
1)
#   }
#
#   # Calculate returns
#   log_ret_tidy <- price_data %>%
```

```
#     group_by(symbol) %>%
#     tq_transmute(select = adjusted,
#                  mutate_fun = periodReturn,
#                  period = type_return,
#                  col_rename = 'ret',
#                  type = return_method)
#
#   # print("-- Head log_ret_tidy --")
#   # print(head(log_ret_tidy))
#
#   # Get ESG Data
#   df_ = log_ret_tidy
#   df_with_ESG <- list()
#
#   for (ticker in unique(df_$symbol)) {
#     dt_ <- df_ %>%
#       dplyr::filter(symbol == ticker)
#     # print(head(dt_))
#
#     dt_ESG = get_historic_esg(ticker)
#     if(!is.null(dt_ESG)){
#       dt_ESG = na.omit(dt_ESG)
#       names(dt_ESG) = c("date", "ESG", "E_Score", "S_Score", "G_Score")
#       dt_ESG = dt_ESG[, c("date", "ESG")]
#
#       start_date <- as.Date(start_date)
#       end_date <- as.Date(end_date)
#
```

```
#       # Générez une séquence de dates entre start_date et end_date (inclusivement) pour les jours ouvrables (business days)

#       # date_range <- seq(from = start_date, to = end_date, by = "days")

#         # workday_dates <- date_range[weekdays(date_range) %in% c("lundi", "mardi", "mercredi", "jeudi", "vendredi")]

#

#       date_range <- base::seq(from = start_date, to = end_date, by = "days")

#       jours_semaine <- c("lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche")

#

#       if (any(weekdays(date_range, abbreviate = FALSE) %in% jours_semaine)){

#           workday_dates <- date_range[weekdays(date_range) %in% c("lundi", "mardi", "mercredi", "jeudi", "vendredi")]

#

#       } else{

#        jours_semaine_en <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")

#                   workday_dates <- date_range[weekdays(date_range, abbreviate = FALSE) %in% jours_semaine_en]

#       }

#

#

#       # Créez une dataframe avec les dates

#       df <- data.frame(date = workday_dates)

#

#       # Fusionnez les dataframes en utilisant la colonne "Date"

#       merged_df <- merge(df, dt_ESG, by = "date", all.x = TRUE)

#

#       df = merged_df%>%

#        tidyr::fill(ESG)

#

#       na_elm = which(is.na(df[,"ESG"]))
```

```
#       non_nul_elm = which(!is.na(df[,"ESG"]))
#
#       # Si les premiers elements sont encore des NA
#       if(length(na_elm)!=0){
#         if(length(non_nul_elm)!=0){
#           df[na_elm,"ESG"] = 0
#
#         }
#
#       }
#
#       # Fusionnez les dataframes en utilisant la colonne "Date"
#       df_final <- merge(dt_, df, by = "date", all.x = TRUE)
#
#       df_with_ESG[[ticker]] <- df_final%>%
#         tidyr::fill(ESG)
#
#
#     }else{
#       dt_$ESG <- 0
#       df_with_ESG[[ticker]] <- dt_
#
#     }
#
#   }
#
#   if(length(df_with_ESG) >= 1){
#     # df_with_ESG <- do.call(rbind, df_with_ESG)
#     df_with_ESG = dplyr::bind_rows(df_with_ESG)
```

```
#   }
#
#   log_ret_xts <- log_ret_tidy %>%
#     spread(symbol, value = ret) %>%
#     tk_xts(date_var = "date", silent = TRUE)
#
#   # benchmark_ret <- benchmark_data %>%
#   #  tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename
= 'benchmark_ret', type = return_method) %>%
#   #  tk_xts()
#
#   mean_ret <- colMeans(na.omit(log_ret_xts))
#
#   annualization_factor <- switch(type_return,
#                          "daily" = 252,
#                          "weekly" = 52,
#                          "monthly" = 12,
#                          "yearly" = 1,
#                          stop("type_return doit être 'daily', 'weekly', 'monthly' ou 'yearly'"))
#
#   cov_mat <- cov(na.omit(log_ret_xts)) * annualization_factor
#
#   if(is.character(user_weights)){
#     stop("User weights should be a numeric vector!")
#   }
#
#   if (!is.null(user_weights)) {
#
#     if(sum(user_weights) < 0 ){
```

```
#       stop("User weights should be positive!")
#     }
#
#     if(sum(user_weights) == 0 ){
#       l_ticker = length(tickers)
#       user_weights = rep((1/l_ticker), l_ticker)
#     } else if(sum(user_weights) > 0 && sum(user_weights) <= 1 ){
#       user_weights = user_weights
#     } else if(sum(user_weights) > 1 && sum(user_weights) <= 100 ){
#       user_weights = user_weights/100
#     } else{
#       user_weights <- user_weights / sum(user_weights)
#     }
#
#   } else{
#
#     user_weights <- rep(1 / length(tickers), length(tickers))
#
#   }
#
#   # if (is.null(user_weights)) {
#   #   user_weights <- rep(1 / length(tickers), length(tickers))
#   # }
#
#   portfolio_ret <- sum(user_weights * mean_ret)
#   portfolio_ret <- ((portfolio_ret + 1)^annualization_factor) - 1
#   portfolio_risk <- sqrt(t(user_weights) %*% (cov_mat %*% user_weights))
#
#   weights <- matrix(user_weights, ncol = 1)
```

```
#
#    portfolio_var <- as.numeric(t(weights) %*% var(na.omit(log_ret_xts)) %*% weights)
#
#    pf_ret <- Return.portfolio(na.omit(log_ret_xts), weights = user_weights)
#
#    log_ret_xts = na.omit(log_ret_xts)
#
#    log_ret_xts$portfolio.returns <- pf_ret$portfolio.returns
#
#    if (is.numeric(rf_return) && length(rf_return) == 1) {
#      rf_return <- rf_return / annualization_factor
#    } else if (is.character(rf_return) && length(rf_return) == 1) {
#      rf_data <- tq_get(rf_return, from = start_date, to = end_date, get = 'stock.prices')
#      rf_log_ret_tidy <- rf_data %>%
#       tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename
= 'rf_return', type = return_method)
#      rf_return <- mean(rf_log_ret_tidy$rf_return)
#    } else {
#      stop("rf doit être un taux numérique ou un ticker valide.")
#    }
#
#    sharpe_ratio <- (portfolio_ret - rf_return) / portfolio_risk
#
#    benchmark_mean_ret <- mean(na.omit(benchmark_ret))
#
#             tracking_error <- sqrt(mean((rowMeans(log_ret_xts) - benchmark_mean_ret)^2)) *
sqrt(annualization_factor)
#
#    # Align dates of log_ret_xts and benchmark_ret
```

```
#    common_dates <- index(na.omit(log_ret_xts)) %in% index(na.omit(benchmark_ret))
#    log_ret_xts_aligned <- log_ret_xts[common_dates]
#    benchmark_ret_aligned <- benchmark_ret[common_dates]
#
#    # print("-- Dim log_ret_xts --")
#    # print(dim(log_ret_xts))
#    # print(head(log_ret_xts,3))
#
#    # print("-Dim benchmark_ret")
#    # print(dim(benchmark_ret))
#    # print(head(benchmark_ret,3))
#
#     beta <- cov(as.numeric(rowMeans(log_ret_xts_aligned)), as.numeric(benchmark_ret_aligned)) /
var(as.numeric(benchmark_ret_aligned))
#
#    beta <- beta * annualization_factor
#              # beta <- cov(as.numeric(rowMeans(log_ret_xts)), as.numeric(benchmark_ret)) /
var(as.numeric(benchmark_ret))
#
#    print("-- log_ret_xts_aligned --")
#    print(log_ret_xts_aligned)
#
#    pf_log_ret_xts_al = log_ret_xts_aligned[, "portfolio.returns"]
#
#    # Calculate beta bull and beta bear
#    beta_bull <- cov(as.numeric(rowMeans(pf_log_ret_xts_al[benchmark_ret_aligned > 0])),
#                as.numeric(benchmark_ret_aligned[benchmark_ret_aligned > 0])) /
#     var(as.numeric(benchmark_ret_aligned[benchmark_ret_aligned > 0]))
#
```

```
#    beta_bear <- cov(as.numeric(rowMeans(pf_log_ret_xts_al[benchmark_ret_aligned < 0])),
#                as.numeric(benchmark_ret_aligned[benchmark_ret_aligned < 0])) /
#     var(as.numeric(benchmark_ret_aligned[benchmark_ret_aligned < 0]))
#
#    # Annualization of beta_bull and bear
#    beta_bull <- beta_bull * annualization_factor
#    beta_bear <- beta_bear * annualization_factor
#
#    # Calculate beta bull and beta bear
#    # beta_bull <- cov(as.numeric(rowMeans(log_ret_xts_aligned[benchmark_ret_aligned > 0])),
#    #              as.numeric(benchmark_ret_aligned[benchmark_ret_aligned > 0])) /
#    #   var(as.numeric(benchmark_ret_aligned[benchmark_ret_aligned > 0]))
#
#    # beta_bear <- cov(as.numeric(rowMeans(log_ret_xts_aligned[benchmark_ret_aligned < 0])),
#    #              as.numeric(benchmark_ret_aligned[benchmark_ret_aligned < 0])) /
#    #   var(as.numeric(benchmark_ret_aligned[benchmark_ret_aligned < 0]))
#
#    alpha <- jenson.alpha(log_ret_xts$portfolio.returns, benchmark_ret, rf = rf_return)
#
#    # Alpha de Jensen modifié
#    modified_jensen_alpha <- portfolio_ret - (rf_return + beta * (benchmark_mean_ret - rf_return))
#
#    downside_deviation <- sqrt(mean(ifelse(rowMeans(log_ret_xts) < MAR, (rowMeans(log_ret_xts) -
MAR)^2, 0))) * sqrt(annualization_factor)
#    sortino_ratio <- (portfolio_ret - MAR) / downside_deviation
#
#    # info_ratio <- ratio.information(log_ret_xts$portfolio.returns, benchmark_ret)
#    # print("--Test head log_ret_xts_aligned$portfolio.returns --")
#    # print(head(log_ret_xts_aligned$portfolio.returns))
```

```
#
#   # print('-- Head benchmark_ret_aligned --')
#   # print(head(benchmark_ret_aligned, 3))
#
#   info_ratio <- ratio.information(log_ret_xts_aligned$portfolio.returns, benchmark_ret_aligned)
#
#   modified_info_ratio <- (portfolio_ret - benchmark_mean_ret) / tracking_error
#
#   # Calcul du skewness
#   skewness <- skewness(log_ret_xts_aligned$portfolio.returns)
#
#   # Calcul du kurtosis
#   kurtosis <- kurtosis(log_ret_xts_aligned$portfolio.returns)
#
#   jb_test_ = normalTest(log_ret_xts_aligned$portfolio.returns, method = c("jb"), na.rm = TRUE)
#
#   jb_test_ = jb_test_@test$p.value[[1]]
#
#   jb_normality <- ifelse(jb_test_ > as.numeric(level_alpha), "Normal", "Not Normal")
#
#   # Calculate metrics by ticker
#   metrics_by_ticker <- sapply(tickers, function(ticker) {
#     ticker_returns <- log_ret_xts[, ticker]
#     common_dates <- index(na.omit(ticker_returns)) %in% index(na.omit(benchmark_ret))
#     ticker_returns_aligned <- ticker_returns[common_dates]
#     benchmark_ret_aligned <- benchmark_ret[common_dates]
#
#
#     ticker_return_annualized <- ((mean(ticker_returns) + 1)^annualization_factor) - 1
```

```
#     ticker_risk_annualized <- sd(ticker_returns) * sqrt(annualization_factor)

#     # ticker_beta <- cov(ticker_returns, benchmark_ret) / var(benchmark_ret)

#                         ticker_beta <- cov(ticker_returns_aligned, benchmark_ret_aligned) /
var(benchmark_ret_aligned)

#     ticker_beta <- ticker_beta * annualization_factor

#

#       ticker_alpha <- ticker_return_annualized - (rf_return + ticker_beta * (benchmark_mean_ret -
rf_return))

#     ticker_sharpe_ratio <- (ticker_return_annualized - rf_return) / ticker_risk_annualized

#                                         ticker_ESG <-       (df_with_ESG[df_with_ESG$symbol ==
ticker,4])[length(df_with_ESG[df_with_ESG$symbol == ticker,4])]

#     ticker_modified_SR <- ifelse(ticker_sharpe_ratio > 0,

#                         ticker_sharpe_ratio * (1 + ticker_ESG)^sc,

#                         ticker_sharpe_ratio * (1 + ticker_ESG)^(-sc)

#     )

#       ticker_sortino_ratio <- (ticker_return_annualized - MAR) / (sd(ticker_returns[ticker_returns <
MAR]) * sqrt(annualization_factor))

#                     ticker_tracking_error <- sqrt(mean((ticker_returns - benchmark_ret)^2)) *
sqrt(annualization_factor)

#     ticker_info_ratio <- (ticker_return_annualized - benchmark_mean_ret) / ticker_tracking_error

#           modified_jensen_alpha_ticker <- ticker_return_annualized - (rf_return + ticker_beta *
(benchmark_mean_ret - rf_return))

#               modified_info_ratio_ticker <- (ticker_return_annualized - benchmark_mean_ret) /
ticker_tracking_error

#

#

#     # Calcul du skewness

#     ticker_skewness <- skewness(ticker_returns)

#     ticker_skewness <- ticker_skewness[[1]]

#

#     # Calcul du kurtosis
```

```
#     ticker_kurtosis <- kurtosis(ticker_returns)
#     ticker_kurtosis <- ticker_kurtosis[[1]]
#
#     # Test de Jarque-Bera
#     ticker_jb_test <- jarque.bera.test(na.omit(ticker_returns))
#     ticker_jb_p_value <- ticker_jb_test$p.value
#     ticker_jb_normality <- ifelse(ticker_jb_p_value > level_alpha, "Normal", "Not Normal")
#
#
#     c(annual_return = ticker_return_annualized,
#       annual_risk = ticker_risk_annualized,
#       beta = ticker_beta,
#       tracking_error = ticker_tracking_error,
#       sharpe_ratio = ticker_sharpe_ratio,
#       modified_SR = ticker_modified_SR,
#       info_ratio = ticker_info_ratio,
#       modified_info_ratio = modified_info_ratio_ticker,
#       jensen_alpha = ticker_alpha,
#       modified_jensen_alpha = modified_jensen_alpha_ticker,
#       sortino_ratio = ticker_sortino_ratio, #11
#       ESG = ticker_ESG,#12 avant amélioration
#       Skewness = ticker_skewness,
#       Kurtosis = ticker_kurtosis,
#       Jb.P_value = ticker_jb_p_value
#       # ,
#       # Jb_test_decision = ticker_jb_normality #16
#     )
#
#   })
```

```
#
#   print("--premier test --")
#   print(metrics_by_ticker)
#
#   esg_line <- which(row.names(metrics_by_ticker) == 'ESG')
#
#   print("--ESG line--")
#   print(esg_line)
#
#   print("-Class user weight-")
#   print(class(user_weights))
#   pf_ESG = sum(as.numeric(metrics_by_ticker[esg_line,]) * user_weights)
#
#   print("--pf_ESG--")
#   print(pf_ESG)
#
#         # pf_ESG = sum(metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),] *
user_weights)
#
#   # pf_ESG = c(pf_ESG)
#
#   # Modified Sharpe ratio #is row 12
#   Modified_SR <- ifelse(as.numeric(sharpe_ratio) > 0,
#                 as.numeric(sharpe_ratio)* (1 + pf_ESG)^sc,
#                 as.numeric(sharpe_ratio)* (1 + pf_ESG)^(-sc)
#   )
#
#                           mean_esg <- sum(as.numeric(metrics_by_ticker[esg_line,])) /
as.numeric(ncol(metrics_by_ticker))
```

```
#         # mean_esg <- sum(metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) / ncol(metrics_by_ticker)
#
#    # print("-- mean_esg --")
#    # print(mean_esg)
#
#
#    # metrics_by_ticker["Relative ESG", ] <- (metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) - mean_esg
#         # Relative_ESG <- (metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) - mean_esg
#    Relative_ESG <- as.numeric(metrics_by_ticker[esg_line, ]) - as.numeric(mean_esg)
#
#    # Stresser le relative ESG
#    # Chercher à les convertir tous en valeur positive
#    Relative_ESG <- (Relative_ESG^(-3)) / 3
#
#    metrics_by_ticker <- rbind(metrics_by_ticker, Relative_ESG)
#
#    rownames(metrics_by_ticker)[nrow(metrics_by_ticker)] <- "Relative_ESG"
#
#    # metrics_by_ticker[13,] is Relative_ESG
#    the_num <- which(row.names(metrics_by_ticker) == 'Relative_ESG')
#    print("-- Test sur la ligne Relative_ESG --")
#    print(the_num)
#
#    metrics_by_ticker[the_num,] <- ifelse(as.numeric(metrics_by_ticker[5,]) > 0,
#                         as.numeric(metrics_by_ticker[5,]) * (1 + as.numeric(metrics_by_ticker[the_num,]))^sc,
```

```
#                       as.numeric(metrics_by_ticker[5,]) * (1 + as.numeric(metrics_by_ticker[the_num,])
)^(-sc))
#
#   # Avant l'amélioration du code
#   # metrics_by_ticker[13,] <- ifelse(metrics_by_ticker[5,] > 0,
#   #                         metrics_by_ticker[5,] * (1 + metrics_by_ticker[13,] )^sc,
#   #                         metrics_by_ticker[5,] * (1 + metrics_by_ticker[13,] )^(-sc))
#
#
#
#   SR_with_MAESG <- NA #
#   metrics_by_ticker = rbind(metrics_by_ticker, SR_with_MAESG)
#
#   tot_row = nrow(metrics_by_ticker)
#
#   if(MA_ESG != 0){
#     metrics_by_ticker[tot_row,] <- (as.numeric(metrics_by_ticker[esg_line,]) - MA_ESG^(-3))/3
#     # metrics_by_ticker[tot_row,] <- (metrics_by_ticker["ESG",] - MA_ESG^(-3))/3
#   }else{
#     metrics_by_ticker[tot_row,] <- NA
#   }
#
#
#
#   # metrics_by_ticker["Relative_ESG", ] <- ifelse(metrics_by_ticker["sharpe_ratio", ] > 0,
#   #                         metrics_by_ticker["sharpe_ratio", ] * (1 + metrics_by_ticker["Relative_ESG",
])^sc,
#   #                         metrics_by_ticker["sharpe_ratio", ] * (1 + metrics_by_ticker["Relative_ESG",
])^(-sc)
#   # )
```

```
#
#    rownames(metrics_by_ticker) <- c("Annual return", "Annual risk", "Beta", "Tracking error",
#                                     "Sharpe ratio", "Modified Sharpe ratio", "Information ratio", "Modified information
ratio",
#                                      "Jensen alpha", "Modified Jensen alpha",
#                                      "Sortino ratio", "ESG",
#                                      "Skewness", "Kurtosis", "JB P Value",
#                                      # "JB test Decision",
#                                      "Relative Sharpe ratio", "Sharpe ratio with MAESG")
#
#    metrics_by_ticker <- metrics_by_ticker[c("ESG","Annual return", "Annual risk", "Beta", "Tracking
error",
#                                             "Sharpe ratio", "Modified Sharpe ratio", "Relative Sharpe ratio", "Sharpe ratio
with MAESG",
#                                             "Information ratio", "Modified information ratio",
#                                             "Jensen alpha", "Modified Jensen alpha",
#                                             "Sortino ratio",
#                                             "Skewness", "Kurtosis", "JB P Value"
#                                             # , "JB test Decision"
#    ),]
#
#
#
#    the_num1 <- which(row.names(metrics_by_ticker) == 'JB P Value')
#    # which(rownames(metrics_by_ticker) == "JB test Decision")
#
#    nbr = 1:17
#    # Ne pas arrondir JB Pvalue
#    remaining_rows <- nbr[nbr != the_num1]
#
```

```
#     metrics_by_ticker[remaining_rows,] <- round(as.numeric(metrics_by_ticker[remaining_rows,]),
6)
#
#    result <- list(
#      log_ret_tidy = log_ret_tidy,
#      df_with_ESG = df_with_ESG,
#      return_table = log_ret_xts,
#      pf_ESG = pf_ESG,
#      pf_annual_return = portfolio_ret,
#      pf_annual_risk = portfolio_risk[1],
#      pf_var = portfolio_var,
#      pf_tracking_error = tracking_error,
#      pf_beta = beta,
#      pf_beta_bull = beta_bull,
#      pf_beta_bear = beta_bear,
#      pf_sharpe_ratio = sharpe_ratio[1],
#      pf_Modified_SR = Modified_SR,
#      pf_info_ratio = info_ratio,
#      pf_modified_info_ratio = modified_info_ratio,
#      pf_jensen_alpha = alpha,
#      pf_modified_jensen_alpha = modified_jensen_alpha,
#      pf_sortino_ratio = sortino_ratio,
#      pf_skewness = skewness[[1]],
#      pf_kurtosis = kurtosis[[1]],
#      pf_jb.pvalue = jb_test_,
#      pf_jb_test.decision = jb_normality,
#      # metrics_by_ticker = as.data.frame(t(metrics_by_ticker))
#      metrics_by_ticker = as.data.frame(metrics_by_ticker)
#    )
```

```
#
#    return(result)
#
#  }
#
#
# }


# Ancienne version qui fonctionnait mais le benchmark devait être corrigé
# calculate_portfolio_metrics <- function(tickers,
#                          benchmark,
#                          MAR = 0, # Minimum acceptable Return
#                          MA_ESG = 0, # Minimum acceptable ESG
#                          rf_return = 0,
#                          sc = 0.05,
#                          start_date = Sys.Date() - 365*2,
#                          end_date = Sys.Date(),
#                          type_return = 'daily',
#                          return_method = 'log',
#                          user_weights = NULL) {
#  # Get stock prices
#  price_data <- tq_get(tickers, from = start_date, to = end_date, get = 'stock.prices')
#  # print("-- Head Price data --")
#  # print(head(price_data))
#
#  if(is.null(benchmark) || nchar(benchmark) == 0){
#    return(NULL)
#  } else{
```

```
#      risk_free_rate <- c("FR1YT=RR", 'US1YT=X', 'DE1YT=RR', "FR5YT=RR", 'US5YT=X', 'DE5YT=RR',
"FR10YT=RR", 'DE10YT=RR', 'US10YT=X')
#    if(benchmark %in% risk_free_rate){
#     # benchmark_data <-  ready_hcDt_new(ticker_info = benchmark,
#     #                      "D", start_date, end_date)
#
#     benchmark_data <-  fetch_inv_prices(ticker_info = benchmark,
#                                 time_frame = 'Daily',
#                                 from = start_date,
#                                 to = end_date)
#
#     benchmark_data$Date <- as.Date(benchmark_data$Date)
#
#     benchmark_ret <- benchmark_data %>%
#       tq_transmute(select = Close,
#                    mutate_fun = periodReturn,
#                    period = type_return,
#                    col_rename = 'benchmark_ret',
#                    type = return_method) %>%
#       tk_xts(date_var = "Date", silent = TRUE)
#
#    } else{
#     benchmark_data <- tq_get(benchmark, from = start_date, to = end_date, get = 'stock.prices')
#
#     benchmark_ret <- benchmark_data %>%
#       tq_transmute(select = adjusted,
#                    mutate_fun = periodReturn,
#                    period = type_return,
#                    col_rename = 'benchmark_ret',
```

```
#                 type = return_method) %>%
#     tk_xts(date_var = "date", silent = TRUE)
#   }
#
#   # Calculate returns
#   log_ret_tidy <- price_data %>%
#    group_by(symbol) %>%
#    tq_transmute(select = adjusted,
#                 mutate_fun = periodReturn,
#                 period = type_return,
#                 col_rename = 'ret',
#                 type = return_method)
#
#   # print("-- Head log_ret_tidy --")
#   # print(head(log_ret_tidy))
#
#   # Get ESG Data
#   df_ = log_ret_tidy
#   df_with_ESG <- list()
#
#   for (ticker in unique(df_$symbol)) {
#    dt_ <- df_ %>%
#     dplyr::filter(symbol == ticker)
#    # print(head(dt_))
#
#    dt_ESG = get_historic_esg(ticker)
#    if(!is.null(dt_ESG)){
#     dt_ESG = na.omit(dt_ESG)
#     names(dt_ESG) = c("date", "ESG", "E_Score", "S_Score", "G_Score")
```

```
#       dt_ESG = dt_ESG[, c("date", "ESG")]
#
#       # start_date <- as.Date(input$date[1])
#       # end_date <- as.Date(input$date[2])
#
#       # Générez une séquence de dates entre start_date et end_date (inclusivement) pour les jours ouvrables (business days)
#       date_range <- seq(from = start_date, to = end_date, by = "days")
#           workday_dates <- date_range[weekdays(date_range) %in% c("lundi", "mardi", "mercredi", "jeudi", "vendredi")]
#
#       # Créez une dataframe avec les dates
#       df <- data.frame(date = workday_dates)
#
#       # Fusionnez les dataframes en utilisant la colonne "Date"
#       merged_df <- merge(df, dt_ESG, by = "date", all.x = TRUE)
#
#       df = merged_df%>%tidyr::fill(ESG)
#
#       na_elm = which(is.na(df[,"ESG"]))
#       non_nul_elm = which(!is.na(df[,"ESG"]))
#
#       # Si les premiers elements sont encore des NA
#       if(length(na_elm)!=0){
#         if(length(non_nul_elm)!=0){
#           df[na_elm,"ESG"] = 0
#
#         }
#
```

```
#       }
#
#       # Fusionnez les dataframes en utilisant la colonne "Date"
#       df_final <- merge(dt_, df, by = "date", all.x = TRUE)
#
#       df_with_ESG[[ticker]] <- df_final%>%fill(ESG)
#
#
#     }else{
#       dt_$ESG <- 0
#       df_with_ESG[[ticker]] <- dt_
#
#     }
#
#   }
#
#   if(length(df_with_ESG) >= 1){
#     # df_with_ESG <- do.call(rbind, df_with_ESG)
#     df_with_ESG = dplyr::bind_rows(df_with_ESG)
#   }
#
#   log_ret_xts <- log_ret_tidy %>%
#     spread(symbol, value = ret) %>%
#     tk_xts(date_var = "date", silent = TRUE)
#
#   # benchmark_ret <- benchmark_data %>%
#   #  tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename
= 'benchmark_ret', type = return_method) %>%
#   #  tk_xts()
```

```
#
#   mean_ret <- colMeans(log_ret_xts)
#
#   annualization_factor <- switch(type_return,
#                          "daily" = 252,
#                          "weekly" = 52,
#                          "monthly" = 12,
#                          "yearly" = 1,
#                          stop("type_return doit être 'daily', 'weekly', 'monthly' ou 'yearly'"))
#
#   cov_mat <- cov(log_ret_xts) * annualization_factor
#
#   if(is.character(user_weights)){
#    stop("User weights should be a numeric vector!")
#   }
#
#   if (!is.null(user_weights)) {
#
#    if(sum(user_weights) < 0 ){
#      stop("User weights should be positive!")
#    }
#
#    if(sum(user_weights) == 0 ){
#      l_ticker = length(tickers)
#      user_weights = rep((1/l_ticker), l_ticker)
#    } else if(sum(user_weights) > 0 && sum(user_weights) <= 1 ){
#      user_weights = user_weights
#    } else if(sum(user_weights) > 1 && sum(user_weights) <= 100 ){
#      user_weights = user_weights/100
```

```
#     } else{
#       user_weights <- user_weights / sum(user_weights)
#     }
#
#   } else{
#
#     user_weights <- rep(1 / length(tickers), length(tickers))
#
#   }
#
#   # if (is.null(user_weights)) {
#   #   user_weights <- rep(1 / length(tickers), length(tickers))
#   # }
#
#   portfolio_ret <- sum(user_weights * mean_ret)
#   portfolio_ret <- ((portfolio_ret + 1)^annualization_factor) - 1
#   portfolio_risk <- sqrt(t(user_weights) %*% (cov_mat %*% user_weights))
#
#   weights <- matrix(user_weights, ncol = 1)
#
#   portfolio_var <- as.numeric(t(weights) %*% var(log_ret_xts) %*% weights)
#
#   pf_ret <- Return.portfolio(log_ret_xts, weights = user_weights)
#
#   log_ret_xts$portfolio.returns <- pf_ret$portfolio.returns
#
#   if (is.numeric(rf_return) && length(rf_return) == 1) {
#     rf_return <- rf_return / annualization_factor
#   } else if (is.character(rf_return) && length(rf_return) == 1) {
```

```
#     rf_data <- tq_get(rf_return, from = start_date, to = end_date, get = 'stock.prices')
#     rf_log_ret_tidy <- rf_data %>%
#       tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename = 'rf_return', type = return_method)
#     rf_return <- mean(rf_log_ret_tidy$rf_return)
#   } else {
#     stop("rf doit être un taux numérique ou un ticker valide.")
#   }
#
#   sharpe_ratio <- (portfolio_ret - rf_return) / portfolio_risk
#
#   benchmark_mean_ret <- mean(benchmark_ret)
#
#             tracking_error <- sqrt(mean((rowMeans(log_ret_xts) - benchmark_mean_ret)^2)) * sqrt(annualization_factor)
#
#   # Align dates of log_ret_xts and benchmark_ret
#   common_dates <- as.Date(intersect(index(na.omit(log_ret_xts)),
#                                     index(na.omit(benchmark_ret))))
#
#   # common_dates <- index(na.omit(log_ret_xts)) %in% index(na.omit(benchmark_ret))
#   log_ret_xts_aligned <- log_ret_xts[common_dates]
#   benchmark_ret_aligned <- benchmark_ret[common_dates]
#
#   # print("-- Dim log_ret_xts --")
#   # print(dim(log_ret_xts))
#   # print(head(log_ret_xts,3))
#
#   # print("-Dim benchmark_ret")
```

```
#    # print(dim(benchmark_ret))

#    # print(head(benchmark_ret,3))

#

#    beta <- cov(as.numeric(rowMeans(log_ret_xts_aligned)), as.numeric(benchmark_ret_aligned)) / var(as.numeric(benchmark_ret_aligned))

#

#            # beta <- cov(as.numeric(rowMeans(log_ret_xts)), as.numeric(benchmark_ret)) / var(as.numeric(benchmark_ret))

#

#        alpha <- jenson.alpha(log_ret_xts_aligned$portfolio.returns, benchmark_ret_aligned, rf = rf_return)

#    # alpha <- jenson.alpha(log_ret_xts$portfolio.returns, benchmark_ret, rf = rf_return)

#

#    # Alpha de Jensen modifié

#    modified_jensen_alpha <- portfolio_ret - (rf_return + beta * (benchmark_mean_ret - rf_return))

#

#    print("Modified alpha")

#    print(modified_jensen_alpha)

#

#    print("-- rowmeans --")

#    # print(rowMeans(log_ret_xts))

#

#    print("--La classe de MAR--")

#    print(class(MAR))

#    print(MAR)

#

#    MAR = as.numeric(MAR)

#

#            downside_deviation <- sqrt(mean(ifelse(rowMeans(log_ret_xts) < as.numeric(MAR), (rowMeans(log_ret_xts) - as.numeric(MAR))^2, 0))) * sqrt(annualization_factor)
```

```
#   print("downside_deviation")
#   print(downside_deviation)
#
#   sortino_ratio <- (portfolio_ret - MAR) / downside_deviation
#   print("sortino ratio")
#   print(sortino_ratio)
#
#
#   # info_ratio <- ratio.information(log_ret_xts$portfolio.returns, benchmark_ret)
#   # print("--Test head log_ret_xts_aligned$portfolio.returns --")
#   # print(head(log_ret_xts_aligned$portfolio.returns))
#
#   # print('-- Head benchmark_ret_aligned --')
#   # print(head(benchmark_ret_aligned, 3))
#
#   info_ratio <- ratio.information(log_ret_xts_aligned$portfolio.returns, benchmark_ret_aligned)
#
#   print("-info_ratio-")
#   print(info_ratio)
#
#   modified_info_ratio <- (portfolio_ret - benchmark_mean_ret) / tracking_error
#
#   print("--modified_info_ratio--")
#   print(modified_info_ratio)
#
#   # Calculate metrics by ticker
#   metrics_by_ticker <- sapply(tickers, function(ticker) {
#     ticker_returns <- log_ret_xts[, ticker]
#
```

```
#     common_dates <- as.Date(intersect(na.omit(ticker_returns),
#                     index(na.omit(benchmark_ret))))
#
#     # common_dates <- index(na.omit(ticker_returns)) %in% index(na.omit(benchmark_ret))
#     ticker_returns_aligned <- ticker_returns[common_dates]
#     benchmark_ret_aligned <- benchmark_ret[common_dates]
#
#     print("class of ticker_returns")
#     print((class(ticker_returns)))
#     print(head(ticker_returns, 6))
#
#     ticker_returns = as.numeric(ticker_returns)
#
#
#     ticker_return_annualized <- ((mean(ticker_returns) + 1)^annualization_factor) - 1
#
#     print("--ticker_return_annualized --")
#
#     print(ticker_return_annualized)
#     print(class(rf_return))
#
#     ticker_risk_annualized <- sd(ticker_returns) * sqrt(annualization_factor)
#     # ticker_beta <- cov(ticker_returns, benchmark_ret) / var(benchmark_ret)
#                         ticker_beta <- cov(ticker_returns_aligned, benchmark_ret_aligned) /
var(benchmark_ret_aligned)
#       ticker_alpha <- ticker_return_annualized - (rf_return + ticker_beta * (benchmark_mean_ret -
rf_return))
#     ticker_sharpe_ratio <- (ticker_return_annualized - rf_return) / ticker_risk_annualized
#                                         ticker_ESG <- (df_with_ESG[df_with_ESG$symbol ==
ticker,4])[length(df_with_ESG[df_with_ESG$symbol == ticker,4])]
```

```
#     ticker_modified_SR <- ifelse(ticker_sharpe_ratio > 0,
#                          ticker_sharpe_ratio * (1 + ticker_ESG)^sc,
#                          ticker_sharpe_ratio * (1 + ticker_ESG)^(-sc)
#     )
#       ticker_sortino_ratio <- (ticker_return_annualized - MAR) / (sd(ticker_returns[ticker_returns < MAR]) * sqrt(annualization_factor))
#
#                     ticker_tracking_error <- sqrt(mean((ticker_returns - benchmark_ret)^2)) * sqrt(annualization_factor)
#     ticker_info_ratio <- (ticker_return_annualized - benchmark_mean_ret) / ticker_tracking_error
#           modified_jensen_alpha_ticker <- ticker_return_annualized - (rf_return + ticker_beta * (benchmark_mean_ret - rf_return))
#               modified_info_ratio_ticker <- (ticker_return_annualized - benchmark_mean_ret) / ticker_tracking_error
#
#
#
#     c(annual_return = ticker_return_annualized,
#       annual_risk = ticker_risk_annualized,
#       beta = ticker_beta,
#       tracking_error = ticker_tracking_error,
#       sharpe_ratio = ticker_sharpe_ratio,
#       modified_SR = ticker_modified_SR,
#       info_ratio = ticker_info_ratio,
#       modified_info_ratio = modified_info_ratio_ticker,
#       jensen_alpha = ticker_alpha,
#       modified_jensen_alpha = modified_jensen_alpha_ticker,
#       sortino_ratio = ticker_sortino_ratio,
#       ESG = ticker_ESG)
#   })
```

```
#
#   pf_ESG = sum(metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),] * user_weights)
#
#    # pf_ESG = c(pf_ESG)
#
#    # Modified Sharpe ratio
#    Modified_SR <- ifelse(sharpe_ratio > 0,
#                    sharpe_ratio* (1 + pf_ESG)^sc,
#                    sharpe_ratio* (1 + pf_ESG)^(-sc)
#    )
#
#          mean_esg <- sum(metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) /
ncol(metrics_by_ticker)
#
#    # print("-- mean_esg --")
#    # print(mean_esg)
#
#
#    # metrics_by_ticker["Relative ESG", ] <- (metrics_by_ticker[which(rownames(metrics_by_ticker)
== 'ESG'),]) - mean_esg
#    Relative_ESG <- (metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) - mean_esg
#
#    # Stresser le relative ESG
#    # Chercher à les convertir tous en valeur positive
#    Relative_ESG <- (Relative_ESG^(-3)) / 3
#
#    metrics_by_ticker <- rbind(metrics_by_ticker, Relative_ESG)
#
#    rownames(metrics_by_ticker)[nrow(metrics_by_ticker)] <- "Relative_ESG "
```

```
#
#    # metrics_by_ticker[13,] is Relative_ESG
#    metrics_by_ticker[13,] <- ifelse(metrics_by_ticker[5,] > 0,
#                            metrics_by_ticker[5,]  * (1 + metrics_by_ticker[13,] )^sc,
#                            metrics_by_ticker[5,]  * (1 + metrics_by_ticker[13,] )^(-sc))
#
#
#    SR_with_MAESG <- NA
#    metrics_by_ticker = rbind(metrics_by_ticker, SR_with_MAESG)
#
#    if(MA_ESG != 0){
#     metrics_by_ticker[14,] <- (metrics_by_ticker[12,] - MA_ESG^(-3))/3
#    }else{
#     metrics_by_ticker[14,] <- NA
#    }
#
#
#
#    # metrics_by_ticker["Relative_ESG", ] <- ifelse(metrics_by_ticker["sharpe_ratio", ] > 0,
#   #                             metrics_by_ticker["sharpe_ratio", ] * (1 + metrics_by_ticker["Relative_ESG",
])^sc,
#   #                             metrics_by_ticker["sharpe_ratio", ] * (1 + metrics_by_ticker["Relative_ESG",
])^(-sc)
#    # )
#
#    rownames(metrics_by_ticker) <- c("Annual return", "Annual risk", "Beta", "Tracking error",
#                            "Sharpe ratio", "Modified Sharpe ratio", "Information ratio", "Modified information
ratio",
#                             "Jensen alpha", "Modified Jensen alpha",
#                             "Sortino ratio", "ESG", "Relative Sharpe ratio", "Sharpe ratio with MAESG")
```

```
#
#    metrics_by_ticker <- metrics_by_ticker[c("ESG","Annual return", "Annual risk", "Beta", "Tracking error",
#                                             "Sharpe ratio", "Modified Sharpe ratio", "Relative Sharpe ratio", "Sharpe ratio with MAESG",
#                                             "Information ratio", "Modified information ratio",
#                                             "Jensen alpha", "Modified Jensen alpha",
#                                             "Sortino ratio"),]
#
#    metrics_by_ticker <- round(metrics_by_ticker, 6)
#
#    result <- list(
#      log_ret_tidy = log_ret_tidy,
#      df_with_ESG = df_with_ESG,
#      return_table = log_ret_xts,
#      pf_ESG = pf_ESG,
#      pf_annual_return = portfolio_ret,
#      pf_annual_risk = portfolio_risk[1],
#      pf_var = portfolio_var,
#      pf_tracking_error = tracking_error,
#      pf_beta = beta,
#      pf_sharpe_ratio = sharpe_ratio[1],
#      pf_Modified_SR = Modified_SR,
#      pf_info_ratio = info_ratio,
#      pf_modified_info_ratio = modified_info_ratio,
#      pf_jensen_alpha = alpha,
#      pf_modified_jensen_alpha = modified_jensen_alpha,
#      pf_sortino_ratio = sortino_ratio,
#      # metrics_by_ticker = as.data.frame(t(metrics_by_ticker))
```

```
#     metrics_by_ticker = as.data.frame(metrics_by_ticker)
#   )
#
#   return(result)
#
# }
#
#
# }

# Define the function
# Fonctionne aussi mais ne règle pas tout
# calculate_portfolio_metrics <- function(tickers,
#                         benchmark,
#                         MAR = 0, # Minimum acceptable Return
#                         MA_ESG = 0, # Minimum acceptable ESG
#                         rf_return = 0,
#                         sc = 0.05,
#                         start_date = Sys.Date() - 365*2,
#                         end_date = Sys.Date(),
#                         type_return = 'daily',
#                         user_weights = NULL) {
#  # Get stock prices
#  price_data <- tq_get(tickers, from = start_date, to = end_date, get = 'stock.prices')
#  print("-- Head Price data --")
#  print(head(price_data))
#
#  benchmark_data <- tq_get(benchmark, from = start_date, to = end_date, get = 'stock.prices')
#
```

```
#  # Calculate returns
#  log_ret_tidy <- price_data %>%
#    group_by(symbol) %>%
#    tq_transmute(select = adjusted,
#                 mutate_fun = periodReturn,
#                 period = type_return,
#                 col_rename = 'ret',
#                 type = 'log')
#
#  # print("-- Head log_ret_tidy --")
#  # print(head(log_ret_tidy))
#
#  # Get ESG Data
#  df_ = log_ret_tidy
#  df_with_ESG <- list()
#
#  for (ticker in unique(df_$symbol)) {
#    dt_ <- df_ %>%
#      dplyr::filter(symbol == ticker)
#    # print(head(dt_))
#
#    dt_ESG = get_historic_esg(ticker)
#    if(!is.null(dt_ESG)){
#      dt_ESG = na.omit(dt_ESG)
#      names(dt_ESG) = c("date", "ESG", "E_Score", "S_Score", "G_Score")
#      dt_ESG = dt_ESG[, c("date", "ESG")]
#
#      # start_date <- as.Date(input$date[1])
#      # end_date <- as.Date(input$date[2])
```

```
#
#      # Générez une séquence de dates entre start_date et end_date (inclusivement) pour les jours ouvrables (business days)
#      date_range <- seq(from = start_date, to = end_date, by = "days")
#          workday_dates <- date_range[weekdays(date_range) %in% c("lundi", "mardi", "mercredi", "jeudi", "vendredi")]
#
#      # Créez une dataframe avec les dates
#      df <- data.frame(date = workday_dates)
#
#      # Fusionnez les dataframes en utilisant la colonne "Date"
#      merged_df <- merge(df, dt_ESG, by = "date", all.x = TRUE)
#
#      df = merged_df%>%fill(ESG)
#
#      na_elm = which(is.na(df[,"ESG"]))
#      non_nul_elm = which(!is.na(df[,"ESG"]))
#
#      # Si les premiers elements sont encore des NA
#      if(length(na_elm)!=0){
#        if(length(non_nul_elm)!=0){
#          df[na_elm,"ESG"] = 0
#
#        }
#
#      }
#
#      # Fusionnez les dataframes en utilisant la colonne "Date"
#      df_final <- merge(dt_, df, by = "date", all.x = TRUE)
```

```
#
#     df_with_ESG[[ticker]] <- df_final%>%fill(ESG)
#
#
#    }else{
#     dt_$ESG <- 0
#     df_with_ESG[[ticker]] <- dt_
#
#    }
#
#  }
#
#  if(length(df_with_ESG) >= 1){
#    # df_with_ESG <- do.call(rbind, df_with_ESG)
#    df_with_ESG = dplyr::bind_rows(df_with_ESG)
#  }
#
#  log_ret_xts <- log_ret_tidy %>%
#    spread(symbol, value = ret) %>%
#    tk_xts()
#
#  benchmark_ret <- benchmark_data %>%
#    tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename =
'benchmark_ret', type = 'log') %>%
#    tk_xts()
#
#  mean_ret <- colMeans(log_ret_xts)
#
#  annualization_factor <- switch(type_return,
```

```
#                       "daily" = 252,
#                       "weekly" = 52,
#                       "monthly" = 12,
#                       "yearly" = 1,
#                       stop("type_return doit être 'daily', 'weekly', 'monthly' ou 'yearly'"))
#
#  cov_mat <- cov(log_ret_xts) * annualization_factor
#
#  if(is.character(user_weights)){
#    stop("User weights should be a numeric vector!")
#  }
#
#  if (!is.null(user_weights)) {
#
#    if(sum(user_weights) < 0 ){
#      stop("User weights should be positive!")
#    }
#
#    if(sum(user_weights) == 0 ){
#      l_ticker = length(tickers)
#      user_weights = rep((1/l_ticker), l_ticker)
#    } else if(sum(user_weights) > 0 && sum(user_weights) <= 1 ){
#      user_weights = user_weights
#    } else if(sum(user_weights) > 1 && sum(user_weights) <= 100 ){
#      user_weights = user_weights/100
#    } else{
#      user_weights <- user_weights / sum(user_weights)
#    }
#
```

```
#   } else{
#
#     user_weights <- rep(1 / length(tickers), length(tickers))
#
#   }
#
#   # if (is.null(user_weights)) {
#   #   user_weights <- rep(1 / length(tickers), length(tickers))
#   # }
#
#   portfolio_ret <- sum(user_weights * mean_ret)
#   portfolio_ret <- ((portfolio_ret + 1)^annualization_factor) - 1
#   portfolio_risk <- sqrt(t(user_weights) %*% (cov_mat %*% user_weights))
#
#   weights <- matrix(user_weights, ncol = 1)
#
#   portfolio_var <- as.numeric(t(weights) %*% var(log_ret_xts) %*% weights)
#
#   pf_ret <- Return.portfolio(log_ret_xts, weights = user_weights)
#
#   log_ret_xts$portfolio.returns <- pf_ret$portfolio.returns
#
#   if (is.numeric(rf_return) && length(rf_return) == 1) {
#     rf_return <- rf_return / annualization_factor
#   } else if (is.character(rf_return) && length(rf_return) == 1) {
#     rf_data <- tq_get(rf_return, from = start_date, to = end_date, get = 'stock.prices')
#     rf_log_ret_tidy <- rf_data %>%
#       tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename
= 'rf_return', type = 'log')
```

```
#    rf_return <- mean(rf_log_ret_tidy$rf_return)
#  } else {
#    stop("rf doit être un taux numérique ou un ticker valide.")
#  }
#
#  sharpe_ratio <- (portfolio_ret - rf_return) / portfolio_risk
#
#  benchmark_mean_ret <- mean(benchmark_ret)
#
#          tracking_error <- sqrt(mean((rowMeans(log_ret_xts) - benchmark_mean_ret)^2)) * sqrt(annualization_factor)
#
#          beta <- cov(as.numeric(rowMeans(log_ret_xts)), as.numeric(benchmark_ret)) / var(as.numeric(benchmark_ret))
#
#  alpha <- jenson.alpha(log_ret_xts$portfolio.returns, benchmark_ret, rf = rf_return)
#
#  # Alpha de Jensen modifié
#  modified_jensen_alpha <- portfolio_ret - (rf_return + beta * (benchmark_mean_ret - rf_return))
#
#  print(log_ret_xts)
#  print(MAR)
#
#  # log_ret_xts <- as.numeric(log_ret_xts)
#  MAR <- as.numeric(MAR)
#
#  # Calculate downside deviation
#    # downside_deviation <- sqrt(mean((rowMeans(log_ret_xts)[rowMeans(log_ret_xts) < MAR] - MAR)^2)) * sqrt(annualization_factor)
#
```

```
#
#   downside_deviation <- sqrt(mean(ifelse(rowMeans(log_ret_xts) < MAR, (rowMeans(log_ret_xts) - MAR)^2, 0))) * sqrt(annualization_factor)
#   sortino_ratio <- (portfolio_ret - MAR) / downside_deviation
#
#   info_ratio <- ratio.information(log_ret_xts$portfolio.returns, benchmark_ret)
#
#   modified_info_ratio <- (portfolio_ret - benchmark_mean_ret) / tracking_error
#   # Calculate metrics by ticker
#   metrics_by_ticker <- sapply(tickers, function(ticker) {
#     ticker_returns <- log_ret_xts[, ticker]
#     ticker_return_annualized <- ((mean(ticker_returns) + 1)^annualization_factor) - 1
#     ticker_risk_annualized <- sd(ticker_returns) * sqrt(annualization_factor)
#     ticker_beta <- cov(ticker_returns, benchmark_ret) / var(benchmark_ret)
#       ticker_alpha <- ticker_return_annualized - (rf_return + ticker_beta * (benchmark_mean_ret - rf_return))
#     ticker_sharpe_ratio <- (ticker_return_annualized - rf_return) / ticker_risk_annualized
#                                       ticker_ESG <- (df_with_ESG[df_with_ESG$symbol == ticker,4])[length(df_with_ESG[df_with_ESG$symbol == ticker,4])]
#     ticker_modified_SR <- ifelse(ticker_sharpe_ratio > 0,
#                          ticker_sharpe_ratio * (1 + ticker_ESG)^sc,
#                          ticker_sharpe_ratio * (1 + ticker_ESG)^(-sc)
#     )
#
#       ticker_sortino_ratio <- (ticker_return_annualized - MAR) / (sd(ticker_returns[ticker_returns < MAR]) * sqrt(annualization_factor))
#                    ticker_tracking_error <- sqrt(mean((ticker_returns - benchmark_ret)^2)) * sqrt(annualization_factor)
#     ticker_info_ratio <- (ticker_return_annualized - benchmark_mean_ret) / ticker_tracking_error
#           modified_jensen_alpha_ticker <- ticker_return_annualized - (rf_return + ticker_beta * (benchmark_mean_ret - rf_return))
```

```
#             modified_info_ratio_ticker <- (ticker_return_annualized - benchmark_mean_ret) / ticker_tracking_error
#
#
#   c(annual_return = ticker_return_annualized,
#     annual_risk = ticker_risk_annualized,
#     beta = ticker_beta,
#     tracking_error = ticker_tracking_error,
#     sharpe_ratio = ticker_sharpe_ratio,
#     modified_SR = ticker_modified_SR,
#     info_ratio = ticker_info_ratio,
#     modified_info_ratio = modified_info_ratio_ticker,
#     jensen_alpha = ticker_alpha,
#     modified_jensen_alpha = modified_jensen_alpha_ticker,
#     sortino_ratio = ticker_sortino_ratio,
#     ESG = ticker_ESG)
#  })
#
#  pf_ESG = sum(metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),] * user_weights)
#
#  # pf_ESG = c(pf_ESG)
#
#  # Modified Sharpe ratio
#  Modified_SR <- ifelse(sharpe_ratio > 0,
#                 sharpe_ratio* (1 + pf_ESG)^sc,
#                 sharpe_ratio* (1 + pf_ESG)^(-sc)
#  )
#
```

```
#       mean_esg <- sum(metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) / ncol(metrics_by_ticker)
#
#  print("-- mean_esg --")
#  print(mean_esg)
#   # metrics_by_ticker["Relative ESG", ] <- (metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) - mean_esg
#  Relative_ESG <- (metrics_by_ticker[which(rownames(metrics_by_ticker) == 'ESG'),]) - mean_esg
#
#  # Stresser le relative ESG
#  # Chercher à les convertir tous en valeur positive
#  Relative_ESG <- (Relative_ESG^(-3)) / 3
#
#  metrics_by_ticker <- rbind(metrics_by_ticker, Relative_ESG)
#
#  rownames(metrics_by_ticker)[nrow(metrics_by_ticker)] <- "Relative_ESG "
#
#  # metrics_by_ticker[13,] is Relative_ESG
#  metrics_by_ticker[13,] <- ifelse(metrics_by_ticker[5,] > 0,
#                        metrics_by_ticker[5,]  * (1 + metrics_by_ticker[13,] )^sc,
#                        metrics_by_ticker[5,]  * (1 + metrics_by_ticker[13,] )^(-sc))
#
#
#  SR_with_MAESG <- NA
#  metrics_by_ticker = rbind(metrics_by_ticker, SR_with_MAESG)
#
#  if(MA_ESG != 0){
#    metrics_by_ticker[14,] <- (metrics_by_ticker[12,] - MA_ESG^(-3))/3
#  }else{
```

```
#    metrics_by_ticker[14,] <- NA
#  }
#
#
#
#   # metrics_by_ticker["Relative_ESG", ] <- ifelse(metrics_by_ticker["sharpe_ratio", ] > 0,
#  #                          metrics_by_ticker["sharpe_ratio", ] * (1 + metrics_by_ticker["Relative_ESG",
])^sc,
#  #                          metrics_by_ticker["sharpe_ratio", ] * (1 + metrics_by_ticker["Relative_ESG",
])^(-sc)
#   # )
#
#   rownames(metrics_by_ticker) <- c("Annual return", "Annual risk", "Beta", "Tracking error",
#                       "Sharpe ratio", "Modified Sharpe ratio", "Information ratio", "Modified information
ratio",
#                       "Jensen alpha", "Modified Jensen alpha",
#                       "Sortino ratio", "ESG", "Relative Sharpe ratio", "Sharpe ratio with MAESG")
#
#   metrics_by_ticker <- metrics_by_ticker[c("ESG","Annual return", "Annual risk", "Beta", "Tracking
error",
#                            "Sharpe ratio", "Modified Sharpe ratio", "Relative Sharpe ratio", "Sharpe ratio
with MAESG",
#                           "Information ratio", "Modified information ratio",
#                           "Jensen alpha", "Modified Jensen alpha",
#                           "Sortino ratio"),]
#
#   metrics_by_ticker <- round(metrics_by_ticker, 6)
#
#   result <- list(
#    log_ret_tidy = log_ret_tidy,
```

```
#    df_with_ESG = df_with_ESG,
#    return_table = log_ret_xts,
#    pf_ESG = pf_ESG,
#    pf_annual_return = portfolio_ret,
#    pf_annual_risk = portfolio_risk[1],
#    pf_var = portfolio_var,
#    pf_tracking_error = tracking_error,
#    pf_beta = beta,
#    pf_sharpe_ratio = sharpe_ratio[1],
#    pf_Modified_SR = Modified_SR,
#    pf_info_ratio = info_ratio,
#    pf_modified_info_ratio = modified_info_ratio,
#    pf_jensen_alpha = alpha,
#    pf_modified_jensen_alpha = modified_jensen_alpha,
#    pf_sortino_ratio = sortino_ratio,
#    # metrics_by_ticker = as.data.frame(t(metrics_by_ticker))
#    metrics_by_ticker = as.data.frame(metrics_by_ticker)
#  )
#
#  return(result)
# }


# # Example usage
# metrics <- calculate_portfolio_metrics(
#  tickers = c("AAPL", "GOOGL", "MSFT", "AMZN"),
#  benchmark = "SPY",
#  rf_return = 0.05,
#  MAR = 0.01,
#  MA_ESG = 10,
```

```
#   start_date = Sys.Date() - 365*2,
#   end_date = Sys.Date(),
#   type_return = 'daily',
#   user_weights = c(0.22, 0.08, 0.30, 0.40)
# )



###############

# Version améliorée yf_get_keys.stats
yf_get_keys.stats <- function(ticker, full = TRUE) {

 ticker = toupper(ticker)

 if(all(ticker == "")){
  # if(length(ticker) == 1  && ticker == ''){
  return(NULL)
 }

 #     httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))

 # Fonction pour obtenir les données ESG pour un seul ticker
 get_keys_for_ticker <- function(ticker, full = TRUE) {

  # ticker = toupper(ticker)

  #     httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))
```

```
  url = glue::glue("https://finance.yahoo.com/quote/{ticker}/key-statistics/")
  # url = paste0("https://finance.yahoo.com/quote/", ticker, "/key-statistics/")
  # url <- "https://finance.yahoo.com/quote/KO/key-statistics/"

  # Envoyer la requête
  UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"

  cookies = c(
    A1 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
    GUC = "AQABCAFm-EdnKEIeOgSl&s=AQAAAIJTL2AD&g=Zvb4qg",
    A3 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
    A1S = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
    cmp = "t=1728944112&j=1&u=1---&v=48",
    EuConsent = "CQFm5cAQFm5cAAOACKFRBLFgAAAAAAAAACiQAAAAAAAA",
    PRF = "t=ADP%2BKO%2BGIS%2BAAPL%2B%5EFVX%2BGNFT.PA%2BMSFT%2BA%2BGOOG%2BBA%2BAL%2BRELIANCE.NS%2BVIIIX%2BFSKAX%2BFCNTX&newChartbetateaser=0%2C1719358224417"
  )

  headers = c(
    accept = "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
    `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",
    `cache-control` = "no-cache",
    pragma = "no-cache",
    priority = "u=0, i",
```

```
  `sec-ch-ua` = '"Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"',
  `sec-ch-ua-mobile` = "?0",
  `sec-ch-ua-platform` = '"Windows"',
  `sec-fetch-dest` = "document",
  `sec-fetch-mode` = "navigate",
  `sec-fetch-site` = "none",
  `sec-fetch-user` = "?1",
  `upgrade-insecure-requests` = "1",
  `user-agent` = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36"
 )

 response <- httr::GET(url,
                httr::add_headers(.headers=headers),
                httr::set_cookies(.cookies = cookies))

 # Faire la requête GET avec les en-têtes spécifiés
 # response <- GET(
 #  url,
 #  add_headers(
 #                                                `accept` = "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
 #    `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",
 #    `cache-control` = "no-cache",
 #    `pragma` = "no-cache",
 #    `priority` = "u=0, i",
 #                `sec-ch-ua` = "\"Google Chrome\";v=\"125\", \"Chromium\";v=\"125\", \"Not.A/Brand\";v=\"24\"",
 #    `sec-ch-ua-mobile` = "?0",
```

```
#    `sec-ch-ua-platform` = "\"Windows\"",
#    `sec-fetch-dest` = "document",
#    `sec-fetch-mode` = "navigate",
#    `sec-fetch-site` = "none",
#    `sec-fetch-user` = "?1",
#    `upgrade-insecure-requests` = "1"
#  )
# )

###
keys_not_available = paste0("https://finance.yahoo.com/quote/", ticker, "/")

if(response$url == keys_not_available){

 df = structure(list(Info = c("Ticker", "Market Cap", "Enterprise Value",
                              "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
                              "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA",
                              "Fiscal Year Ends", "Most Recent Quarter  (mrq)", "Profit Margin",
                              "Operating Margin  (ttm)", "Return on Assets  (ttm)", "Return on Equity  (ttm)",
                              "Revenue  (ttm)", "Revenue Per Share  (ttm)", "Quarterly Revenue Growth  (yoy)",
                              "Gross Profit  (ttm)", "EBITDA", "Net Income Avi to Common  (ttm)",
                              "Diluted EPS  (ttm)", "Quarterly Earnings Growth  (yoy)", "Total Cash  (mrq)",
                              "Total Cash Per Share  (mrq)", "Total Debt  (mrq)", "Total Debt/Equity  (mrq)",
                              "Current Ratio  (mrq)", "Book Value Per Share  (mrq)", "Operating Cash Flow  (ttm)",
                              "Levered Free Cash Flow  (ttm)", "Beta (5Y Monthly)", "52 Week Range 3",
                              "S&P 500 52-Week Change 3", "52 Week High 3", "52 Week Low 3",
                              "50-Day Moving Average 3", "200-Day Moving Average 3", "Avg Vol (3 month) 3",
                              "Avg Vol (10 day) 3", "Shares Outstanding 5", "Implied Shares Outstanding 6",
                              "Float 8", "% Held by Insiders 1", "% Held by Institutions 1",
```

```
                    "Shares Short  4", "Short Ratio  4", "Short % of Float  4",
                    "Short % of Shares Outstanding  4", "Shares Short (prior month 5/15/2024) 4",
                    "Forward Annual Dividend Rate 4", "Forward Annual Dividend Yield 4",
                    "Trailing Annual Dividend Rate 3", "Trailing Annual Dividend Yield 3",
                    "5 Year Average Dividend Yield 4", "Payout Ratio 4", "Dividend Date 3",
                    "Ex-Dividend Date 4", "Last Split Factor 2", "Last Split Date 3"
 ), Detail = c(ticker, rep('-',60))),
 row.names = c(NA, -61L), class = c("tbl_df",
                        "tbl", "data.frame"))



}else{
 content <- httr::content(response, "text")

 html_parsed <- rvest::read_html(response)

 tables = html_parsed %>%
  rvest::html_nodes('table') %>%
  rvest::html_table()



 if (length(tables) >= 10){
  # After the rbind
  # names(finacial_h_table) = c('Info', 'Detail')

  tables <- tables[sapply(tables, function(tbl) ncol(tbl) > 1)]

  ## Valuation Measures
  ### 9 rows
```

```
val_measure = tables[[1]]
val_measure = val_measure[, c(1,2)]
# names(val_measure) = c("Info", "Current")
names(val_measure) = c("X1", "X2")

# Financial Highlights tab
## Fiscal Year
### 2 rows
Fiscal_Year = tables[[2]]

## Profitability
### 2 rows
Profitability = tables[[3]]

## Management Effectiveness
### 2 rows
Management_Effectiveness = tables[[4]]

## Income Statement
### 8 rows
INC.stat = tables[[5]]

## Balance Sheet
### 6 rows
BS = tables[[6]]

## Cash Flow Statement
### 2 rows
CF = tables[[7]]
```

```
  ## Trading Information tab
  ## Stock Price History
  # Enlever les derniers chiffres en fin de texte
  ### 7 rows
  St_price_h = tables[[8]]

  # Share Statistics
  ### 12 rows
  Share_stats = tables[[9]]

  ## Dividends & Splits
  ### 10 rows
  Div_split = tables[[10]]

  df_init = as.data.frame(matrix(NA, ncol = 2, nrow = 1))
  names(df_init) = c('X1', 'X2')
  df_init$X1 = c("Ticker")
  df_init$X2 = ticker

  df = rbind(df_init,
          val_measure, Fiscal_Year, Profitability, Management_Effectiveness,
          INC.stat, BS, CF, St_price_h, Share_stats, Div_split)

  names(df) = c('Info', 'Detail')

 }else{

  df = structure(list(Info = c("Ticker", "Market Cap", "Enterprise Value",
```

```
                "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
                "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA",
                "Fiscal Year Ends", "Most Recent Quarter  (mrq)", "Profit Margin",
                "Operating Margin  (ttm)", "Return on Assets  (ttm)", "Return on Equity  (ttm)",
                "Revenue  (ttm)", "Revenue Per Share  (ttm)", "Quarterly Revenue Growth  (yoy)",
                "Gross Profit  (ttm)", "EBITDA", "Net Income Avi to Common  (ttm)",
                "Diluted EPS  (ttm)", "Quarterly Earnings Growth  (yoy)", "Total Cash  (mrq)",
                "Total Cash Per Share  (mrq)", "Total Debt  (mrq)", "Total Debt/Equity  (mrq)",
                "Current Ratio  (mrq)", "Book Value Per Share  (mrq)", "Operating Cash Flow  (ttm)",
                "Levered Free Cash Flow  (ttm)", "Beta (5Y Monthly)", "52 Week Range 3",
                "S&P 500 52-Week Change 3", "52 Week High 3", "52 Week Low 3",
                "50-Day Moving Average 3", "200-Day Moving Average 3", "Avg Vol (3 month) 3",
                "Avg Vol (10 day) 3", "Shares Outstanding 5", "Implied Shares Outstanding 6",
                "Float 8", "% Held by Insiders 1", "% Held by Institutions 1",
                "Shares Short (6/14/2024) 4", "Short Ratio (6/14/2024) 4", "Short % of Float
(6/14/2024) 4",
                "Short % of Shares Outstanding (6/14/2024) 4", "Shares Short (prior month
5/15/2024) 4",
                "Forward Annual Dividend Rate 4", "Forward Annual Dividend Yield 4",
                "Trailing Annual Dividend Rate 3", "Trailing Annual Dividend Yield 3",
                "5 Year Average Dividend Yield 4", "Payout Ratio 4", "Dividend Date 3",
                "Ex-Dividend Date 4", "Last Split Factor 2", "Last Split Date 3"
    ), Detail = c(ticker, rep('-',60))),
    row.names = c(NA, -61L), class = c("tbl_df",
                    "tbl", "data.frame"))

  }
 }
```

```
  ###

  if(full){

    if(!is.null(tables)){

      final_df = df

      return(final_df)

    }else{
      names(val_measure) = c('Info', 'Detail')

      return(df)
    }

  } else{

    return(df)
  }

}

for (i in 1:length(ticker)) {
  if(i == 1){

    final_data = get_keys_for_ticker(ticker[i], #full=
                                     full)
```

```
 }else{

  final_dt = get_keys_for_ticker(ticker[i], #full=
                          full)

  final_data = cbind(final_data, final_dt[, 'Detail'])

 }



}

# Rename final data
names(final_data)[2:length(final_data)] = final_data[1, 2:length(final_data)]
# remoove first row
final_data = final_data[-1,]

final_data = final_data %>%
 as_tibble()%>%
 column_to_rownames(var = "Info")

if (full) {
 return(final_data)
} else{
 if(length(ticker)==1){
  # df_val_measure = structure(list(Info = c("Ticker", "Market Cap", "Enterprise Value",
  #                                 "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
  #                                 "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA"),
  #                         Detail = c(ticker, final_data[1:9, ])),
```

```
#                 row.names = c(NA, -10L),
#                 class =  c("tbl_df","tbl", "data.frame"))


df = structure(list(Info = c(final_data[1:9, ], final_data[c(10:31), ], final_data[c(32:60), ])),
        row.names = c("Market Cap", "Enterprise Value",
                "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
                "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA",
                "Fiscal Year Ends", "Most Recent Quarter  (mrq)", "Profit Margin",
                "Operating Margin  (ttm)", "Return on Assets  (ttm)", "Return on Equity  (ttm)",
                "Revenue  (ttm)", "Revenue Per Share  (ttm)", "Quarterly Revenue Growth  (yoy)",
                "Gross Profit  (ttm)", "EBITDA", "Net Income Avi to Common  (ttm)",
                "Diluted EPS  (ttm)", "Quarterly Earnings Growth  (yoy)", "Total Cash  (mrq)",
                "Total Cash Per Share  (mrq)", "Total Debt  (mrq)", "Total Debt/Equity  (mrq)",
                "Current Ratio  (mrq)", "Book Value Per Share  (mrq)", "Operating Cash Flow  (ttm)",
                "Levered Free Cash Flow  (ttm)", "Beta (5Y Monthly)", "52 Week Range 3",
                "S&P 500 52-Week Change 3", "52 Week High 3", "52 Week Low 3",
                "50-Day Moving Average 3", "200-Day Moving Average 3", "Avg Vol (3 month) 3",
                "Avg Vol (10 day) 3", "Shares Outstanding 5", "Implied Shares Outstanding 6",
                "Float 8", "% Held by Insiders 1", "% Held by Institutions 1",
                "Shares Short (6/14/2024) 4", "Short Ratio (6/14/2024) 4", "Short % of Float (6/14/2024) 4",
                "Short % of Shares Outstanding (6/14/2024) 4", "Shares Short (prior month 5/15/2024) 4",
                "Forward Annual Dividend Rate 4", "Forward Annual Dividend Yield 4",
                "Trailing Annual Dividend Rate 3", "Trailing Annual Dividend Yield 3",
                "5 Year Average Dividend Yield 4", "Payout Ratio 4", "Dividend Date 3",
                "Ex-Dividend Date 4", "Last Split Factor 2", "Last Split Date 3"
        ), class = c("tbl_df","tbl", "data.frame"))
```

```
names(df) = ticker
dt1 = df[1:9, ]
dt1$Info = c("Market Cap", "Enterprise Value",
        "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
        "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA")

dt1 = dt1 %>%
 as_tibble()%>%
 column_to_rownames(var = "Info")

dt2 = df[c(10:31), ]
dt2$Info = c("Fiscal Year Ends", "Most Recent Quarter  (mrq)", "Profit Margin",
        "Operating Margin  (ttm)", "Return on Assets  (ttm)", "Return on Equity  (ttm)",
        "Revenue  (ttm)", "Revenue Per Share  (ttm)", "Quarterly Revenue Growth  (yoy)",
        "Gross Profit  (ttm)", "EBITDA", "Net Income Avi to Common  (ttm)",
        "Diluted EPS  (ttm)", "Quarterly Earnings Growth  (yoy)", "Total Cash  (mrq)",
        "Total Cash Per Share  (mrq)", "Total Debt  (mrq)", "Total Debt/Equity  (mrq)",
        "Current Ratio  (mrq)", "Book Value Per Share  (mrq)", "Operating Cash Flow  (ttm)",
        "Levered Free Cash Flow  (ttm)")

dt2 = dt2 %>%
 as_tibble()%>%
 column_to_rownames(var = "Info")

dt3 = df[c(32:60), ]

dt3$Info = c("Beta (5Y Monthly)", "52 Week Range 3",
        "S&P 500 52-Week Change 3", "52 Week High 3", "52 Week Low 3",
        "50-Day Moving Average 3", "200-Day Moving Average 3", "Avg Vol (3 month) 3",
```

```
            "Avg Vol (10 day) 3", "Shares Outstanding 5", "Implied Shares Outstanding 6",
            "Float 8", "% Held by Insiders 1", "% Held by Institutions 1",
            "Shares Short (6/14/2024) 4", "Short Ratio (6/14/2024) 4", "Short % of Float (6/14/2024) 4",
            "Short % of Shares Outstanding (6/14/2024) 4", "Shares Short (prior month 5/15/2024) 4",
            "Forward Annual Dividend Rate 4", "Forward Annual Dividend Yield 4",
            "Trailing Annual Dividend Rate 3", "Trailing Annual Dividend Yield 3",
            "5 Year Average Dividend Yield 4", "Payout Ratio 4", "Dividend Date 3",
            "Ex-Dividend Date 4", "Last Split Factor 2", "Last Split Date 3")

  dt3 = dt3 %>%
   as_tibble()%>%
   column_to_rownames(var = "Info")

  return(list(
   val_measure = dt1,
   f_Highlights = dt2,
   t_Information = dt3
  ))

 }else{
  return(list(
   val_measure = final_data[1:9, ],
   f_Highlights = final_data[c(10:31), ],
   t_Information = final_data[c(32:60), ]
  ))
 }

}
```

```
}


# Version améliorée yf_get_keys.stats
# yf_get_keys.stats <- function(ticker, full = TRUE) {
#
#   ticker = base::toupper(ticker)
#
#   if(all(ticker == "")){
#     # if(length(ticker) == 1  && ticker == ''){
#     return(NULL)
#   }
#
#           #    httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))
#
#   # Fonction pour obtenir les données ESG pour un seul ticker
#   get_keys_for_ticker <- function(ticker, full = TRUE) {
#
#     # ticker = toupper(ticker)
#
#               #    httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))
#
#     # url = glue::glue("https://finance.yahoo.com/quote/{ticker}/key-statistics/")
#     url = paste0("https://finance.yahoo.com/quote/", ticker, "/key-statistics/")
#
#     # url = paste0("https://finance.yahoo.com/quote/", ticker, "/key-statistics/")
#     # url <- "https://finance.yahoo.com/quote/KO/key-statistics/"
```

```
#
#   cookies = c(
#                     A1    =    "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-
Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
#     GUC = "AQABCAFm-EdnKEIeOgSl&s=AQAAAIJTL2AD&g=Zvb4qg",
#                     A3    =    "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-
Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
#                    A1S    =    "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-
Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
#     cmp = "t=1728944112&j=1&u=1---&v=48",
#     EuConsent = "CQFm5cAQFm5cAAOACKFRBLFgAAAAAAAAACiQAAAAAAAA",
#                                                                                  PRF
=
"t=ADP%2BKO%2BGIS%2BAAPL%2B%5EFVX%2BGNFT.PA%2BMSFT%2BA%2BGOOG%2BBA%2B
AL%2BRELIANCE.NS%2BVIIIX%2BFSKAX%2BFCNTX&newChartbetateaser=0%2C1719358224417
"
#   )
#
#   headers = c(
#                                                                                  accept
=
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.7",
#     `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",
#     `cache-control` = "no-cache",
#     pragma = "no-cache",
#     priority = "u=0, i",
#     `sec-ch-ua` = '"Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"',
#     `sec-ch-ua-mobile` = "?0",
#     `sec-ch-ua-platform` = '"Windows"',
#     `sec-fetch-dest` = "document",
#     `sec-fetch-mode` = "navigate",
#     `sec-fetch-site` = "none",
#     `sec-fetch-user` = "?1",
```

```
#     `upgrade-insecure-requests` = "1",
#     `user-agent` = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36"
#   )
#
#
#   # Envoyer la requête
#   # UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"
#
#  # response <- httr::GET(url, httr::add_headers(`Connection` = "keep-alive", `User-Agent` = UA))
#
#     response <- httr::GET(url, httr::add_headers(.headers=headers), httr::set_cookies(.cookies = cookies))
#
#   # Faire la requête GET avec les en-têtes spécifiés
#   # response <- httr::GET(
#   #  url,
#   #  add_headers(
#                                                 #                                                 `accept` = "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
#   #    `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",
#   #    `cache-control` = "no-cache",
#   #    `pragma` = "no-cache",
#   #    `priority` = "u=0, i",
#           #           `sec-ch-ua` = "\"Google Chrome\";v=\"125\", \"Chromium\";v=\"125\", \"Not.A/Brand\";v=\"24\"",
#   #    `sec-ch-ua-mobile` = "?0",
#   #    `sec-ch-ua-platform` = "\"Windows\"",
#   #    `sec-fetch-dest` = "document",
```

```
#   #    `sec-fetch-mode` = "navigate",
#   #    `sec-fetch-site` = "none",
#   #    `sec-fetch-user` = "?1",
#   #    `upgrade-insecure-requests` = "1"
#   #  )
#   # )
#
#   content <- httr::content(response, "text")
#
#   html_parsed <- rvest::read_html(response)
#
#   tables = html_parsed %>%
#    rvest::html_nodes('table') %>%
#    rvest::html_table()
#
#
#   if (base::length(tables) >= 10){
#    # After the rbind
#    # names(finacial_h_table) = c('Info', 'Detail')
#
#    tables <- tables[sapply(tables, function(tbl) ncol(tbl) > 1)]
#
#    ## Valuation Measures
#    ### 9 rows
#    val_measure = tables[[1]]
#    val_measure <- as.data.frame(val_measure)
#    val_measure = val_measure[, c(1,2)]
#    # names(val_measure) = c("Info", "Current")
#    names(val_measure) = c("X1", "X2")
```

```
#
#     # Financial Highlights tab
#     ## Fiscal Year
#     ### 2 rows
#     Fiscal_Year = tables[[2]]
#
#     ## Profitability
#     ### 2 rows
#     Profitability = tables[[3]]
#
#     ## Management Effectiveness
#     ### 2 rows
#     Management_Effectiveness = tables[[4]]
#
#     ## Income Statement
#     ### 8 rows
#     INC.stat = tables[[5]]
#
#     ## Balance Sheet
#     ### 6 rows
#     BS = tables[[6]]
#
#     ## Cash Flow Statement
#     ### 2 rows
#     CF = tables[[7]]
#
#     ## Trading Information tab
#     ## Stock Price History
#     # Enlever les derniers chiffres en fin de texte
```

```
#     ### 7 rows
#     St_price_h = tables[[8]]
#
#     # Share Statistics
#     ### 12 rows
#     Share_stats = tables[[9]]
#
#     ## Dividends & Splits
#     ### 10 rows
#     Div_split = tables[[10]]
#
#     # val_measure <- as.data.frame(val_measure)
#     # Fiscal_Year <- as.data.frame(Fiscal_Year)
#     # Profitability <- as.data.frame(Profitability)
#     # Management_Effectiveness <- as.data.frame(Management_Effectiveness)
#     # INC.stat <- as.data.frame(INC.stat)
#     # BS <- as.data.frame(BS)
#     # CF <- as.data.frame(CF)
#     # St_price_h <- as.data.frame(St_price_h)
#     # Share_stats <- as.data.frame(Share_stats)
#     # Div_split <- as.data.frame(Div_split)
#
#     df_init = as.data.frame(matrix(NA, ncol = 2, nrow = 1))
#     names(df_init) = c('X1', 'X2')
#     df_init$X1 = c("Ticker")
#     df_init$X2 = ticker
#
#     df = rbind(df_init,
#                val_measure, Fiscal_Year, Profitability, Management_Effectiveness,
```

```
#             INC.stat, BS, CF, St_price_h, Share_stats, Div_split)
#
#     # Fonction pour afficher des informations sur chaque dataframe
#     # debug_df_info <- function(df, name) {
#     #   cat("\n----\n")
#     #   cat("Dataframe:", name, "\n")
#     #   cat("Nombre de colonnes:", ncol(df), "\n")
#     #   cat("Noms des colonnes:", colnames(df), "\n")
#     #   cat("Premières lignes du dataframe:\n")
#     #   print(head(df))
#     #   cat("----\n")
#     # }
#
#     # Appliquer la fonction de debug sur chaque dataframe
#     # debug_df_info(df_init, "df_init")
#     # debug_df_info(val_measure, "val_measure")
#     # debug_df_info(Fiscal_Year, "Fiscal_Year")
#     # debug_df_info(Profitability, "Profitability")
#     # debug_df_info(Management_Effectiveness, "Management_Effectiveness")
#     # debug_df_info(INC.stat, "INC.stat")
#     # debug_df_info(BS, "BS")
#     # debug_df_info(CF, "CF")
#     # debug_df_info(St_price_h, "St_price_h")
#     # debug_df_info(Share_stats, "Share_stats")
#     # debug_df_info(Div_split, "Div_split")
#
#     # df = rbind(df_init[, c(1,2)],
#             #                 val_measure[, c(1,2)], Fiscal_Year[, c(1,2)], Profitability[, c(1,2)],
Management_Effectiveness[, c(1,2)],
```

```
#       #           INC.stat[, c(1,2)], BS[, c(1,2)], CF[, c(1,2)], St_price_h[, c(1,2)], Share_stats[, c(1,2)],
Div_split[, c(1,2)])
#
#     names(df) = c('Info', 'Detail')
#
#   }else{
#
#     df = structure(list(Info = c("Ticker", "Market Cap", "Enterprise Value",
#                         "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
#                         "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA",
#                         "Fiscal Year Ends", "Most Recent Quarter  (mrq)", "Profit Margin",
#                         "Operating Margin  (ttm)", "Return on Assets  (ttm)", "Return on Equity  (ttm)",
#                         "Revenue  (ttm)", "Revenue Per Share  (ttm)", "Quarterly Revenue Growth  (yoy)",
#                         "Gross Profit  (ttm)", "EBITDA", "Net Income Avi to Common  (ttm)",
#                         "Diluted EPS  (ttm)", "Quarterly Earnings Growth  (yoy)", "Total Cash  (mrq)",
#                         "Total Cash Per Share  (mrq)", "Total Debt  (mrq)", "Total Debt/Equity  (mrq)",
#                        "Current Ratio  (mrq)", "Book Value Per Share  (mrq)", "Operating Cash Flow  (ttm)",
#                         "Levered Free Cash Flow  (ttm)", "Beta (5Y Monthly)", "52 Week Range 3",
#                         "S&P 500 52-Week Change 3", "52 Week High 3", "52 Week Low 3",
#                         "50-Day Moving Average 3", "200-Day Moving Average 3", "Avg Vol (3 month) 3",
#                         "Avg Vol (10 day) 3", "Shares Outstanding 5", "Implied Shares Outstanding 6",
#                         "Float 8", "% Held by Insiders 1", "% Held by Institutions 1",
#                               "Shares Short (6/14/2024) 4", "Short Ratio (6/14/2024) 4", "Short % of Float
(6/14/2024) 4",
#                               "Short % of Shares Outstanding (6/14/2024) 4", "Shares Short (prior month
5/15/2024) 4",
#                         "Forward Annual Dividend Rate 4", "Forward Annual Dividend Yield 4",
#                         "Trailing Annual Dividend Rate 3", "Trailing Annual Dividend Yield 3",
#                         "5 Year Average Dividend Yield 4", "Payout Ratio 4", "Dividend Date 3",
#                         "Ex-Dividend Date 4", "Last Split Factor 2", "Last Split Date 3"
```

```
#     ), Detail = c(ticker, rep('-',60))),
#     row.names = c(NA, -61L), class = c("tbl_df",
#                                        "tbl", "data.frame"))
#
#   }
#
#   if(full){
#
#     if(!is.null(tables)){
#
#       final_df = df
#
#       return(final_df)
#
#     }else{
#       names(val_measure) = c('Info', 'Detail')
#
#       # return(df)
#       return(val_measure)
#     }
#
#   } else{
#
#     return(df)
#   }
#
# }
#
#  for (i in 1:length(ticker)) {
```

```
#    if(i == 1){
#
#      final_data = get_keys_for_ticker(ticker[i], #full=
#                                       full)
#
#    }else{
#
#      final_dt = get_keys_for_ticker(ticker[i], #full=
#                                     full)
#
#      final_data = cbind(final_data, final_dt[, 'Detail'])
#
#    }
#
#
#  }
#
#  # Rename final data
#  names(final_data)[2:length(final_data)] = final_data[1, 2:length(final_data)]
#  # remoove first row
#  final_data = final_data[-1,]
#
#  final_data = final_data %>%
#    as_tibble()%>%
#    column_to_rownames(var = "Info")
#
#  if (full) {
#    return(final_data)
#  } else{
```

```
#   if(length(ticker)==1){
#    # df_val_measure = structure(list(Info = c("Ticker", "Market Cap", "Enterprise Value",
#    #                                          "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
#    #                                          "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA"),
#    #                                Detail = c(ticker, final_data[1:9, ])),
#    #                           row.names = c(NA, -10L),
#    #                           class =  c("tbl_df","tbl", "data.frame"))
#
#    df = structure(list(Info = c(final_data[1:9, ], final_data[c(10:31), ], final_data[c(32:60), ])),
#    row.names = c("Market Cap", "Enterprise Value",
#             "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
#             "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA",
#             "Fiscal Year Ends", "Most Recent Quarter  (mrq)", "Profit Margin",
#             "Operating Margin  (ttm)", "Return on Assets  (ttm)", "Return on Equity  (ttm)",
#             "Revenue  (ttm)", "Revenue Per Share  (ttm)", "Quarterly Revenue Growth  (yoy)",
#             "Gross Profit  (ttm)", "EBITDA", "Net Income Avi to Common  (ttm)",
#             "Diluted EPS  (ttm)", "Quarterly Earnings Growth  (yoy)", "Total Cash  (mrq)",
#             "Total Cash Per Share  (mrq)", "Total Debt  (mrq)", "Total Debt/Equity  (mrq)",
#             "Current Ratio  (mrq)", "Book Value Per Share  (mrq)", "Operating Cash Flow  (ttm)",
#             "Levered Free Cash Flow  (ttm)", "Beta (5Y Monthly)", "52 Week Range 3",
#             "S&P 500 52-Week Change 3", "52 Week High 3", "52 Week Low 3",
#             "50-Day Moving Average 3", "200-Day Moving Average 3", "Avg Vol (3 month) 3",
#             "Avg Vol (10 day) 3", "Shares Outstanding 5", "Implied Shares Outstanding 6",
#             "Float 8", "% Held by Insiders 1", "% Held by Institutions 1",
#              "Shares Short (6/14/2024) 4", "Short Ratio (6/14/2024) 4", "Short % of Float (6/14/2024)
4",
#            "Short % of Shares Outstanding (6/14/2024) 4", "Shares Short (prior month 5/15/2024) 4",
#             "Forward Annual Dividend Rate 4", "Forward Annual Dividend Yield 4",
#             "Trailing Annual Dividend Rate 3", "Trailing Annual Dividend Yield 3",
```

```
#             "5 Year Average Dividend Yield 4", "Payout Ratio 4", "Dividend Date 3",
#             "Ex-Dividend Date 4", "Last Split Factor 2", "Last Split Date 3"
#   ), class = c("tbl_df","tbl", "data.frame"))
#
#   names(df) = ticker
#   dt1 = df[1:9, ]
#   dt1$Info = c("Market Cap", "Enterprise Value",
#                "Trailing P/E", "Forward P/E", "PEG Ratio (5yr expected)", "Price/Sales",
#                "Price/Book", "Enterprise Value/Revenue", "Enterprise Value/EBITDA")
#
#   dt1 = dt1 %>%
#    as_tibble()%>%
#    column_to_rownames(var = "Info")
#
#   dt2 = df[c(10:31), ]
#   dt2$Info = c("Fiscal Year Ends", "Most Recent Quarter  (mrq)", "Profit Margin",
#                "Operating Margin  (ttm)", "Return on Assets  (ttm)", "Return on Equity  (ttm)",
#                "Revenue  (ttm)", "Revenue Per Share  (ttm)", "Quarterly Revenue Growth  (yoy)",
#                "Gross Profit  (ttm)", "EBITDA", "Net Income Avi to Common  (ttm)",
#                "Diluted EPS  (ttm)", "Quarterly Earnings Growth  (yoy)", "Total Cash  (mrq)",
#                "Total Cash Per Share  (mrq)", "Total Debt  (mrq)", "Total Debt/Equity  (mrq)",
#                "Current Ratio  (mrq)", "Book Value Per Share  (mrq)", "Operating Cash Flow  (ttm)",
#                "Levered Free Cash Flow  (ttm)")
#
#   dt2 = dt2 %>%
#    as_tibble()%>%
#    column_to_rownames(var = "Info")
#
#   dt3 = df[c(32:60), ]
```

```
#
#     dt3$Info = c("Beta (5Y Monthly)", "52 Week Range 3",
#                  "S&P 500 52-Week Change 3", "52 Week High 3", "52 Week Low 3",
#                  "50-Day Moving Average 3", "200-Day Moving Average 3", "Avg Vol (3 month) 3",
#                  "Avg Vol (10 day) 3", "Shares Outstanding 5", "Implied Shares Outstanding 6",
#                  "Float 8", "% Held by Insiders 1", "% Held by Institutions 1",
#                 "Shares Short (6/14/2024) 4", "Short Ratio (6/14/2024) 4", "Short % of Float (6/14/2024)
4",
#                  "Short % of Shares Outstanding (6/14/2024) 4", "Shares Short (prior month 5/15/2024)
4",
#                  "Forward Annual Dividend Rate 4", "Forward Annual Dividend Yield 4",
#                  "Trailing Annual Dividend Rate 3", "Trailing Annual Dividend Yield 3",
#                  "5 Year Average Dividend Yield 4", "Payout Ratio 4", "Dividend Date 3",
#                  "Ex-Dividend Date 4", "Last Split Factor 2", "Last Split Date 3")
#
#     dt3 = dt3 %>%
#       as_tibble()%>%
#       column_to_rownames(var = "Info")
#
#     return(list(
#       val_measure = dt1,
#       f_Highlights = dt2,
#       t_Information = dt3
#     ))
#
#   }else{
#     return(list(
#       val_measure = final_data[1:9, ],
#       f_Highlights = final_data[c(10:31), ],
```

```
#       t_Information = final_data[c(32:60), ]
#     ))
#   }
#
#  }
#
#
# }



# yf_get_keys.stats <- function(ticker, full = TRUE) {
#   ticker <- base::toupper(ticker)
#
#   if (all(ticker == "")) {
#     return(NULL)
#   }
#
#   # Fonction pour obtenir les données ESG pour un seul ticker
#   get_keys_for_ticker <- function(ticker, full = TRUE) {
#     url <- paste0("https://finance.yahoo.com/quote/", ticker, "/key-statistics/")
#
#     cookies <- c(
#                         A1    =    "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-
Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
#       GUC = "AQABCAFm-EdnKEIeOgSl&s=AQAAAIJTL2AD&g=Zvb4qg",
#                         A3    =    "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-
Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
#                       A1S    =    "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-
Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
#       cmp = "t=1728944112&j=1&u=1---&v=48",
```

```
#      EuConsent = "CQFm5cAQFm5cAAOACKFRBLFgAAAAAAAAACiQAAAAAAAA",
#                                                    PRF          =
"t=ADP%2BKO%2BGIS%2BAAPL%2B%5EFVX%2BGNFT.PA%2BMSFT%2BA%2BGOOG%2BBA%2B
AL%2BRELIANCE.NS%2BVIIIX%2BFSKAX%2BFCNTX&newChartbetateaser=0%2C1719358224417
"
#    )
#
#    headers <- c(
#                                                    accept          =
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.7",
#      `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",
#      `cache-control` = "no-cache",
#      pragma = "no-cache",
#      `sec-ch-ua` = '"Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"',
#      `sec-ch-ua-mobile` = "?0",
#      `sec-ch-ua-platform` = '"Windows"',
#      `sec-fetch-dest` = "document",
#      `sec-fetch-mode` = "navigate",
#      `sec-fetch-site` = "none",
#      `sec-fetch-user` = "?1",
#      `upgrade-insecure-requests` = "1",
#      `user-agent` = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/129.0.0.0 Safari/537.36"
#    )
#
#    response <- httr::GET(url, httr::add_headers(.headers = headers), httr::set_cookies(.cookies =
cookies))
#    content <- httr::content(response, "text")
#
#    html_parsed <- rvest::read_html(response)
```

```
#    tables <- html_parsed %>% rvest::html_nodes('table') %>% rvest::html_table()
#
#    if (length(tables) == 10) {
#      val_measure <- tables[[1]][, c(1, 2)]
#      names(val_measure) <- c("X1", "X2")
#      Fiscal_Year <- tables[[2]]
#      Profitability <- tables[[3]]
#      Management_Effectiveness <- tables[[4]]
#      INC_stat <- tables[[5]]
#      BS <- tables[[6]]
#      CF <- tables[[7]]
#      St_price_h <- tables[[8]]
#      Share_stats <- tables[[9]]
#      Div_split <- tables[[10]]
#
#      df_init <- data.frame(X1 = "Ticker", X2 = ticker, stringsAsFactors = FALSE)
#     df <- rbind(df_init, val_measure, Fiscal_Year, Profitability, Management_Effectiveness, INC_stat,
BS, CF, St_price_h, Share_stats, Div_split)
#      names(df) <- c('Info', 'Detail')
#    } else {
#      df <- structure(list(
#        Info = c("Ticker", "Market Cap", "Enterprise Value", "Trailing P/E", "Forward P/E",
#                 "PEG Ratio (5yr expected)", "Price/Sales", "Price/Book", "Enterprise Value/Revenue",
#                 "Enterprise Value/EBITDA", "Fiscal Year Ends", "Most Recent Quarter  (mrq)",
#                 "Profit Margin", "Operating Margin  (ttm)", "Return on Assets  (ttm)",
#                 "Return on Equity  (ttm)", "Revenue  (ttm)", "Revenue Per Share  (ttm)",
#                 "Quarterly Revenue Growth  (yoy)", "Gross Profit  (ttm)", "EBITDA",
#                "Net Income Avi to Common  (ttm)", "Diluted EPS  (ttm)", "Quarterly Earnings Growth  (yoy)",
#                 "Total Cash  (mrq)", "Total Cash Per Share  (mrq)", "Total Debt  (mrq)",
```

```
#             "Total Debt/Equity  (mrq)", "Current Ratio  (mrq)", "Book Value Per Share  (mrq)",
#             "Operating Cash Flow  (ttm)", "Levered Free Cash Flow  (ttm)", "Beta (5Y Monthly)",
#             "52 Week Range", "S&P 500 52-Week Change", "52 Week High", "52 Week Low",
#             "50-Day Moving Average", "200-Day Moving Average", "Avg Vol (3 month)",
#             "Avg Vol (10 day)", "Shares Outstanding", "Implied Shares Outstanding",
#             "Float", "% Held by Insiders", "% Held by Institutions", "Shares Short",
#             "Short Ratio", "Short % of Float", "Short % of Shares Outstanding",
#             "Shares Short (prior month)", "Forward Annual Dividend Rate",
#             "Forward Annual Dividend Yield", "Trailing Annual Dividend Rate",
#             "Trailing Annual Dividend Yield", "5 Year Average Dividend Yield",
#             "Payout Ratio", "Dividend Date", "Ex-Dividend Date", "Last Split Factor",
#             "Last Split Date"),
#       Detail = c(ticker, rep("-", 60))
#     ), class = c("tbl_df", "tbl", "data.frame"))
#   }
#
#   if (full) {
#     return(df)
#   } else {
#     return(df[1:10, ])
#   }
# }
#
# final_data <- purrr::map_dfc(ticker, ~ get_keys_for_ticker(.x, full = full))
#
# final_data <- final_data %>%
#   dplyr::as_tibble() %>%
#   tibble::column_to_rownames(var = "Info")
#
```

```
#   return(final_data)
# }




# yf_get_keys.stats(c("PRU", "AAPL", "RTX"))
# df = yf_get_keys.stats(c("PRU", "AAPL", "RTX"))

# Version améliorée yf_get_keys.stats
yf_get_summary <- function(ticker) {

 if(all(ticker == "")){
  # if(length(ticker) == 1  && ticker == ''){
  return(NULL)
 }

 #      httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))

 # Fonction pour obtenir les données ESG pour un seul ticker
 get_summary_for_ticker <- function(ticker) {
  #      httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))

  # url = glue::glue("https://finance.yahoo.com/quote/{ticker}/")
  url = paste0("https://finance.yahoo.com/quote/", ticker, "/")
  # url <- "https://finance.yahoo.com/quote/KO/"

  # Envoyer la requête
```

```
UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"
response <- httr::GET(url, add_headers(`Connection` = "keep-alive", `User-Agent` = UA))

# Faire la requête GET avec les en-têtes spécifiés
# response <- httr::GET(
#   url,
#   add_headers(
#     `accept` = "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
#     `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",
#     `cache-control` = "no-cache",
#     `pragma` = "no-cache",
#     `priority` = "u=0, i",
#     `sec-ch-ua` = "\"Google Chrome\";v=\"125\", \"Chromium\";v=\"125\", \"Not.A/Brand\";v=\"24\"",
#     `sec-ch-ua-mobile` = "?0",
#     `sec-ch-ua-platform` = "\"Windows\"",
#     `sec-fetch-dest` = "document",
#     `sec-fetch-mode` = "navigate",
#     `sec-fetch-site` = "none",
#     `sec-fetch-user` = "?1",
#     `upgrade-insecure-requests` = "1"
#   )
# )

content <- httr::content(response, "text")

html_parsed <- rvest::read_html(response)
```

```
label = html_parsed  %>% rvest::html_nodes(".label") %>% rvest::html_text(trim = TRUE)
label = label[1:16]
value = html_parsed  %>% rvest::html_nodes(".value") %>% rvest::html_text(trim = TRUE)
value = value[1:16]

# Créer le dataframe
df <- data.frame(Info = label, Value = value)

# tables = html_parsed %>%
#   rvest::rvest::html_nodes('table') %>%
#   rvest::html_table()



if (!is.null(df)){

  names(df)[2] = ticker

}else{

  df = structure(list(Info = c("Previous Close", "Open", "Bid", "Ask",
                               "Day's Range", "52 Week Range", "Volume", "Avg. Volume", "Market Cap (intraday)",
                               "Beta (5Y Monthly)", "PE Ratio (TTM)", "EPS (TTM)", "Earnings Date",
                               "Forward Dividend & Yield", "Ex-Dividend Date", "1y Target Est"
  ), Value = rep("-", 16)), class = "data.frame", row.names = c(NA,
                                                                -16L))
  names(df)[2] = ticker
}

return(df)
```

```
}

for (i in 1:length(ticker)) {
  if(i == 1){

    final_data = get_summary_for_ticker(ticker[i])

  }else{

    final_dt = get_summary_for_ticker(ticker[i])



    tick = ticker[i]
    final_data = cbind(final_data, final_dt[, tick])
    names(final_data)[i+1] = tick
    # print(tick)

  }



}

final_data = final_data %>%
  as_tibble()%>%
  column_to_rownames(var = "Info")

return(final_data)
```

```
}

# yf_get_summary(c("PRU", "AAPL"))

# yf_get_summary("PRU")

# Fonction pour obtenir les noms complets des tickers
yf.get_long_names <- function(tickers) {
 # Initialiser une liste pour stocker les résultats
 results <- list()

 cookies = c(
  GUC = "AQABCAFnLLFnXEIeOgSl&s=AQAAAD8j2FLC&g=ZytikA",
  EuConsent = "CQFm5cAQFm5cAAOACKFRBOFgAAAAAAAAACiQAAAAAAAA",
  A1 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAGxLGdcZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAq5W1hyNRjdnqqQ3_Y1KWsg",
  A3 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAGxLGdcZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAq5W1hyNRjdnqqQ3_Y1KWsg",
  A1S = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAGxLGdcZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAq5W1hyNRjdnqqQ3_Y1KWsg",
  cmp = "t=1730896521&j=1&u=1---&v=51",
  PRF = "t=GIS%2BPRU%2BETGLX%2BKO%2BMETA%2B0P000023MW.L%2BCPRI%2BAAA%2B0P0000XPCJ.F%2B0P0001FG1E.F%2BMAMI.F%2B8AECQ.MI%2BZPDX.MU%2BZPDX.BE%2BZPDX.DE&newChartbetateaser=0%2C1719358224417"
 )

 for (ticker in tickers) {
  url <- paste0("https://fr.finance.yahoo.com/quote/", ticker, "/")
```

```
  UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"

  res_ <- httr::GET(url,
            httr::set_cookies(.cookies = cookies),
            httr::add_headers(`Connection` = "keep-alive", `User-Agent` = UA))

  # Extraire les documents Excel
  html_content <- rvest::read_html(res_)

  # long_name <- html_content %>%
  #   rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[1]/div[1]/div/section/h1') %>%
  #   rvest::html_text(trim = TRUE)

  # Essayer de lire la page HTML et d'extraire le nom complet
  long_name <- tryCatch({
    # read_html(url) %>%
    #   rvest::html_nodes(xpath = '//*[@id="quote-header-info"]/div[2]/div[1]/div[1]/h1') %>%
    #   rvest::html_text(trim = TRUE)

    html_content %>%
      rvest::html_nodes(xpath = '//*[@id="nimbus-app"]/section/section/section/article/section[1]/div[1]/div/div/section/h1') %>%
      rvest::html_text(trim = TRUE)

  }, error = function(e) {
    NA  # Retourner NA en cas d'erreur
  })
```

```
    # Si le nom est valide, ajouter à la liste des résultats
    if (!is.na(long_name) && length(long_name) > 0) {
      # Supprimer tout ce qui est après la première parenthèse
      clean_name <- sub("\\s*\\(.*\\)", "", long_name)

      # clean_name <- gsub("\\s*\\([^\\)]+\\)", "", long_name)

      results[[ticker]] <- clean_name
    }
  }



  if(length(results)==1){
    return(clean_name)

  }else{
    # Convertir la liste en data.frame
    result_df <- data.frame(
      Ticker = names(results),
      `Long name` = unlist(results),
      stringsAsFactors = FALSE
    )
    return(result_df)
  }
}

# Exemple d'utilisation
# yf.get_long_names("KO")
```

```
#
# tickers <- c("AAPL", "GOOGL", "MSFT", "INVALID")
# long_names_df <- yf.get_long_names(tickers)
# print(long_names_df)



# Version améliorée de la fonction
yf_isin_to_ticker <- function(isins, typeDisp = "Equity") {
 # Initialize an empty list to store the results
 isins = toupper(isins)

 if(all(isins == "")){
  # if(length(ticker) == 1  && ticker == ''){
  return(NULL)
 }

 results <- list()
 ticker_list = NULL

 # Initialize empty lists for results and valid ISINs
 valid_isins <- character(0)

 # Iterate over each ISIN
 for (isin in isins) {
  # Check ISIN length and format (12 characters, alpha-numeric)
  if (nchar(isin) == 12 && grepl("^[A-Z]{2}[0-9]{10}$", isin)) {
   valid_isins <- c(valid_isins, isin)
  }
 }
```

```
 # Iterate over each ISIN
 for (isin in valid_isins) {
  # Define the URL
  url <- paste0("https://query2.finance.yahoo.com/v1/finance/search?q=", isin, "&lang=en-US&region=US&quotesCount=6&newsCount=3&listsCount=2&enableFuzzyQuery=false&quotesQueryId=tss_match_phrase_query&multiQuoteQueryId=multi_quote_single_token_query&newsQueryId=news_cie_vespa&enableCb=true&enableNavLinks=true&enableEnhancedTrivialQuery=true&enableResearchReports=true&enableCulturalAssets=true&enableLogoUrl=true&uiConfig=[object%20Object]&searchConfig=[object%20Object]&recommendCount=5")

  # Define the headers
  # headers <- c(
  #   "accept" = "*/*",
  #   "accept-language" = "fr-SN,fr;q=0.9,ee-TG;q=0.8,ee;q=0.7,fr-FR;q=0.6,en-US;q=0.5,en;q=0.4",
  #   "cache-control" = "no-cache",
  #   "pragma" = "no-cache",
  #   "priority" = "u=1, i",
  #  "sec-ch-ua" = "\"Not/A)Brand\";v=\"8\", \"Chromium\";v=\"126\", \"Google Chrome\";v=\"126\"",
  #   "sec-ch-ua-mobile" = "?0",
  #   "sec-ch-ua-platform" = "\"Windows\"",
  #   "sec-fetch-dest" = "empty",
  #   "sec-fetch-mode" = "cors",
  #   "sec-fetch-site" = "same-site"
  # )

  # Envoyer la requête
  UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"

  # Make the GET request
```

```
  # response <- httr::GET(url, add_headers(headers))
  response <- httr::GET(url, httr::add_headers(`Connection` = "keep-alive", `User-Agent` = UA))

  # Parse the JSON response
  data <- jsonlite::fromJSON(content(response, "text"))

  if ("quotes" %in% names(data) && length(data$quotes) != 0) {
   data <- data$quotes

   # data <- data[data$typeDisp == typeDisp,]

   #print(quotes)
   #for (quote in quotes) {
   # Extract relevant information (e.g., symbol, longname, exchange)
   data = data[, c("symbol", "longname")]

   # print(data)

   ticker <- data$symbol
   longname <- data$longname
   results[[length(results) + 1]] <- list(isin = isin,
                                          ticker = ticker,
                                          longname = longname
   )

   ticker_list = append(ticker_list, ticker)
  }
 }
```

```
 if(length(results) >= 1){
   # df = dplyr::bind_rows(test$results)
   df = dplyr::bind_rows(results)
   # Return the results
   return(list(df = df,
               results = results,
               tick = ticker_list))
 }


}


# test = yf_isin_to_ticker(c("US0378331005", "FO12", "US0231351067", "US5949181045"))
#
# test1 = yf_isin_to_ticker(c("US0378331005", "US0231351067", "US5949181045"))
#
# yf_isin_to_ticker(c("US0378331075")) #will return null bcse ISIN does not exist
#
# test$results #list
# test$df #dataframe
# test$df$ticker #ticker from dataframe
# test$tick #ticker
#
# test1$results #list
# test1$df #dataframe
# test1$df$ticker #ticker from dataframe
# test1$tick #ticker


################################################################################
####
```

Exemple

Pour tester avec les isin

Saisir directement ISIN

"NL0014559478"

CAC_next_20 = c("FR0000184798", "FR0010411983", "FR0006174348", "FR0013280286", "FR0013154002", "FR0000054470", "BE0003470755", "FR0010313833", "FR0000130452", "GB00BDSFG982", "FR0000120404", "FR0010040865", "FR0010613471", "FR0010242511", "FR0014000MR3", "FR0000121964", "FR0010908533", "FR0013176526", "FR0010533075", "FR0000121220")

CAC_40 = c("FR0000121667", "FR0011981968", "NL00150001Q9", "FR0000125338", "FR0000131906", "FR0000130809", "FR0000051807", "FR0000121329", "FR0000124141", "FR0010208488", "FR0000130650", "FR0000125486", "FR0000052292", "FR0010307819", "FR0000133308", "FR0000125007", "FR0000121972", "FR0000120172", "FR0013326246", "NL0000235190", "FR0000131104", "FR0000051732", "FR0000121014", "FR0000045072", "FR0000130577", "FR0010220475", "FR0000120628", "FR0000121485", "NL0000226223", "LU1598757687", "FR0000127771", "FR0000120073", "FR0000120503", "FR0000120321", "FR0000120578", "FR0000121261", "FR0000073272", "FR0000120644", "FR0000120693", "FR0000120271")

SBF_120 = c("FR0000121667", "FR0013181864", "NL00150001Q9", "FR0000131906", "FR0006174348", "FR0010340141", "FR0012435121", "FR0000130809", "FR0010241638", "FR0011742329", "FR0000121329", "FR0010208488", "FR0000130650", "FR0000125486", "FR0011675362", "FR0000053225", "FR0010221234", "FR0000125585", "LU0088087324", "FR0010307819", "FR0000133308", "FR0000121972", "GB00BDSFG982", "FR0000120404", "FR0010613471", "FR0000035081", "LU0569974404", "FR0000121964", "FR0000121147", "FR0000077919", "FR0000050809", "FR0010533075", "FR0000121220", "FR0000045072", "FR0000184798", "FR0010220475", "FR0004125920", "FR0000064578", "FR0013280286", "FR0000039091", "FR0000120966", "FR0011950732", "LU1598757687", "FR0011726835", "FR0000039299", "FR0011476928", "FR0010667147", "FR0000120073", "FR0004007813", "FR0010313833", "BE0003470755", "FR0000054470", "FR0013154002", "FR0000120321", "FR0000120503", "NL0014559478", "FR0000121204", "FR0000121261", "FR0013506730", "FR0000060402", "FR0010242511", "FR0000120693", "FR0000044448", "FR0013269123", "FR0000130395", "FR0000073298", "FR0000071946", "FR0010828137", "FR0011981968", "FR0000125338", "FR0010411983", "FR0000051807", "FR0000124141", "FR0000121709", "NL0006294274", "FR0000131757", "FR0000052292", "FR0000125007", "FR0000124570", "FR0010386334", "FR0010040865", "FR0000054900", "FR0012757854", "FR0014000MR3", "FR0000120172", "FR0013447729", "FR0013326246", "FR0000130213", "FR0010112524", "FR0000051732", "NL0000235190", "FR0005691656", "FR0000131104", "FR0000121014", "FR0000120222", "FR0000120685", "FR0000130577", "FR0000121485", "FR0000120628", "FR0000120859", "NL0000226223", "FR0000127771", "FR0000130452", "FR0000031122",

```
"FR0000120578", "FR0013379484", "FR0000031577", "FR0000073272", "FR0013227113",
"FR0013451333", "FR0010259150", "FR0004035913", "FR0013258662", "FR0000121725",
"FR0010451203", "FR0000121121", "FR0000120644", "FR0013153541", "FR0000120271",
"FR0010908533", "FR0013176526")

#

# test_cac20 = yf_isin_to_ticker(CAC_next_20)

# test_cac40 = yf_isin_to_ticker(CAC_40)

# # test_sbf120 = yf_isin_to_ticker(SBF_120)

# # Liste des ISIN dans isin_list qui ne sont pas présents dans test$df$isin

# missing_isin_cac20 <- setdiff(CAC_next_20, test_cac20$df$isin)

# # missing_isin_cac20

# # [1] "GB00BDSFG982" "FR0010613471" "FR0010242511" "FR0014000MR3"

# missing_isin_cac40 <- setdiff(CAC_40, test_cac40$df$isin)

# # > missing_isin_cac40

# # [1] "NL00150001Q9"

#

# test = yf_isin_to_ticker(CAC_next_20)

# test = yf_isin_to_ticker(CAC_next_20)

# # Ou

# # tick_not_present = NULL

# # for(tick in isin_list){

# #   if(!(tick %in% (test$df$isin)))

# #     tick_not_present = append(tick_not_present, tick)

# # }



# CAC_40

# FR0000121667, FR0011981968, NL00150001Q9, FR0000125338, FR0000131906, FR0000130809,
FR0000051807, FR0000121329, FR0000124141, FR0010208488, FR0000130650, FR0000125486,
FR0000052292, FR0010307819, FR0000133308, FR0000125007, FR0000121972, FR0000120172,
FR0013326246, NL0000235190, FR0000131104, FR0000051732, FR0000121014, FR0000045072,
```

FR0000130577, FR0010220475, FR0000120628, FR0000121485, NL0000226223, LU1598757687, FR0000127771, FR0000120073, FR0000120503, FR0000120321, FR0000120578, FR0000121261, FR0000073272, FR0000120644, FR0000120693, FR0000120271

CAC_next_19

FR0000184798, FR0010411983, FR0006174348, FR0013280286, FR0013154002, FR0000054470, BE0003470755, FR0010313833, FR0000130452, GB00BDSFG982, FR0000120404, FR0010040865, FR0010613471, FR0010242511, FR0014000MR3, FR0000121964, FR0010908533, FR0013176526, FR0010533075, FR0000121220

FR0000184798, FR0010411983, FR0006174348, FR0013280286, FR0013154002, FR0000054470

CAC_next_20 mais le dernier isin appartient à deux ticker

FR0000184798, FR0010411983, FR0006174348, FR0013280286, FR0013154002, FR0000054470, BE0003470755, FR0010313833, FR0000130452, GB00BDSFG982, FR0000120404, FR0010040865, FR0010613471, FR0010242511, FR0014000MR3, FR0000121964, FR0010908533, FR0013176526, FR0010533075, FR0000121220 #, NL0014559478

CAC_10 pour le test

FR0000184798, FR0010411983, FR0000120404, FR0010040865, FR0000121964,

FR0010908533, FR0013176526, FR0010533075, FR0000121220, FR0006174348

le 7ième FR0010908533

double ticker

FR0014000MR3,

FR0010613471, FR0010242511 n'existe pas dans yahoo

sbf_120

FR0000121667, FR0013181864, NL00150001Q9, FR0000131906, FR0006174348, FR0010340141, FR0012435121, FR0000130809, FR0010241638, FR0011742329, FR0000121329, FR0010208488, FR0000130650, FR0000125486, FR0011675362, FR0000053225, FR0010221234, FR0000125585, LU0088087324, FR0010307819, FR0000133308, FR0000121972, GB00BDSFG982, FR0000120404,

FR0010613471, FR0000035081, LU0569974404, FR0000121964, FR0000121147, FR0000077919, FR0000050809, FR0010533075, FR0000121220, FR0000045072, FR0000184798, FR0010220475, FR0004125920, FR0000064578, FR0013280286, FR0000039091, FR0000120966, FR0011950732, LU1598757687, FR0011726835, FR0000039299, FR0011476928, FR0010667147, FR0000120073, FR0004007813, FR0010313833, BE0003470755, FR0000054470, FR0013154002, FR0000120321, FR0000120503, NL0014559478, FR0000121204, FR0000121261, FR0013506730, FR0000060402, FR0010242511, FR0000120693, FR0000044448, FR0013269123, FR0000130395, FR0000073298, FR0000071946, FR0010828137, FR0011981968, FR0000125338, FR0010411983, FR0000051807, FR0000124141, FR0000121709, NL0006294274, FR0000131757, FR0000052292, FR0000125007, FR0000124570, FR0010386334, FR0010040865, FR0000054900, FR0012757854, FR0014000MR3, FR0000120172, FR0013447729, FR0013326246, FR0000130213, FR0010112524, FR0000051732, NL0000235190, FR0005691656, FR0000131104, FR0000121014, FR0000120222, FR0000120685, FR0000130577, FR0000121485, FR0000120628, FR0000120859, NL0000226223, FR0000127771, FR0000130452, FR0000031122, FR0000120578, FR0013379484, FR0000031577, FR0000073272, FR0013227113, FR0013451333, FR0010259150, FR0004035913, FR0013258662, FR0000121725, FR0010451203, FR0000121121, FR0000120644, FR0013153541, FR0000120271, FR0010908533, FR0013176526

FR0000120404, FR0010340141, FR0000031122, FR0000120073,

NL0000235190, FR0000060402, FR0013258662, FR0010220475,

FR0000071946, FR0004125920, LU0569974404, LU1598757687,

FR0010313833, FR0000051732, FR0000120628, FR0000120966,

FR0013280286, FR0000131104, FR0000039299, FR0000120503,

FR0006174348

Saisir directement nom des ticker

KO, AAPL, ADP, AC.PA, ADP.PA, AF.PA, AI.PA, AIR.PA

get_keys_for_ticker(c("AC.PA", "ADP.PA", "AF.PA", "AI.PA", "AIR.PA"))

##

```
# Define the function to retreive ticker from ISIN

# yf_isin_to_ticker <- function(isin) {

#

#   if(all(ticker == "")){

#     # if(length(ticker) == 1  && ticker == ''){

#     return(NULL)

#   }

#

#   # Define the URL

#       url <- paste0("https://query2.finance.yahoo.com/v1/finance/search?q=", isin, "&lang=en-US&region=US&quotesCount=6&newsCount=3&listsCount=2&enableFuzzyQuery=false&quotesQueryId=tss_match_phrase_query&multiQuoteQueryId=multi_quote_single_token_query&newsQueryId=news_cie_vespa&enableCb=true&enableNavLinks=true&enableEnhancedTrivialQuery=true&enableResearchReports=true&enableCulturalAssets=true&enableLogoUrl=true&uiConfig=[object%20Object]&searchConfig=[object%20Object]&recommendCount=5")

#

#   # Define the headers

#   headers <- c(

#     "accept" = "*/*",

#     "accept-language" = "fr-SN,fr;q=0.9,ee-TG;q=0.8,ee;q=0.7,fr-FR;q=0.6,en-US;q=0.5,en;q=0.4",

#     "cache-control" = "no-cache",

#     "pragma" = "no-cache",

#     "priority" = "u=1, i",

#     "sec-ch-ua" = "\"Not/A)Brand\";v=\"8\", \"Chromium\";v=\"126\", \"Google Chrome\";v=\"126\"",

#     "sec-ch-ua-mobile" = "?0",

#     "sec-ch-ua-platform" = "\"Windows\"",

#     "sec-fetch-dest" = "empty",

#     "sec-fetch-mode" = "cors",

#     "sec-fetch-site" = "same-site"

#   )
```

```
#
#   # Make the GET request
#   response <- httr::GET(url, add_headers(headers))
#
#   # Parse the JSON response
#   data <- jsonlite::fromJSON(content(response, "text"))
#
#   if("quotes" %in%names(data)){
#     data = data$quotes
#
#     # data = data[, c("symbol", "longname", "exchange", "exchDisp",
#     #                 "sectorDisp",  "industryDisp", "index", "quoteType",
#     #                 "typeDisp", "score", "logoUrl"
#     #                 # Utiliser cette partie pour verifier si le ticker € à yahoo finance
#     #                 # , "isYahooFinance"
#     #                 )]
#
#     # Return the data
#     return(data)
#   }else{
#     return(NULL)
#   }
#
#   # print(names(data))
#
#
# }


# # Use the function
```

```
# stock_data <- yf_isin_to_ticker("US0378331005")

# print(stock_data$symbol)

#

# yf_isin_to_ticker("FR0000131104")$symbol


# Fonction principale


analyze_portfolio_performance <- function(tickers, start_date = Sys.Date()-365, end_date = Sys.Date()) {
 all_data <- list()


 best_data_tbl <- list()


 all_data_tbl <- list()


 for (ticker in tickers) {
  # print('test0_nom du ticker')
  # print(ticker)
  Data <- try(tidyquant::tq_get(ticker,
                    get = "stock.prices",
                    from = as.Date(start_date),
                    to = as.Date(end_date) + 1),
         silent = TRUE)


  # print('test1_Dt of ticker')
  # print(head(Data, 2))


  # if (inherits(Data, "try-error") || nrow(Data) == 0) next
```

```
if (inherits(Data, "try-error") || is.null(Data) || nrow(Data) == 0) {
 next
}

names(Data) <- str_to_title(names(Data))
Data <- Data[, c("Symbol", "Date", "Close")]
names(Data)[1] <- "Ticker"

Data <- na.omit(Data)

if(!is.null(dim(Data))){
 # Assurez-vous que la colonne Close est numérique
 Data <- Data %>%
  mutate(Close = as.numeric(Close))%>%
  arrange(Date)

 Data$Return <- c(0, diff(log(Data$Close)))

 Data$Return <- round(Data$Return, 5)

 # Data <- Data%>%
 #  tq_transmute(select = Close,
 #               mutate_fun = periodReturn,
 #               period = "daily",
 #               col_rename = 'Return',
 #               type = 'log')
 #
 # Data$Ticker = ticker
 #
```

```
# Data <- Data[, c("Ticker", "Date", "Return")]


window = 2 # two days
# portfolio_returns = daily_returns
vol = rep(NA, length(Data$Return))
for (i in seq(window, length(Data$Return))) {
  # Calcul de la volatilité sur une fenêtre mobile
  vol[i] <- sd(Data$Return[(i - window + 1):i], na.rm = TRUE)
}


vol[which(is.na(vol))] = 0


vol = round(vol, 5)


Data$Volatility = vol


# # Calculer les rendements logarithmiques
# Data <- Data %>%
#   dplyr::mutate(Return = log(Close / lag(Close))) %>%
#   dplyr::filter(!is.na(Return))
#
# Data$Return = NA
# Data[[1, "Return"]] = 0
#
# for (j in 2:nrow(Data)){
#    i = j - 1
#    Data[[j, "Return"]] = log(Data[[j, "Close"]] / Data[[i, "Close"]])
#    # Data$Return[j] = log(Data$Return[j] / Data$Return[i])
#    # Data$Return[j] = 0
```

```
# }

first_price <- Data[Data$Date == min(Data$Date), "Close"][[1]]
last_price <- Data[Data$Date == max(Data$Date), "Close"][[1]]

years <- nrow(Data) / 252

annualized_returns <- (last_price / first_price)^(1 / years) - 1
# print('test2_rendement annualisés')
# print(annualized_returns)

# Data$LogReturn <- c(0, diff(log(Data$Close)))
# variance <- var(Data$LogReturn, na.rm = TRUE)
variance <- var(Data$Return, na.rm = TRUE)
annual_variance <- variance * 252
annual_volatility <- sqrt(annual_variance)

# print('test3_Volatilité')
# print(annual_volatility)

# the_hist <- get_historic_esg(ticker)

dt_ESG = get_historic_esg(ticker)
if(!is.null(dt_ESG)){
  dt_ESG = na.omit(dt_ESG)
  names(dt_ESG) = c("Date", "ESG", "E_Score", "S_Score", "G_Score")
  dt_ESG = dt_ESG[, c("Date", "ESG")]
```

```
# the_hist <- dt_ESG


# print('max Date histo')

# print(max(the_hist$Date))


ESG_note <- dt_ESG[dt_ESG$Date == max(dt_ESG$Date), "ESG"][[1]]

# print(ESG_note)


start_date <- as.Date(start_date)

end_date <- as.Date(end_date)


# Générez une séquence de dates entre start_date et end_date (inclusivement) pour les jours ouvrables (business days)


date_range <- base::seq(from = start_date, to = end_date, by = "days")

jours_semaine <- c("lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche")


if (any(weekdays(date_range, abbreviate = FALSE) %in% jours_semaine)){

 workday_dates <- date_range[weekdays(date_range) %in% c("lundi", "mardi", "mercredi", "jeudi", "vendredi")]


} else{

 jours_semaine_en <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")

 workday_dates <- date_range[weekdays(date_range, abbreviate = FALSE) %in% jours_semaine_en]


}


# Créez une dataframe avec les dates

df <- data.frame(Date = workday_dates)
```

```
# Fusionnez les dataframes en utilisant la colonne "Date"
# merged_df <- merge(dt_ESG, df,  by = "Date", all.x = TRUE)

merged_df <- dplyr::full_join(dt_ESG, df,  by = "Date")

# merged_df <- dplyr::full_join(dt_ESG, df,  by = "Date", keep = NULL, copy = TRUE)

merged_df <- merged_df%>%
 arrange(Date)

df = merged_df%>%
 tidyr::fill(ESG)

na_elm = which(is.na(df[,"ESG"]))
non_nul_elm = which(!is.na(df[,"ESG"]))

# Si les premiers elements sont encore des NA
if(length(na_elm)!=0){
 if(length(non_nul_elm)!=0){
  df[na_elm,"ESG"] = 0

 }

}

# Fusionnez les dataframes en utilisant la colonne "Date"
df_final <- merge(Data, df, by = "Date", all.x = TRUE)
```

```
  Data <- df_final%>%
    tidyr::fill(ESG)



 }else{
  Data$ESG <- 0
  ESG_note <- 0

 }

 # # Début pour obtenir les données historique ESG
 # if (is.null(the_hist) || nrow(the_hist) == 0) {
 #   ESG_note <- 0
 # }else{
 #   the_hist <- na.omit(the_hist)
 #   names(the_hist) <- c("Date", "ESG", "E_Score", "S_Score", "G_Score")
 #   the_hist <- the_hist[, c("Date", "ESG")]
 #
 #   # print('max Date histo')
 #   # print(max(the_hist$Date))
 #
 #   ESG_note <- the_hist[the_hist$Date == max(the_hist$Date), "ESG"][[1]]
 #   # print(ESG_note)
 #
 # }

 all_data[[ticker]] <- data.frame(Return = annualized_returns, Volatility = annual_volatility, ESG =
ESG_note, row.names = ticker)
```

```
    all_data_tbl[[ticker]] <- Data[, c("Ticker", "Date", "Close", "Return")]

    best_data_tbl[[ticker]] <- Data[, c("Ticker", "Date", "Close", "Return", "Volatility", "ESG")]

  }else{

    next

  }



 }

 result <- do.call(rbind, all_data)

 best_data_tbl <- do.call(rbind, best_data_tbl)

 result_tbl <- do.call(rbind, all_data_tbl)

 return(list(result = result,
         table = result_tbl,
         best_table = best_data_tbl))

}

# analyze_portfolio_performance <- function(tickers, start_date = Sys.Date()-365, end_date = Sys.Date()) {
#  all_data <- list()
#
```

```
#  all_data_tbl <- list()
#
#  for (ticker in tickers) {
#    # print('test0_nom du ticker')
#    # print(ticker)
#    Data <- try(tq_get(ticker,
#                       get = "stock.prices",
#                       from = as.Date(start_date),
#                       to = as.Date(end_date) + 1),
#                silent = TRUE)
#    # print('test1_Dt of ticker')
#    # print(head(Data, 2))
#
#    # if (inherits(Data, "try-error") || nrow(Data) == 0) next
#
#    if (inherits(Data, "try-error") || is.null(Data) || nrow(Data) == 0) {
#      next
#    }
#
#    names(Data) <- str_to_title(names(Data))
#    Data <- Data[, c("Symbol", "Date", "Close")]
#    names(Data)[1] <- "Ticker"
#
#    first_price <- Data[Data$Date == min(Data$Date), "Close"][[1]]
#    last_price <- Data[Data$Date == max(Data$Date), "Close"][[1]]
#
#    years <- nrow(Data) / 252
#
#    annualized_returns <- (last_price / first_price)^(1 / years) - 1
```

```
#    # print('test2_rendement annualisés')
#    # print(annualized_returns)
#
#    Data$LogReturn <- c(0, diff(log(Data$Close)))
#    variance <- var(Data$LogReturn, na.rm = TRUE)
#    annual_variance <- variance * 252
#    annual_volatility <- sqrt(annual_variance)
#
#    # print('test3_Volatilité')
#    # print(annual_volatility)
#
#
#    the_hist <- get_historic_esg(ticker)
#
#    # if(is.null(the_hist)){
#    #
#    #   ESG_note = 0
#    #
#    # }
#    if (is.null(the_hist) || nrow(the_hist) == 0) {
#     ESG_note <- 0
#    }else{
#     the_hist <- na.omit(the_hist)
#     names(the_hist) <- c("Date", "ESG", "E_Score", "S_Score", "G_Score")
#     the_hist <- the_hist[, c("Date", "ESG")]
#
#     # print('max Date histo')
#     # print(max(the_hist$Date))
#
```

```
#     ESG_note <- the_hist[the_hist$Date == max(the_hist$Date), "ESG"][[1]]
#     # print(ESG_note)
#
#
#     # if(max(the_hist$Date) < start_date){
#     #   ESG_note <- the_hist[the_hist$Date == max(the_hist$Date), "ESG"][[1]]
#     #
#     #   print('test4.1_histo ESG')
#     #   print(ESG_note)
#     #   # rbind(the_hist, c(end_date, the_hist[the_hist$Date == max(the_hist$Date), "ESG"][[1]])
#     # } else {
#     #
#     #   print('test4.2_Losk la date recent est plus grande que start_date')
#     #
#     #   print("the_head_histo")
#     #   print(head(the_hist))
#     #
#     #   print("the_tail_histo")
#     #   print(tail(the_hist))
#     #
#     #   date_range <- seq.Date(from = start_date, to = end_date, by = "day")
#     #
#           #     workday_dates <- date_range[weekdays(date_range) %in% c("Monday", "Tuesday",
"Wednesday", "Thursday", "Friday")]
#     #   df <- data.frame(Date = workday_dates)
#     #   merged_df <- merge(df, the_hist, by = "Date", all.x = TRUE)
#     #
#     #   df = merged_df%>%fill(ESG)
#     #
```

```
#     #   na_elm = which(is.na(df[,"ESG"]))
#     #   non_nul_elm = which(!is.na(df[,"ESG"]))
#     #
#     #   # Si les premiers elements sont encore des NA
#     #   if(length(na_elm)!=0){
#     #     if(length(non_nul_elm)!=0){
#     #       df[na_elm,"ESG"] = 0
#     #
#     #     }
#     #
#     #   }
#     #
#     #   print(class(df))
#     #   df = as.data.frame(df)
#     #
#     #
#     #   print('test_sur les données ESG')
#     #   print(head(df, 10))
#     #
#     #   print('test_la dernière donnée ESG')
#     #   print(tail(df, 1))
#     #   ESG_note <- df[df$Date == max(df$Date), "ESG"][[1]]
#     #   print(ESG_note)
#     #   print('test5_ESG recent note')
#     #   print(ESG_note)
#     # }
#   }
#
```

```
#     all_data[[ticker]] <- data.frame(Return = annualized_returns, Volatility = annual_volatility, ESG = ESG_note, row.names = ticker)
#
#     all_data_tbl[[ticker]] <- Data[, c("Ticker", "Date", "Close", "LogReturn")]
#   }
#
#   result <- do.call(rbind, all_data)
#
#   result_tbl <- do.call(rbind, all_data_tbl)
#
#   return(list(result = result,
#               table = result_tbl))
# }



# Pour optimiser le portefeuille
# La fonction est la plus meilleur

# Définir la fonction pour analyser et tracer la performance du portefeuille
analyze_portfolio_performance_plot <- function(performance_data,
                                               num_simulations = 100000,
                                               type_simulation = 'uniform',
                                               size_type = 15,
                                               user_weights = NULL) {

  # Préparer les données pour la simulation
  returns <- performance_data$Return
  volatilities <- performance_data$Volatility
  esg_scores <- performance_data$ESG
```

```
 tickers <- rownames(performance_data)

 # Calculer la matrice de covariance des rendements
 cov_matrix <- cov(matrix(performance_data$Volatility, nrow = length(tickers), ncol = length(tickers)))

 # Initialiser une liste pour stocker les portefeuilles simulés
 simulated_portfolios <- vector("list", num_simulations)

 # Vérifier les poids fournis par l'utilisateur
 if (!is.null(user_weights)) {
  if (sum(user_weights) > 1) {
   stop("Total weight should be at most 1 (i.e., 100%).")
  }
 }

 for (i in 1:num_simulations) {
  # Générer des poids en fonction du type de simulation
  if (!is.null(user_weights)) {
   weights <- user_weights

  } else {
   if (type_simulation == 'uniform') {
    weights <- runif(length(tickers))
    weights <- weights / sum(weights)

   } else if (type_simulation == 'montecarlo') {
    weights <- abs(rnorm(length(tickers)))
    weights <- weights / sum(weights)
```

```
    } else if (type_simulation == 'dirichlet') {
      weights <- rdirichlet(1, rep(1, length(tickers)))[1,]
    } else {
      stop("Invalid type_simulation. Choose from 'uniform', 'montecarlo', 'dirichlet'.")
    }
  }

  # Calculer le retour du portefeuille
  port_return <- sum(weights * returns)

  # Calculer la volatilité du portefeuille
  port_volatility <- sqrt(t(weights) %*% cov_matrix %*% weights)

  # Calculer le score ESG du portefeuille
  port_esg <- sum(weights * esg_scores)

  # Stocker les résultats
  simulated_portfolios[[i]] <- list(Return = port_return, Volatility = port_volatility, ESG = port_esg, Weights = weights)
 }

 # Extraire les résultats des portefeuilles simulés
 returns <- sapply(simulated_portfolios, function(x) x$Return)
 volatilities <- sapply(simulated_portfolios, function(x) x$Volatility)
 esg_scores <- sapply(simulated_portfolios, function(x) x$ESG)
 weights_matrix <- do.call(rbind, lapply(simulated_portfolios, function(x) x$Weights))

 # Normaliser les données
 volatilities_norm <- (volatilities - min(volatilities)) / (max(volatilities) - min(volatilities))
```

```
# J'ai décidé de ne pas normaliser les rendements et notes ESG
# returns_norm <- (returns - min(returns)) / (max(returns) - min(returns))
# esg_scores_norm <- (esg_scores - min(esg_scores)) / (max(esg_scores) - min(esg_scores))

# Créer un dataframe des résultats
results <- data.frame(Return = returns, Volatility = volatilities_norm, ESG = esg_scores)
# results <- data.frame(Return = returns_norm, Volatility = volatilities_norm, ESG = esg_scores_norm)

df <- results %>%
  dplyr::select(Return, Volatility, ESG) %>%
  mutate(
    Profit = case_when(
      Return > 0 ~ "Gain",
      Return < 0 ~ "Loss",
      TRUE ~ "Neutral"
    )
  )

# Si j'ai normalisé les rendements
# df <- results %>%
#   dplyr::select(Return, Volatility, ESG) %>%
#   mutate(
#     Profit = case_when(
#       Return > 0.5 ~ "Gain", #A cause du ranking
#       Return < 0.5 ~ "Loss", #A cause du ranking
#       TRUE ~ "Neutral"
#     )
```

```
#  )

df$Profit <- factor(df$Profit, levels = c("Loss", "Gain", "Neutral"))

# Créer un graphique 3D avec plotly
fig <- plot_ly(df, x = ~Return,
          y = ~Volatility,
          z = ~ESG,
          color = ~Profit,
          colors = c('#BF382A', "darkgreen", '#0C4B8E'))

fig <- fig %>% add_markers(marker = list(size = size_type))
fig <- fig %>% layout(scene = list(xaxis = list(title = 'Return'),
                        yaxis = list(title = 'Volatility'),
                        zaxis = list(title = 'ESG score')),
                title = "Simulated Portfolio Performance")

# Trouver les poids optimaux en fonction des simulations
# optimal_weights <- weights_matrix[which.max(returns_norm), ]
optimal_weights <- weights_matrix[which.max(returns), ]

# Retourner la figure, les poids optimaux, le dataframe des simulations et la matrice variance covariance
return(list(fig = fig,
        optimal_weights = optimal_weights,
        cov_matrix = cov_matrix,  #j'ai ajouter la matrice variance covariance
        simulation = results,
        fig_df = df) #fig_df is the figure dataframe
)
```

```
}



# Définir la fonction pour analyser et tracer la performance du portefeuille
# Eviter la simulation
# Meilleure fonction
update_analyze_pf_perf_plot <- function(performance_data,
                                        # num_simulations = 100000,
                                        # type_simulation = 'uniform',
                                        size_type = 15,
                                        user_weights = NULL) {

  # Préparer les données pour la simulation
  returns <- performance_data$Return
  volatilities <- performance_data$Volatility
  esg_scores <- performance_data$ESG
  # tickers <- rownames(performance_data)
  tickers <- unique(performance_data$Ticker)



  spread_dt <- performance_data[, c("Ticker", "Date", "Return")] %>%
    spread(Ticker, value = Return) %>%
    tk_xts(date_var = "Date", silent = TRUE)

  # Calculer la matrice de covariance des rendements
  cov_matrix <- cov(spread_dt)*252

  # Calculer la matrice de covariance des rendements
  # cov_matrix <- cov(matrix(volatilities, nrow = length(tickers), ncol = length(tickers)))
```

```
l_ticker = length(tickers)
# Normalisation des poids utilisateur s'ils sont fournis
if (!is.null(user_weights)) {
  if(sum(user_weights) == 0 ){
    user_weights = rep((1/l_ticker), l_ticker)

  } else if(sum(user_weights) > 0 && sum(user_weights) <= 1){
    user_weights = user_weights

  } else if(sum(user_weights) > 1 && sum(user_weights) <= 100 ){
    user_weights = user_weights/100

  } else{
    user_weights <- user_weights / sum(user_weights)

  }

}else{
  user_weights = rep((1/l_ticker), l_ticker)
}

port_ret <- performance_data %>%
  tq_portfolio(assets_col = Ticker,
               returns_col = Return,
               weights = user_weights,
               col_rename = 'Return',
               geometric = FALSE)
```

```
# Annualize the portfolio returns
port_ret$Annualised_return <- port_ret$Return * 252

# Calculate daily portfolio volatility
port_volatility <- performance_data %>%
 tq_portfolio(assets_col = Ticker,
           returns_col = Volatility,
           weights = user_weights,
           col_rename = 'Volatility',
           geometric = FALSE)

# Annualize portfolio volatility
port_volatility$Annualised_volatility <- port_volatility$Volatility * sqrt(252)

port_ESG <- performance_data %>%
 tq_portfolio(assets_col = Ticker,
           returns_col = ESG,
           weights = user_weights,
           col_rename = 'ESG',
           geometric = FALSE)

port_data <- port_ret%>%
 mutate(Volatility = port_volatility$Volatility,
      Annualised_volatility = port_volatility$Annualised_volatility,
      ESG = port_ESG$ESG)

# Créer un dataframe des résultats
results = port_data[, c("Date", "Annualised_return", "Annualised_volatility", "ESG")]
```

```
names(results) = c("Date", "Return", "Volatility", "ESG")
# print("--Head result")
# print(head(results))
# print(tail(results))

df <- results %>%
 dplyr::select(Return, Volatility, ESG) %>%
 mutate(
  Profit = case_when(
   Return > 0 ~ "Gain",
   Return < 0 ~ "Loss",
   TRUE ~ "Neutral"
  )
 )

df$Profit <- factor(df$Profit, levels = c("Loss", "Gain", "Neutral"))

# Créer un graphique 3D avec plotly
fig <- plot_ly(df, x = ~Return,
          y = ~Volatility,
          z = ~ESG,
          color = ~Profit,
          colors = c('#BF382A', "darkgreen", '#0C4B8E'))

fig <- fig %>% add_markers(marker = list(size = size_type))
fig <- fig %>% layout(scene = list(xaxis = list(title = 'Return'),
                       yaxis = list(title = 'Volatility'),
                       zaxis = list(title = 'ESG score')),
```

```
                title = "Simulated Portfolio Performance")


 # Trouver les poids optimaux en fonction des simulations
 # optimal_weights <- weights_matrix[which.max(returns), ]



 # Retourner la figure, les poids optimaux, le dataframe des simulations et la matrice variance covariance
 return(list(fig = fig,
        # optimal_weights = optimal_weights,
        cov_matrix = cov_matrix,  #j'ai ajouter la matrice variance covariance
        # simulation = results,
        full_data = port_data,
        fig_df = df) #fig_df is the figure dataframe
 )
}



# Deuxième option pour la fonction update_analyze_pf_perf_plot
# update_analyze_pf_perf_plot <- function(performance_data,
#                         type_return = "daily",
#                         # rf,
#                         # num_simulations = 100000,
#                         # type_simulation = 'uniform',
#                         size_type = 15,
#                         user_weights = NULL) {
#
#   # Préparer les données pour la simulation
#   returns <- performance_data$Return
```

```
#   volatilities <- performance_data$Volatility
#   esg_scores <- performance_data$ESG
#   # tickers <- rownames(performance_data)
#   tickers <- unique(performance_data$Ticker)
#
#
#   spread_dt <- performance_data[, c("Ticker", "Date", "Return")] %>%
#    spread(Ticker, value = Return) %>%
#    tk_xts(date_var = "Date", silent = TRUE)
#
#   annualization_factor <- switch(type_return,
#                          "daily" = 252,
#                          "weekly" = 52,
#                          "monthly" = 12,
#                          "yearly" = 1,
#                          stop("type_return doit être 'daily', 'weekly', 'monthly' ou 'yearly'"))
#
#   # # Traitement du taux sans risque (Risk Free Rate)
#   # if (is.numeric(rf) && length(rf) == 1) {
#   #   rf_return <- rf / annualization_factor
#   # } else if (is.character(rf) && length(rf) == 1) {
#   #   rf_data <- tq_get(rf, from = start_date, to = end_date, get = 'stock.prices')
#   #   rf_log_ret_tidy <- rf_data %>%
#   #    tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename
= 'rf_return', type = 'log')
#   #   rf_return <- mean(rf_log_ret_tidy$rf_return)
#   #
#   # } else {
#   #   stop("rf doit être un taux numérique ou un ticker valide.")
```

```
#  #
#  # }
#
#  # Calculer la matrice de covariance des rendements
#  cov_matrix <- cov(spread_dt)*annualization_factor
#
#  # Calculer la matrice de covariance des rendements
#  # cov_matrix <- cov(matrix(volatilities, nrow = length(tickers), ncol = length(tickers)))
#
#  l_ticker = length(tickers)
#  # Normalisation des poids utilisateur s'ils sont fournis
#  if (!is.null(user_weights)) {
#   if(sum(user_weights) == 0 ){
#    user_weights = rep((1/l_ticker), l_ticker)
#
#   } else if(sum(user_weights) > 0 && sum(user_weights) <= 1){
#    user_weights = user_weights
#
#   } else if(sum(user_weights) > 1 && sum(user_weights) <= 100 ){
#    user_weights = user_weights/100
#
#   } else{
#    user_weights <- user_weights / sum(user_weights)
#
#   }
#
#  }else{
#   user_weights = rep((1/l_ticker), l_ticker)
#  }
```

```
#
#  port_ret <- performance_data %>%
#    tq_portfolio(assets_col = Ticker,
#                 returns_col = Return,
#                 weights = user_weights,
#                 col_rename = 'Return',
#                 geometric = FALSE)
#
#
#  # Annualize the portfolio returns
#  port_ret$Annualised_return <- port_ret$Return * annualization_factor
#  # port_ret$Annualised_return <- ((port_ret$Return + 1)^annualization_factor) - 1
#
#  # Calculate daily portfolio volatility
#  port_volatility <- performance_data %>%
#    tq_portfolio(assets_col = Ticker,
#                 returns_col = Volatility,
#                 weights = user_weights,
#                 col_rename = 'Volatility',
#                 geometric = FALSE)
#
#  # Annualize portfolio volatility
#  port_volatility$Annualised_volatility <- port_volatility$Volatility * sqrt(252)
#
#  port_ESG <- performance_data %>%
#    tq_portfolio(assets_col = Ticker,
#                 returns_col = ESG,
#                 weights = user_weights,
#                 col_rename = 'ESG',
```

```
#             geometric = FALSE)
#
#  port_data <- port_ret%>%
#   mutate(Volatility = port_volatility$Volatility,
#          Annualised_volatility = port_volatility$Annualised_volatility,
#          ESG = port_ESG$ESG)
#
#  # Créer un dataframe des résultats
#  results = port_data[, c("Date", "Annualised_return", "Annualised_volatility", "ESG")]
#
#  names(results) = c("Date", "Return", "Volatility", "ESG")
#
#  # #Nouveau ajout
#  # # Pour pouvoir l'utiliser il faut le décommenter
#  # # for efficient portf
#  # new_port_dt = port_data[, c("Date", "Return", "Volatility", "Annualised_volatility")]
#  # if(sum(new_port_dt[1,-1]) == 0){
#  #   new_port_dt = new_port_dt[-1,]
#  # }
#  #
#  # wts = user_weights
#  # port_ret <- ((new_port_dt$Return + 1)^annualization_factor) - 1
#  # port_returns <- port_ret
#  #
#  # print("port return")
#  # print(port_ret)
#  #
#  # port_sd <- sqrt(t(wts) %*% (cov_matrix %*% wts))
#  # port_sd <- port_sd[1]
```

```
#   # # port_sd <- new_port_dt$Annualised_volatility

#   # port_risk <- port_sd

#   #

#   # print("port risk")

#   # print(port_sd)

#   #

#   # sharpe_ratio <- (port_ret - rf_return) / port_sd

#   #

#   # portfolio_values <- tibble(Return = port_returns, Risk = port_risk, SharpeRatio = sharpe_ratio)

#   #

#       # all_wts = as.data.frame(matrix(user_weights, ncol = length(user_weights), nrow = nrow(portfolio_values)))

#   # # all_wts <- as_tibble(wts)

#   #

#   # colnames(all_wts) <- tickers

#   #

#   # portfolio_values <- bind_cols(all_wts, portfolio_values)

#   #

#   # min_var <- portfolio_values[which.min(portfolio_values$Risk),]

#   #

#   # max_sr <- portfolio_values[which.max(portfolio_values$SharpeRatio),]

#   #

#   # # Graphiques

#   # min_var_plot <- min_var %>%

#   #   gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%

#   #   ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +

#   #   geom_bar(stat = "identity") +

#   #   theme_minimal() +

#   #   labs(x = 'Assets', y = 'Weights', title = "Minimum variance portfolio Weights") +
```

```
#  #  scale_y_continuous(labels = scales::percent)
#  #
#  # max_sr_plot <- max_sr %>%
#  #  gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%
#  #  ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +
#  #  geom_bar(stat = "identity") +
#  #  theme_minimal() +
#  #  labs(x = 'Assets', y = 'Weights', title = "Tangency portfolio weights") +
#  #  scale_y_continuous(labels = scales::percent)
#  #
#  # eff_frontier_plot <- portfolio_values %>%
#  #  ggplot(aes(x = Risk, y = Return, color = SharpeRatio)) +
#  #  geom_point() +
#  #  theme_classic() +
#  #  scale_y_continuous(labels = scales::percent) +
#  #  scale_x_continuous(labels = scales::percent) +
#  #   labs(x = 'Annualized Risk', y = 'Annualized Returns', title = "Portfolio Optimization & Efficient
Frontier") +
#  #  geom_point(aes(x = Risk, y = Return #, size = 3
#  #  ), shape = 22, fill = "red", data = min_var) +  # Point carré
#  #  geom_point(aes(x = Risk, y = Return #, size = 3
#  #  ), shape = 24, fill = "red", data = max_sr) +  # Point triangle
#  #  annotate('text', x = min_var$Risk, y = min_var$Return, label = "Minimum variance portfolio") +
#  #  annotate('text', x = max_sr$Risk, y = max_sr$Return, label = "Tangency Portfolio")
#
#
#  df <- results %>%
#   dplyr::select(Return, Volatility, ESG) %>%
#   mutate(
```

```
#     Profit = case_when(
#       Return > 0 ~ "Gain",
#       Return < 0 ~ "Loss",
#       TRUE ~ "Neutral"
#     )
#   )
#
#   df$Profit <- factor(df$Profit, levels = c("Loss", "Gain", "Neutral"))
#
#   # Créer un graphique 3D avec plotly
#   fig <- plot_ly(df, x = ~Return,
#               y = ~Volatility,
#               z = ~ESG,
#               color = ~Profit,
#               colors = c('#BF382A', "darkgreen", '#0C4B8E'))
#
#   fig <- fig %>% add_markers(marker = list(size = size_type))
#   fig <- fig %>% layout(scene = list(xaxis = list(title = 'Return'),
#                                      yaxis = list(title = 'Volatility'),
#                                      zaxis = list(title = 'ESG score')),
#                         title = "Simulated Portfolio Performance")
#
#
#
#   # Retourner la figure, les poids optimaux, le dataframe des simulations et la matrice variance covariance
#   return(list(fig = fig,
#           # optimal_weights = optimal_weights,
#           cov_matrix = cov_matrix,  #j'ai ajouter la matrice variance covariance
```

```
#           # simulation = results,
#           full_data = port_data,
#           # Utiliser plutôt la version de simulation pour les mi et max
#           # min_var_port_w = ggplotly(min_var_plot),
#           # max_var_port_w = ggplotly(max_sr_plot),
#           # pf_opt_eff_front = ggplotly(eff_frontier_plot),
#           fig_df = df) #fig_df is the figure dataframe
#   )
# }

# besty = update_analyze_pf_perf_plot(performance_data = portfolio_performance$best_table,
#                         size_type = 15,
#                         rf = 0.05,
#                         type_return = "daily",
#                         user_weights = NULL
# )



# ## J'ai ajusté pour mieux prendre en compte les user weights
# # Ancienne version quui fonctionne aussi mais elle est lente.
# best_analyze_pf_perf <- function(tickers_to_analyze,
#                         rf,
#                         start_date = Sys.Date()-365,
#                         end_date = Sys.Date(),
#                         type_return = 'daily',
#                         user_weights = NULL,
#                         num_simulations = 500,
#                         type_simulation = 'uniform') {
```

```
#
#  # Téléchargement des données de prix
#  price_data <- tidyquant::tq_get(tickers_to_analyze,
#                        from = start_date,
#                        to = end_date,
#                        get = 'stock.prices')
#
#  # Calcul des rendements log
#  log_ret_tidy <- price_data %>%
#    dplyr::group_by(symbol) %>%
#    tidyquant::tq_transmute(select = adjusted,
#                      mutate_fun = periodReturn,
#                      period = type_return,
#                      col_rename = 'ret',
#                      type = 'log')
#
#  # Remove na lines
#  log_ret_tidy <- na.omit(log_ret_tidy)
#
#  log_ret_xts <- log_ret_tidy %>%
#    spread(symbol, value = ret) %>%
#    timetk::tk_xts(date_var = "date")
#
#  mean_ret <- colMeans(log_ret_xts)
#
#  annualization_factor <- switch(type_return,
#                        "daily" = 252,
#                        "weekly" = 52,
#                        "monthly" = 12,
```

```
#                       "yearly" = 1,
#                       stop("type_return doit être 'daily', 'weekly', 'monthly' ou 'yearly'"))
#
#  cov_mat <- cov(log_ret_xts) * annualization_factor
#
#  # Traitement du taux sans risque (Risk Free Rate)
#  if (is.numeric(rf) && length(rf) == 1) {
#    rf_return <- rf / annualization_factor
#
#  } else if (is.character(rf) && length(rf) == 1) {
#    rf_data <- tq_get(rf, from = start_date, to = end_date, get = 'stock.prices')
#    rf_log_ret_tidy <- rf_data %>%
#      tq_transmute(select = adjusted, mutate_fun = periodReturn, period = type_return, col_rename
= 'rf_return', type = 'log')
#    rf_return <- mean(rf_log_ret_tidy$rf_return)
#
#  } else {
#    stop("rf doit être un taux numérique ou un ticker valide.")
#
#  }
#
#  # Normalisation des poids utilisateur s'ils sont fournis
#  if (!is.null(user_weights)) {
#    if(sum(user_weights) == 0 ){
#      l_ticker = length(tickers_to_analyze)
#      user_weights = rep((1/l_ticker), l_ticker)
#
#    } else if(sum(user_weights) > 0 && sum(user_weights) <= 1){
#      user_weights = user_weights
```

```
#
#   } else if(sum(user_weights) > 1 && sum(user_weights) <= 100 ){
#     user_weights = user_weights/100
#
#   } else{
#     user_weights <- user_weights / sum(user_weights)
#
#   }
#
# }
#
#  num_simulations = as.numeric(num_simulations)
#
#  # Initialisation des matrices et vecteurs pour les simulations
#  all_wts <- matrix(nrow = num_simulations, ncol = length(tickers_to_analyze))
#
#  port_returns <- vector('numeric', length = num_simulations)
#  # print(head(port_returns))
#
#  port_risk <- vector('numeric', length = num_simulations)
#  # print(head(port_risk))
#
#  sharpe_ratio <- vector('numeric', length = num_simulations)
#  # print(head(sharpe_ratio))
#
#  # port_esg <- vector('numeric', length = num_simulations)
#
#
#
#  # Simulation des portefeuilles
```

```
#  for (i in 1:num_simulations) {
#   if (!is.null(user_weights) && i == 1) {
#     wts <- user_weights
#
#   } else {
#     if (type_simulation == 'uniform') {
#       wts <- runif(length(tickers_to_analyze))
#
#     } else if (type_simulation == 'montecarlo') {
#       wts <- abs(rnorm(length(tickers_to_analyze)))
#
#     } else if (type_simulation == 'dirichlet') {
#       wts <- rgamma(length(tickers_to_analyze), shape = 1, scale = 1)
#
#     } else {
#       stop("type_simulation doit être 'uniform' ou 'dirichlet'.")
#
#     }
#     wts <- wts / sum(wts)
#   }
#
#   all_wts[i,] <- wts
#
#   port_ret <- sum(wts * mean_ret)
#   port_ret <- ((port_ret + 1)^annualization_factor) - 1
#   port_returns[i] <- port_ret
#
#   port_sd <- sqrt(t(wts) %*% (cov_mat %*% wts))
#   port_risk[i] <- port_sd
```

```
#
#    sharpe_ratio[i] <- (port_ret - rf_return) / port_sd
#
#  }
#
#  portfolio_values <- tibble(Return = port_returns, Risk = port_risk, SharpeRatio = sharpe_ratio)
#
#  all_wts <- as_tibble(all_wts)
#
#  colnames(all_wts) <- tickers_to_analyze
#
#  portfolio_values <- bind_cols(all_wts, portfolio_values)
#
#  min_var <- portfolio_values[which.min(portfolio_values$Risk),]
#
#  max_sr <- portfolio_values[which.max(portfolio_values$SharpeRatio),]
#
#  # Calcul du rendement, du risque et des ratios pour le portefeuille initial (réel)
#  if (is.null(user_weights)) {
#    user_weights <- runif(length(tickers_to_analyze))
#    user_weights <- user_weights / sum(user_weights)
#
#  }
#
#  real_port_ret <- sum(user_weights * mean_ret)
#
#  real_port_ret <- ((real_port_ret + 1)^annualization_factor) - 1
#
#  real_port_risk <- sqrt(t(user_weights) %*% (cov_mat %*% user_weights))
```

```
#
#  real_sharpe_ratio <- (real_port_ret - rf_return) / real_port_risk
#
#                                                                    optimal_w                    <-
portfolio_values[which.max(portfolio_values$Return),][,c(1:length(tickers_to_analyze))]
#
#  # Pour obtenir la table (ie real_data)
#  if (is.null(user_weights)) {
#   wts <- as.numeric(optimal_w[1,])
#   wts_tbl <- tibble(symbol = tickers_to_analyze,
#                     wts = wts)
#   # Merge data
#   real_data <- dplyr::left_join(log_ret_tidy, wts_tbl, by = 'symbol')
#
#   real_data <- real_data %>%
#     dplyr::mutate(wt_return = wts * ret)
#
#   ptf_ret <- real_data %>%
#     group_by(date) %>%
#     summarise(Portfolio_Return = sum(wt_return))
#
#  } else {
#   wts <- user_weights
#   wts_ <- as.numeric(optimal_w[1,])
#   wts_tbl <- tibble(symbol = tickers_to_analyze,
#                     wts = wts, #user weights
#                     opt_wts = wts_ #optimal weight
#   )
#
```

```
#    # Merge data
#    real_data <- dplyr::left_join(log_ret_tidy, wts_tbl, by = 'symbol')
#
#    real_data <- real_data %>%
#      dplyr::mutate(wt_return = wts * ret,
#                    opt_wt_return = opt_wts * ret)
#
#    ptf_ret <- real_data %>%
#      dplyr::group_by(date) %>%
#      dplyr::summarise(Portfolio_Return = sum(wt_return),
#                       Opt_Portfolio_Return = sum(opt_wt_return))
#
#  }
#
#  sav_real_dt <- data.frame(date = index(log_ret_xts), coredata(log_ret_xts))
#
#  # Merge data
#  real_data <- dplyr::left_join(sav_real_dt, ptf_ret, by = 'date')
#
#  # Calcul du risque et du Sharpe ratio pour les vraies données
#  width <- switch(type_return,
#                  "daily" = 252,
#                  "weekly" = 52,
#                  "monthly" = 12,
#                  "yearly" = 1)
#
#  real_data <- real_data %>%
#    dplyr::mutate(
#      Portfolio_Risk = rollapply(Portfolio_Return, width = width, FUN = sd, align = "right", fill = NA)
```

```
#     # ,
#     # Sharpe_Ratio = (Portfolio_Return - rf_return) / Portfolio_Risk
#   )
#
#  window = 5
#  portfolio_returns = real_data$Portfolio_Return
#  portfolio_risk = real_data$Portfolio_Risk
#  for (i in seq(window, nrow(real_data))) {
#   # Calcul de la volatilité sur une fenêtre mobile
#   portfolio_risk[i] <- sd(portfolio_returns[(i - window + 1):i], na.rm = TRUE)
#  }
#
#  portfolio_risk[which(is.na(portfolio_risk))] = 0
#
#  real_data$Portfolio_Risk = portfolio_risk
#
#  real_data <- real_data %>%
#   dplyr::mutate(
#    Sharpe_Ratio = (Portfolio_Return - rf_return) / Portfolio_Risk
#   )
#
#
#  # Graphiques
#  min_var_plot <- min_var %>%
#   gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%
#   ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +
#   geom_bar(stat = "identity") +
#   theme_minimal() +
#   labs(x = 'Assets', y = 'Weights', title = "Minimum variance portfolio Weights") +
```

```
#    scale_y_continuous(labels = scales::percent)
#
#  max_sr_plot <- max_sr %>%
#    gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%
#    ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +
#    geom_bar(stat = "identity") +
#    theme_minimal() +
#    labs(x = 'Assets', y = 'Weights', title = "Tangency portfolio weights") +
#    scale_y_continuous(labels = scales::percent)
#
#  eff_frontier_plot <- portfolio_values %>%
#    ggplot(aes(x = Risk, y = Return, color = SharpeRatio)) +
#    geom_point() +
#    theme_classic() +
#    scale_y_continuous(labels = scales::percent) +
#    scale_x_continuous(labels = scales::percent) +
#      labs(x = 'Annualized Risk', y = 'Annualized Returns', title = "Portfolio Optimization & Efficient
Frontier") +
#    geom_point(aes(x = Risk, y = Return #, size = 3
#    ), shape = 22, fill = "red", data = min_var) +  # Point carré
#    geom_point(aes(x = Risk, y = Return #, size = 3
#    ), shape = 24, fill = "red", data = max_sr) +  # Point triangle
#    annotate('text', x = min_var$Risk, y = min_var$Return, label = "Minimum variance portfolio") +
#    annotate('text', x = max_sr$Risk, y = max_sr$Return, label = "Tangency Portfolio")
#
#  return(list(
#    table = real_data,
#    table_simul = portfolio_values,
#    Optimal_weight = optimal_w,
```

```
#    min_var_port_w = ggplotly(min_var_plot),
#    max_var_port_w = ggplotly(max_sr_plot),
#    pf_opt_eff_front = ggplotly(eff_frontier_plot)
#  ))
# }



## Nouvelle fonction qui fonctionne très bien et elle est rapide
## Nouvelle fonction qui fonctionne très bien et elle est rapide
best_analyze_pf_perf <- function(data,
                        rf = 0.02,
                        type_return = 'daily',
                        user_weights = NULL,
                        num_simulations = 500,
                        type_simulation = 'uniform') {


 # Validation des entrées
 validate_inputs <- function() {
  required_cols <- c("Ticker", "Date", "Return", "Volatility", "ESG")
  if (!all(required_cols %in% colnames(data))) {
   stop("Missing required columns: ",
        paste(setdiff(required_cols, colnames(data)), collapse = ", "))
  }

  if (!is.numeric(rf)) stop("rf must be numeric")

  valid_types <- c('daily', 'weekly', 'monthly', 'yearly')
  if (!type_return %in% valid_types) {
   stop("type_return must be one of: ", paste(valid_types, collapse = ", "))
```

```
 }
}

validate_inputs()

# Constants
ANNUALIZATION_FACTORS <- c(daily = 252, weekly = 52, monthly = 12, yearly = 1)
annualization_factor <- ANNUALIZATION_FACTORS[[type_return]]
rf_return <- rf / annualization_factor

# Data preparation
prepare_data <- function() {
 tickers_to_analyze <- unique(data$Ticker)

 log_ret_tidy <- data %>%
  dplyr::select(date = Date, symbol = Ticker, ret = Return, esg = ESG) %>%
  dplyr::arrange(date, symbol)

 log_ret_xts <- log_ret_tidy %>%
  dplyr::select(date, symbol, ret) %>%
  tidyr::spread(symbol, ret) %>%
  timetk::tk_xts(date_var = "date", silent = TRUE)

 # log_ret_xts <- log_ret_tidy %>%
 #  dplyr::select(date, symbol, ret) %>%
 #  tidyr::spread(symbol, ret) %>%
 #  timetk::tk_xts(date_var = "date") %>%
 #  # Assurez-vous que seules les colonnes numériques sont présentes
 #  as.matrix()
```

```
  esg_scores <- data %>%
    dplyr::group_by(Ticker) %>%
    dplyr::summarise(ESG = mean(ESG)) %>%
    dplyr::pull(ESG, name = Ticker)

  list(
    tickers = tickers_to_analyze,
    ret_tidy = log_ret_tidy,
    ret_xts = log_ret_xts,
    esg = esg_scores
  )
}

data_prep <- prepare_data()

# Portfolio calculations
calculate_portfolio_metrics <- function(weights, mean_returns, cov_matrix, esg) {
  port_ret <- sum(weights * mean_returns)
  port_ret <- ((port_ret + 1)^annualization_factor) - 1
  port_sd <- sqrt(as.numeric(t(weights) %*% (cov_matrix %*% weights)))
  sharpe <- (port_ret - rf_return) / port_sd
  esg_score <- sum(weights * esg)

  c(Return = port_ret, Risk = port_sd, SharpeRatio = sharpe, ESG = esg_score)
}

# Generate weights
generate_weights <- function(n_assets, method) {
```

```
 weights <- switch(method,
            "uniform" = runif(n_assets),
            "montecarlo" = abs(rnorm(n_assets)),
            "dirichlet" = rgamma(n_assets, shape = 1, scale = 1),
            stop("Invalid simulation type"))
 weights / sum(weights)
}

# Main calculations
mean_ret <- colMeans(data_prep$ret_xts)
cov_mat <- cov(data_prep$ret_xts) * annualization_factor
n_assets <- length(data_prep$tickers)

# Normalize user weights if provided
if (!is.null(user_weights)) {
 user_weights <- if(sum(user_weights) == 0) {
  rep(1/n_assets, n_assets)
 } else if(sum(user_weights) > 0 && sum(user_weights) <= 1) {
  user_weights
 } else if(sum(user_weights) > 1 && sum(user_weights) <= 100) {
  user_weights/100
 } else {
  user_weights / sum(user_weights)
 }
}

# Simulate portfolios
weights_matrix <- matrix(
 replicate(num_simulations, generate_weights(n_assets, type_simulation)),
```

```
  nrow = num_simulations, byrow = TRUE
)

if (!is.null(user_weights)) {
  weights_matrix[1,] <- user_weights
}

# Calculate portfolio metrics for all simulations
portfolio_metrics <- t(apply(weights_matrix, 1, calculate_portfolio_metrics,
                             mean_ret, cov_mat, data_prep$esg))

# Create results tibble
portfolio_values <- as_tibble(cbind(
  weights_matrix,
  portfolio_metrics
))

colnames(portfolio_values)[1:n_assets] <- data_prep$tickers

# Find optimal portfolios
min_var <- portfolio_values[which.min(portfolio_values$Risk),]
# print("head(min_var)")
# print(head(min_var))
max_sr <- portfolio_values[which.max(portfolio_values$SharpeRatio),]
# print("head(max_sr)")
# print(head(max_sr))
optimal_w <- portfolio_values[which.max(portfolio_values$Return),][,1:n_assets]

# Generate plots
```

```
# Création des graphiques
# [Code des graphiques inchangé, mais ajout de l'ESG dans les visualisations]

# Création du graph 3 D

generate_plots <- function() {
 # Graphiques
 min_var_plot <- min_var %>%
  dplyr::select(-ESG)%>%
  gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%
  ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(x = 'Assets', y = 'Weights', title = "Minimum variance portfolio Weights") +
  scale_y_continuous(labels = scales::percent)

 max_sr_plot <- max_sr %>%
  dplyr::select(-ESG)%>%
  gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%
  ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(x = 'Assets', y = 'Weights', title = "Tangency portfolio weights") +
  scale_y_continuous(labels = scales::percent)

 eff_frontier_plot <- portfolio_values %>%
  ggplot(aes(x = Risk, y = Return, color = SharpeRatio)) +
  geom_point() +
  theme_classic() +
```

```
scale_y_continuous(labels = scales::percent) +
scale_x_continuous(labels = scales::percent) +
labs(x = 'Annualized Risk', y = 'Annualized Returns',
   title = "Portfolio Optimization & Efficient Frontier") +
# Point pour le portefeuille à variance minimale
geom_point(data = min_var,
        aes(x = Risk, y = Return),
        shape = 22,
        fill = "red",
        color = "red",
        size = 3) +
# Point pour le portefeuille tangent
geom_point(data = max_sr,
        aes(x = Risk, y = Return),
        shape = 24,
        # shape = 8,
        fill = "red",
        color = "red",
        size = 5) +
# Annotations
annotate('text',
      x = min_var$Risk,
      y = min_var$Return,
      label = "Minimum variance portfolio",
      vjust = -1) +
annotate('text',
      x = max_sr$Risk,
      y = max_sr$Return,
      label = "Tangency Portfolio",
```

```
        vjust = -1) +
    # Droite tangente (Capital Market Line)
    geom_segment(aes(x = 0,
                     y = rf_return,
                     # xend = max(portfolio_values$Risk),
                     xend = max_risk,
                     # yend = rf_return + (max_sr$Return - rf_return) / max_sr$Risk * max(portfolio_values$Risk)),
                     yend = rf_return + (max_sr$Return - rf_return) / max_sr$Risk * max_risk),
                 color = "red",
                 linetype = "dashed")

  # Création du graph 3 D
  # Calcul de la pente de la droite tangente
  slope <- (max_sr$Return - rf_return) / max_sr$Risk

  # Création des grilles pour la surface
  # Dans la fonction generate_plots()
  max_risk <- max(portfolio_values[["Risk"]])
  risk_grid <- seq(0, max_risk, length.out = 50)
  esg_grid <- seq(min(portfolio_values[["ESG"]]), max(portfolio_values[["ESG"]]), length.out = 50)

  # Création de la matrice de rendements pour la surface tangente
  return_matrix <- outer(risk_grid, rep(1, length(esg_grid)), function(risk, esg) {
    rf_return + slope * risk
  })

  eff_frontier_3d <- plot_ly() %>%
    # Points des portefeuilles
```

```
add_trace(
  data = portfolio_values,
  type = 'scatter3d',
  mode = 'markers',
  x = ~ESG,      # Axe horizontal
  y = ~Risk,     # Axe profondeur
  z = ~Return,   # Axe vertical
  marker = list(
    size = 3,
    color = ~SharpeRatio,
    colorscale = 'Viridis'
  ),
  name = "Portfolios"
) %>%
# Capital Market Line comme une surface plane
add_surface(
  x = matrix(rep(esg_grid, length(risk_grid)), nrow = length(risk_grid)),
  y = matrix(rep(risk_grid, each = length(esg_grid)), nrow = length(risk_grid)),
  z = matrix(rep(return_matrix, each = length(esg_grid)), nrow = length(risk_grid)),
  opacity = 0.4,
  colorscale = list(c(0, 1), c("red", "red")),
  showscale = FALSE,
  showlegend = TRUE,
  name = "Tangent Surface"
) %>%
# Point du minimum de variance
add_trace(
  x = min_var[["ESG"]],
  y = min_var[["Risk"]],
```

```
  z = min_var[["Return"]],
  type = "scatter3d",
  mode = "markers",
  marker = list(
    size = 8,
    color = "red",
    symbol = "square"
  ),
  name = "Minimum Variance Portfolio"
) %>%
# Point du portefeuille tangent
add_trace(
  x = max_sr[["ESG"]],
  y = max_sr[["Risk"]],
  z = max_sr[["Return"]],
  type = "scatter3d",
  mode = "markers",
  marker = list(
    size = 8,
    color = "red",
    symbol = "triangle-up"
  ),
  name = "Tangency Portfolio"
) %>%
layout(
  scene = list(
    camera = list(
      eye = list(x = 1.5, y = 1.5, z = 1.5),
      center = list(x = 0, y = 0, z = 0)
```

```
      ),
      xaxis = list(
       title = "ESG Score"
      ),
      yaxis = list(
       title = "Risk",
       tickformat = ".1%"
      ),
      zaxis = list(
       title = "Returns",
       tickformat = ".1%"
      ),
      aspectmode = "manual",
      aspectratio = list(x = 1, y = 0.8, z = 1)
    ),
    title = "Portfolio Optimization & Efficient Frontier (3D)"
  )

 return(list(min_var_plot = ggplotly(min_var_plot),
          max_sr_plot = ggplotly(max_sr_plot),
          eff_frontier_plot = ggplotly(eff_frontier_plot),
          eff_frontier_2d = ggplotly(eff_frontier_plot),
          eff_frontier_3d = eff_frontier_3d))
}

plots <- generate_plots()

# Ajouter la fonction calculate_real_data
calculate_real_data <- function(data_prep, user_weights, optimal_w) {
```

```
tickers_to_analyze <- data_prep$tickers
log_ret_tidy <- data_prep$ret_tidy
log_ret_xts <- data_prep$ret_xts

if (is.null(user_weights)) {
 wts <- as.numeric(optimal_w[1,])
 wts_tbl <- tibble(symbol = tickers_to_analyze,
                   wts = wts)
 # Merge data
 real_data <- dplyr::left_join(log_ret_tidy, wts_tbl, by = 'symbol')

 real_data <- real_data %>%
  dplyr::mutate(wt_return = wts * ret)

 ptf_ret <- real_data %>%
  dplyr::group_by(date) %>%
  dplyr::summarise(Portfolio_Return = sum(wt_return))

} else {
 wts <- user_weights
 wts_ <- as.numeric(optimal_w[1,])
 wts_tbl <- tibble(symbol = tickers_to_analyze,
                   wts = wts, #user weights
                   opt_wts = wts_ #optimal weight
 )

 # Merge data
 real_data <- dplyr::left_join(log_ret_tidy, wts_tbl, by = 'symbol')
```

```
 real_data <- real_data %>%
  dplyr::mutate(wt_return = wts * ret,
                opt_wt_return = opt_wts * ret)

 ptf_ret <- real_data %>%
  dplyr::group_by(date) %>%
  dplyr::summarise(Portfolio_Return = sum(wt_return),
                   Opt_Portfolio_Return = sum(opt_wt_return))
}

# Créer sav_real_dt avec l'index des dates
sav_real_dt <- data.frame(
 date = zoo::index(log_ret_xts),
 as.data.frame(log_ret_xts)
)

# Merge data
real_data <- dplyr::left_join(sav_real_dt, ptf_ret, by = 'date')

# Calcul du risque et du Sharpe ratio
width <- switch(type_return,
                "daily" = 252,
                "weekly" = 52,
                "monthly" = 12,
                "yearly" = 1)

# Calcul de la volatilité avec une fenêtre mobile
window <- 5
portfolio_returns <- real_data$Portfolio_Return
```

```
  portfolio_risk <- numeric(length(portfolio_returns))

  for (i in seq(window, length(portfolio_returns))) {
    portfolio_risk[i] <- sd(portfolio_returns[(i - window + 1):i], na.rm = TRUE)
  }

  portfolio_risk[is.na(portfolio_risk)] <- 0

  real_data$Portfolio_Risk <- portfolio_risk
  real_data$Sharpe_Ratio <- (real_data$Portfolio_Return - rf_return) / real_data$Portfolio_Risk

  return(real_data)
 }

 # Return results
 list(
  table = calculate_real_data(data_prep, user_weights, optimal_w),
  table_simul = portfolio_values,
  Optimal_weight = optimal_w,
  min_var_port_w = plots$min_var_plot,
  max_var_port_w = plots$max_sr_plot,
  efficient_frontier_2d = plots$eff_frontier_plot,
  efficient_frontier_3d = plots$eff_frontier_3d,
  esg_scores = data_prep$esg
 )
}



# Version du 22.01.2025
```

```
# best_analyze_pf_perf <- function(data,
#                         rf = 0.02,
#                         type_return = 'daily',
#                         user_weights = NULL,
#                         num_simulations = 500,
#                         type_simulation = 'uniform') {
#
#   # Validation des entrées
#   validate_inputs <- function() {
#     required_cols <- c("Ticker", "Date", "Return", "Volatility", "ESG")
#     if (!all(required_cols %in% colnames(data))) {
#       stop("Missing required columns: ",
#           paste(setdiff(required_cols, colnames(data)), collapse = ", "))
#     }
#
#     if (!is.numeric(rf)) stop("rf must be numeric")
#
#     valid_types <- c('daily', 'weekly', 'monthly', 'yearly')
#     if (!type_return %in% valid_types) {
#       stop("type_return must be one of: ", paste(valid_types, collapse = ", "))
#     }
#   }
#
#   validate_inputs()
#
#   # Constants
#   ANNUALIZATION_FACTORS <- c(daily = 252, weekly = 52, monthly = 12, yearly = 1)
#   annualization_factor <- ANNUALIZATION_FACTORS[[type_return]]
#   rf_return <- rf / annualization_factor
```

```
#
#  # Data preparation
#  prepare_data <- function() {
#   tickers_to_analyze <- unique(data$Ticker)
#
#   log_ret_tidy <- data %>%
#    dplyr::select(date = Date, symbol = Ticker, ret = Return, esg = ESG) %>%
#    dplyr::arrange(date, symbol)
#
#   log_ret_xts <- log_ret_tidy %>%
#    dplyr::select(date, symbol, ret) %>%
#    tidyr::spread(symbol, ret) %>%
#    timetk::tk_xts(date_var = "date")
#
#   esg_scores <- data %>%
#    dplyr::group_by(Ticker) %>%
#    dplyr::summarise(ESG = mean(ESG)) %>%
#    dplyr::pull(ESG, name = Ticker)
#
#   list(
#    tickers = tickers_to_analyze,
#    ret_tidy = log_ret_tidy,
#    ret_xts = log_ret_xts,
#    esg = esg_scores
#   )
#  }
#
#  data_prep <- prepare_data()
#
```

```
#  # Portfolio calculations
#  calculate_portfolio_metrics <- function(weights, mean_returns, cov_matrix, esg) {
#    port_ret <- sum(weights * mean_returns)
#    port_ret <- ((port_ret + 1)^annualization_factor) - 1
#    port_sd <- sqrt(as.numeric(t(weights) %*% (cov_matrix %*% weights)))
#    sharpe <- (port_ret - rf_return) / port_sd
#    esg_score <- sum(weights * esg)
#
#    c(Return = port_ret, Risk = port_sd, SharpeRatio = sharpe, ESG = esg_score)
#  }
#
#  # Generate weights
#  generate_weights <- function(n_assets, method) {
#    weights <- switch(method,
#                  "uniform" = runif(n_assets),
#                  "montecarlo" = abs(rnorm(n_assets)),
#                  "dirichlet" = rgamma(n_assets, shape = 1, scale = 1),
#                  stop("Invalid simulation type"))
#    weights / sum(weights)
#  }
#
#  # Main calculations
#  mean_ret <- colMeans(data_prep$ret_xts)
#  cov_mat <- cov(data_prep$ret_xts) * annualization_factor
#  n_assets <- length(data_prep$tickers)
#
#  # Normalize user weights if provided
#  if (!is.null(user_weights)) {
#    user_weights <- if(sum(user_weights) == 0) {
```

```
#     rep(1/n_assets, n_assets)
#   } else if(sum(user_weights) > 0 && sum(user_weights) <= 1) {
#     user_weights
#   } else if(sum(user_weights) > 1 && sum(user_weights) <= 100) {
#     user_weights/100
#   } else {
#     user_weights / sum(user_weights)
#   }
# }
#
# # Simulate portfolios
# weights_matrix <- matrix(
#   replicate(num_simulations, generate_weights(n_assets, type_simulation)),
#   nrow = num_simulations, byrow = TRUE
# )
#
# if (!is.null(user_weights)) {
#   weights_matrix[1,] <- user_weights
# }
#
# # Calculate portfolio metrics for all simulations
# portfolio_metrics <- t(apply(weights_matrix, 1, calculate_portfolio_metrics,
#                              mean_ret, cov_mat, data_prep$esg))
#
# # Create results tibble
# portfolio_values <- as_tibble(cbind(
#   weights_matrix,
#   portfolio_metrics
# ))
```

```
#
#  colnames(portfolio_values)[1:n_assets] <- data_prep$tickers
#
#  # Find optimal portfolios
#  min_var <- portfolio_values[which.min(portfolio_values$Risk),]
#  print("head(min_var)")
#  print(head(min_var))
#  max_sr <- portfolio_values[which.max(portfolio_values$SharpeRatio),]
#  print("head(max_sr)")
#  print(head(max_sr))
#  optimal_w <- portfolio_values[which.max(portfolio_values$Return),][,1:n_assets]
#
#  # Generate plots
#  # Création des graphiques
#  # [Code des graphiques inchangé, mais ajout de l'ESG dans les visualisations]
#
#  # Création du graph 3 D
#  # Calcul de la pente de la droite tangente
#  slope <- (max_sr$Return - rf_return) / max_sr$Risk
#
#  # Création des grilles pour la surface
#  # Important: utiliser les mêmes plages que dans le graphique 2D
#  risk_grid <- seq(0, max(portfolio_values$Risk), length.out = 50)
#  esg_grid <- seq(min(portfolio_values$ESG), max(portfolio_values$ESG), length.out = 50)
#
#  # Création de la matrice de rendements pour la surface tangente
#  # Utiliser exactement la même équation que dans le graphique 2D
#  return_matrix <- matrix(
#    rf_return + slope * risk_grid,
```

```
#    nrow = length(risk_grid),
#    ncol = length(esg_grid),
#    byrow = FALSE
#  )
#
#
#  # Graphique de la frontière efficiente en 3D avec ESG
#  eff_frontier_3d_test <- plot_ly() %>%
#    # Points des portefeuilles
#    add_trace(
#      data = portfolio_values,
#      type = 'scatter3d',
#      mode = 'markers',
#      x = ~Risk,
#      y = ~Return,
#      z = ~ESG,
#      marker = list(
#        size = 3,
#        color = ~SharpeRatio,
#        colorscale = 'Viridis'
#      ),
#      name = "Portfolios"
#    ) %>%
#    # Surface tangente
#    add_surface(
#      x = risk_grid,
#      y = return_matrix[,1],
#      z = matrix(rep(esg_grid, each = length(risk_grid)),
#                 nrow = length(risk_grid),
```

```
#             ncol = length(esg_grid)),
#     opacity = 0.3,
#     colorscale = list(c(0,1), c("red", "red")),
#     showscale = FALSE,
#     name = "Tangent Surface"
#   ) %>%
#   # Point du minimum de variance
#   add_trace(
#     x = min_var$Risk,
#     y = min_var$Return,
#     z = min_var$ESG,
#     type = "scatter3d",
#     mode = "markers",
#     marker = list(
#       size = 8,
#       color = "red",
#       symbol = "square"  # Correspond au shape=22 du graphique 2D
#     ),
#     name = "Minimum Variance Portfolio"
#   ) %>%
#   # Point du portefeuille tangent
#   add_trace(
#     x = max_sr$Risk,
#     y = max_sr$Return,
#     z = max_sr$ESG,
#     type = "scatter3d",
#     mode = "markers",
#     marker = list(
#       size = 8,
```

```
#       color = "red",
#       symbol = "triangle-up"  # Correspond au shape=24 du graphique 2D
#     ),
#     name = "Tangency Portfolio"
#   ) %>%
#   layout(
#     scene = list(
#       camera = list(
#         eye = list(x = 1.5, y = 1.5, z = 1.5),
#         center = list(x = 0, y = 0, z = 0)
#       ),
#       xaxis = list(
#         title = "Risk",
#         tickformat = ".1%"
#       ),
#       yaxis = list(
#         title = "Returns",
#         tickformat = ".1%"
#       ),
#       zaxis = list(title = "ESG Score"),
#       aspectmode = "manual",
#       aspectratio = list(x = 1, y = 1, z = 0.8)
#     ),
#     title = "Portfolio Optimization & Efficient Frontier (3D)"
#   )
#
#
#  generate_plots <- function() {
#    # Graphiques
```

```
#    min_var_plot <- min_var %>%
#     dplyr::select(-ESG)%>%
#     gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%
#     ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +
#     geom_bar(stat = "identity") +
#     theme_minimal() +
#     labs(x = 'Assets', y = 'Weights', title = "Minimum variance portfolio Weights") +
#     scale_y_continuous(labels = scales::percent)
#
#    max_sr_plot <- max_sr %>%
#     dplyr::select(-ESG)%>%
#     gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%
#     ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +
#     geom_bar(stat = "identity") +
#     theme_minimal() +
#     labs(x = 'Assets', y = 'Weights', title = "Tangency portfolio weights") +
#     scale_y_continuous(labels = scales::percent)
#
#    # eff_frontier_plot <- portfolio_values %>%
#    #   ggplot(aes(x = Risk, y = Return, color = SharpeRatio)) +
#    #   geom_point() +
#    #   theme_classic() +
#    #   scale_y_continuous(labels = scales::percent) +
#    #   scale_x_continuous(labels = scales::percent) +
#    #    labs(x = 'Annualized Risk', y = 'Annualized Returns', title = "Portfolio Optimization & Efficient
Frontier") +
#    #   geom_point(aes(x = Risk, y = Return #, size = 3
#    #   ), shape = 22, fill = "red", data = min_var) +  # Point carré
#    #   geom_point(aes(x = Risk, y = Return #, size = 3
```

```
#   #  ), shape = 24, fill = "red", data = max_sr) +  # Point triangle
#   #  annotate('text', x = min_var$Risk, y = min_var$Return, label = "Minimum variance portfolio") +
#   #  annotate('text', x = max_sr$Risk, y = max_sr$Return, label = "Tangency Portfolio")+
#   #  # Ajout de la droite tangente
#   #  geom_abline(
#   #    intercept = rf_return,
#   #    slope = (max_sr$Return - rf_return) / max_sr$Risk,
#   #    color = "red",
#   #    linetype = "dashed"
#   #  )
#
#   eff_frontier_plot <- portfolio_values %>%
#    ggplot(aes(x = Risk, y = Return, color = SharpeRatio)) +
#    geom_point() +
#    theme_classic() +
#    scale_y_continuous(labels = scales::percent) +
#    scale_x_continuous(labels = scales::percent) +
#    labs(x = 'Annualized Risk', y = 'Annualized Returns',
#         title = "Portfolio Optimization & Efficient Frontier") +
#    # Point pour le portefeuille à variance minimale
#    geom_point(data = min_var,
#               aes(x = Risk, y = Return),
#               shape = 22,
#               fill = "red",
#               size = 3) +
#    # Point pour le portefeuille tangent
#    geom_point(data = max_sr,
#               aes(x = Risk, y = Return),
#               # shape = 24,
```

```
#             shape = 25,
#             fill = "red",
#             size = 4) +
#     # Annotations
#     annotate('text',
#           x = min_var$Risk,
#           y = min_var$Return,
#           label = "Minimum variance portfolio",
#           vjust = -1) +
#     annotate('text',
#           x = max_sr$Risk,
#           y = max_sr$Return,
#           label = "Tangency Portfolio",
#           vjust = -1) +
#     # Droite tangente (Capital Market Line)
#     geom_segment(aes(x = 0,
#                 y = rf_return,
#                 xend = max(portfolio_values$Risk),
#               yend = rf_return + (max_sr$Return - rf_return) / max_sr$Risk * max(portfolio_values$Risk)),
#               color = "red",
#               linetype = "dashed")
#
#
#   # Création du graph 3 D
#   # Calcul de la pente de la droite tangente
#   slope <- (max_sr$Return - rf_return) / max_sr$Risk
#
#   # Création des grilles pour la surface
#   # Important: utiliser les mêmes plages que dans le graphique 2D
```

```
#   risk_grid <- seq(0, max(portfolio_values$Risk), length.out = 50)
#   esg_grid <- seq(min(portfolio_values$ESG), max(portfolio_values$ESG), length.out = 50)
#
#   # Création de la matrice de rendements pour la surface tangente
#   # Utiliser exactement la même équation que dans le graphique 2D
#   return_matrix <- matrix(
#     rf_return + slope * risk_grid,
#     nrow = length(risk_grid),
#     ncol = length(esg_grid),
#     byrow = FALSE
#   )
#
#
#   # # Graphique de la frontière efficiente en 3D avec ESG
#   eff_frontier_3d <- plot_ly() %>%
#     # Points des portefeuilles
#     add_trace(
#       data = portfolio_values,
#       type = 'scatter3d',
#       mode = 'markers',
#       x = ~Risk,
#       y = ~Return,
#       z = ~ESG,
#       marker = list(
#         size = 3,
#         color = ~SharpeRatio,
#         colorscale = 'Viridis'
#       ),
#       name = "Portfolios"
```

```
#     ) %>%
#     # Surface tangente
#     add_surface(
#       x = risk_grid,
#       y = return_matrix[,1],
#       z = matrix(rep(esg_grid, each = length(risk_grid)),
#                  nrow = length(risk_grid),
#                  ncol = length(esg_grid)),
#       opacity = 0.3,
#       colorscale = list(c(0,1), c("red", "red")),
#       showscale = FALSE,
#       name = "Tangent Surface"
#     ) %>%
#     # Point du minimum de variance
#     add_trace(
#       x = min_var$Risk,
#       y = min_var$Return,
#       z = min_var$ESG,
#       type = "scatter3d",
#       mode = "markers",
#       marker = list(
#         size = 8,
#         color = "red",
#         symbol = "square"  # Correspond au shape=22 du graphique 2D
#       ),
#       name = "Minimum Variance Portfolio"
#     ) %>%
#     # Point du portefeuille tangent
#     add_trace(
```

```
#       x = max_sr$Risk,
#       y = max_sr$Return,
#       z = max_sr$ESG,
#       type = "scatter3d",
#       mode = "markers",
#       marker = list(
#         size = 8,
#         color = "red",
#         symbol = "triangle-up"  # Correspond au shape=24 du graphique 2D
#       ),
#       name = "Tangency Portfolio"
#     ) %>%
#     layout(
#       scene = list(
#         camera = list(
#           eye = list(x = 1.5, y = 1.5, z = 1.5),
#           center = list(x = 0, y = 0, z = 0)
#         ),
#         xaxis = list(
#           title = "Risk",
#           tickformat = ".1%"
#         ),
#         yaxis = list(
#           title = "Returns",
#           tickformat = ".1%"
#         ),
#         zaxis = list(title = "ESG Score"),
#         aspectmode = "manual",
#         aspectratio = list(x = 1, y = 1, z = 0.8)
```

```
#       ),
#       title = "Portfolio Optimization & Efficient Frontier (3D)"
#     )
#
#    # # Nouvelle version
#    # # Graphique de la frontière efficiente en 3D avec ESG
#    # eff_frontier_3d <- plot_ly() %>%
#    #   # Points des portefeuilles
#    #   add_trace(
#    #     data = portfolio_values,
#    #     type = 'scatter3d',
#    #     mode = 'markers',
#    #     x = ~Risk,
#    #     y = ~Return,
#    #     z = ~ESG,
#    #     marker = list(
#    #       size = 3,
#    #       color = ~SharpeRatio,
#    #       colorscale = 'Viridis'
#    #     ),
#    #     name = "Portfolios"
#    #   ) %>%
#    #   # Surface tangente modifiée
#    #   add_surface(
#    #     x = c(0, max(portfolio_values$Risk)),  # Début à 0 pour l'axe x
#           #           y = c(rf_return, rf_return + (max_sr$Return - rf_return) / max_sr$Risk *
max(portfolio_values$Risk)),  # Points de la CML
#    #     # z = matrix(rep(seq(min(portfolio_values$ESG),
#    #     #                max(portfolio_values$ESG),
```

```
#   #   #               length.out = 50),
#   #   #            each = 2),
#   #   #         nrow = 2,
#   #   #         ncol = 50),
#   #   z = matrix(rep(esg_grid, each = length(risk_grid)),
#   #           nrow = length(risk_grid),
#   #           ncol = length(esg_grid)),
#   #   opacity = 0.3,
#   #   colorscale = list(c(0,1), c("red", "red")),
#   #   showscale = FALSE,
#   #   name = "Capital Market Line Surface"
#   # ) %>%
#   # # Point du minimum de variance
#   # add_trace(
#   #   x = min_var$Risk,
#   #   y = min_var$Return,
#   #   z = min_var$ESG,
#   #   type = "scatter3d",
#   #   mode = "markers",
#   #   marker = list(
#   #     size = 8,
#   #     color = "red",
#   #     symbol = "square"
#   #   ),
#   #   name = "Minimum Variance Portfolio"
#   # ) %>%
#   # # Point du portefeuille tangent
#   # add_trace(
#   #   x = max_sr$Risk,
```

```
#   #   y = max_sr$Return,
#   #   z = max_sr$ESG,
#   #   type = "scatter3d",
#   #   mode = "markers",
#   #   marker = list(
#   #     size = 8,
#   #     color = "red",
#   #     symbol = "triangle-up"
#   #   ),
#   #   name = "Tangency Portfolio"
#   # ) %>%
#   # layout(
#   #   scene = list(
#   #     camera = list(
#   #       eye = list(x = 1.5, y = 1.5, z = 1.5),
#   #       center = list(x = 0, y = 0, z = 0)
#   #     ),
#   #     xaxis = list(
#   #       title = "Risk",
#   #       tickformat = ".1%"
#   #     ),
#   #     yaxis = list(
#   #       title = "Returns",
#   #       tickformat = ".1%"
#   #     ),
#   #     zaxis = list(title = "ESG Score"),
#   #     aspectmode = "manual",
#   #     aspectratio = list(x = 1, y = 1, z = 0.8)
#   #   ),
```

```
#   #    title = "Portfolio Optimization & Efficient Frontier (3D)"
#   #  )
#
#   return(list(min_var_plot = ggplotly(min_var_plot),
#               max_sr_plot = ggplotly(max_sr_plot),
#               eff_frontier_plot = ggplotly(eff_frontier_plot),
#               eff_frontier_2d = ggplotly(eff_frontier_plot),
#               eff_frontier_3d = eff_frontier_3d))
# }
#
# plots <- generate_plots()
#
# # Ajouter la fonction calculate_real_data
# calculate_real_data <- function(data_prep, user_weights, optimal_w) {
#   tickers_to_analyze <- data_prep$tickers
#   log_ret_tidy <- data_prep$ret_tidy
#   log_ret_xts <- data_prep$ret_xts
#
#   if (is.null(user_weights)) {
#     wts <- as.numeric(optimal_w[1,])
#     wts_tbl <- tibble(symbol = tickers_to_analyze,
#                       wts = wts)
#     # Merge data
#     real_data <- dplyr::left_join(log_ret_tidy, wts_tbl, by = 'symbol')
#
#     real_data <- real_data %>%
#       dplyr::mutate(wt_return = wts * ret)
#
#     ptf_ret <- real_data %>%
```

```
#       dplyr::group_by(date) %>%
#       dplyr::summarise(Portfolio_Return = sum(wt_return))
#
#   } else {
#     wts <- user_weights
#     wts_ <- as.numeric(optimal_w[1,])
#     wts_tbl <- tibble(symbol = tickers_to_analyze,
#                       wts = wts, #user weights
#                       opt_wts = wts_ #optimal weight
#     )
#
#     # Merge data
#     real_data <- dplyr::left_join(log_ret_tidy, wts_tbl, by = 'symbol')
#
#     real_data <- real_data %>%
#       dplyr::mutate(wt_return = wts * ret,
#                     opt_wt_return = opt_wts * ret)
#
#     ptf_ret <- real_data %>%
#       dplyr::group_by(date) %>%
#       dplyr::summarise(Portfolio_Return = sum(wt_return),
#                        Opt_Portfolio_Return = sum(opt_wt_return))
#   }
#
#   # Créer sav_real_dt avec l'index des dates
#   sav_real_dt <- data.frame(
#     date = zoo::index(log_ret_xts),
#     as.data.frame(log_ret_xts)
#   )
```

```
#
#   # Merge data
#   real_data <- dplyr::left_join(sav_real_dt, ptf_ret, by = 'date')
#
#   # Calcul du risque et du Sharpe ratio
#   width <- switch(type_return,
#               "daily" = 252,
#               "weekly" = 52,
#               "monthly" = 12,
#               "yearly" = 1)
#
#   # Calcul de la volatilité avec une fenêtre mobile
#   window <- 5
#   portfolio_returns <- real_data$Portfolio_Return
#   portfolio_risk <- numeric(length(portfolio_returns))
#
#   for (i in seq(window, length(portfolio_returns))) {
#     portfolio_risk[i] <- sd(portfolio_returns[(i - window + 1):i], na.rm = TRUE)
#   }
#
#   portfolio_risk[is.na(portfolio_risk)] <- 0
#
#   real_data$Portfolio_Risk <- portfolio_risk
#   real_data$Sharpe_Ratio <- (real_data$Portfolio_Return - rf_return) / real_data$Portfolio_Risk
#
#   return(real_data)
# }
#
#  # Return results
```

```
#   list(
#     table = calculate_real_data(data_prep, user_weights, optimal_w),
#     table_simul = portfolio_values,
#     Optimal_weight = optimal_w,
#     min_var_port_w = plots$min_var_plot,
#     max_var_port_w = plots$max_sr_plot,
#     # pf_opt_eff_front = plots$eff_frontier_plot,
#     efficient_frontier_2d = plots$eff_frontier_plot,
#     efficient_frontier_3d = plots$eff_frontier_3d,
#     efficient_frontier_3d.b = eff_frontier_3d_test,
#     esg_scores = data_prep$esg
#   )
# }

## Nouvelle fonction qui fonnctionne bien mais un peu lente
## Le nom 'new_best_analyze_pf_perf' a été changée en 'best_analyze_pf_perf'
# best_analyze_pf_perf <- function(data,
#                                  rf,
#                                  type_return = 'daily',
#                                  user_weights = NULL,
#                                  num_simulations = 500,
#                                  type_simulation = 'uniform') {
#
#   # Vérification des colonnes requises
#   required_cols <- c("Ticker", "Date", "Return", "Volatility", "ESG")
#   if (!all(required_cols %in% colnames(data))) {
#     stop("La dataframe doit contenir les colonnes: ", paste(required_cols, collapse = ", "))
#   }
#
```

```
#  # Obtenir la liste des tickers uniques
#  tickers_to_analyze <- unique(data$Ticker)
#
#  # Préparation des données pour le calcul des rendements
#  log_ret_tidy <- data %>%
#    dplyr::select(date = Date, symbol = Ticker, ret = Return, esg = ESG) %>%
#    dplyr::arrange(date, symbol)
#
#  # Création de la matrice de rendements
#  log_ret_xts <- log_ret_tidy %>%
#    dplyr::select(date, symbol, ret) %>%
#    tidyr::spread(symbol, ret) %>%
#    timetk::tk_xts(date_var = "date")
#
#  # Création de la matrice ESG
#  # ici ESG_score moyen
#  esg_scores <- data %>%
#    dplyr::group_by(Ticker) %>%
#    dplyr::summarise(ESG = mean(ESG)) %>%
#    dplyr::pull(ESG, name = Ticker)
#
#  mean_ret <- colMeans(log_ret_xts)
#
#  annualization_factor <- switch(type_return,
#                                 "daily" = 252,
#                                 "weekly" = 52,
#                                 "monthly" = 12,
#                                 "yearly" = 1,
#                                 stop("type_return doit être 'daily', 'weekly', 'monthly' ou 'yearly'"))
```

```
#
#  cov_mat <- cov(log_ret_xts) * annualization_factor
#
#  # Traitement du rf
#  if (is.numeric(rf)) {
#   rf_return <- rf / annualization_factor
#  } else {
#   stop("rf doit être un taux numérique")
#  }
#
#  # Normalisation des poids utilisateur
#  if (!is.null(user_weights)) {
#   if(sum(user_weights) == 0) {
#    user_weights = rep(1/length(tickers_to_analyze), length(tickers_to_analyze))
#
#   } else if(sum(user_weights) > 0 && sum(user_weights) <= 1){
#    user_weights = user_weights
#
#   } else if(sum(user_weights) > 1 && sum(user_weights) <= 100 ){
#    user_weights = user_weights/100
#
#   } else {
#    user_weights <- user_weights / sum(user_weights)
#   }
#  }
#
#  # Initialisation des matrices pour les simulations
#  all_wts <- matrix(nrow = num_simulations, ncol = length(tickers_to_analyze))
#  port_returns <- vector('numeric', length = num_simulations)
```

```
#  port_risk <- vector('numeric', length = num_simulations)
#  sharpe_ratio <- vector('numeric', length = num_simulations)
#  port_esg <- vector('numeric', length = num_simulations)
#
#  # Simulation des portefeuilles
#  for (i in 1:num_simulations) {
#   if (!is.null(user_weights) && i == 1) {
#    wts <- user_weights
#   } else {
#    wts <- switch(type_simulation,
#              "uniform" = runif(length(tickers_to_analyze)),
#              "montecarlo" = abs(rnorm(length(tickers_to_analyze))),
#              "dirichlet" = rgamma(length(tickers_to_analyze), shape = 1, scale = 1),
#              stop("type_simulation invalide"))
#
#    wts <- wts / sum(wts)
#   }
#
#   all_wts[i,] <- wts
#
#   # Calcul des métriques du portefeuille
#   port_ret <- sum(wts * mean_ret)
#   port_ret <- ((port_ret + 1)^annualization_factor) - 1
#   port_returns[i] <- port_ret
#
#   port_sd <- sqrt(t(wts) %*% (cov_mat %*% wts))
#   port_risk[i] <- port_sd
#
#   sharpe_ratio[i] <- (port_ret - rf_return) / port_sd
```

```
#
#   # Calcul du score ESG du portefeuille
#   port_esg[i] <- sum(wts * esg_scores)
# }
#
# # Création du tableau des résultats
# portfolio_values <- tibble(
#   Return = port_returns,
#   Risk = port_risk,
#   SharpeRatio = sharpe_ratio,
#   ESG = port_esg
# )
#
# # Avant la création du tibble, s'assurer que les noms de colonnes sont uniques
# all_wts <- as_tibble(all_wts, .name_repair = "unique") %>%
#   setNames(tickers_to_analyze)
#
# # colnames(all_wts) <- tickers_to_analyze
# portfolio_values <- bind_cols(all_wts, portfolio_values)
#
# # Identification des portefeuilles optimaux
# min_var <- portfolio_values[which.min(portfolio_values$Risk),]
# max_sr <- portfolio_values[which.max(portfolio_values$SharpeRatio),]
# max_esg <- portfolio_values[which.max(portfolio_values$ESG),]
#
# if (is.null(user_weights)) {
#   user_weights <- runif(length(tickers_to_analyze))
#   user_weights <- user_weights / sum(user_weights)
#
```

```
#  }
#
#  real_port_ret <- sum(user_weights * mean_ret)
#
#  real_port_ret <- ((real_port_ret + 1)^annualization_factor) - 1
#
#  real_port_risk <- sqrt(t(user_weights) %*% (cov_mat %*% user_weights))
#
#  real_sharpe_ratio <- (real_port_ret - rf_return) / real_port_risk
#
#                                                                    optimal_w                    <-
portfolio_values[which.max(portfolio_values$Return),][,c(1:length(tickers_to_analyze))]
#
#  # Pour obtenir la table (ie real_data)
#  if (is.null(user_weights)) {
#   wts <- as.numeric(optimal_w[1,])
#   wts_tbl <- tibble(symbol = tickers_to_analyze,
#                     wts = wts)
#   # Merge data
#   real_data <- dplyr::left_join(log_ret_tidy, wts_tbl, by = 'symbol')
#
#   real_data <- real_data %>%
#     dplyr::mutate(wt_return = wts * ret)
#
#   ptf_ret <- real_data %>%
#     group_by(date) %>%
#     summarise(Portfolio_Return = sum(wt_return))
#
#  } else {
```

```
#   wts <- user_weights
#   wts_ <- as.numeric(optimal_w[1,])
#   wts_tbl <- tibble(symbol = tickers_to_analyze,
#                     wts = wts, #user weights
#                     opt_wts = wts_ #optimal weight
#   )
#
#   # Merge data
#   real_data <- dplyr::left_join(log_ret_tidy, wts_tbl, by = 'symbol')
#
#   real_data <- real_data %>%
#     dplyr::mutate(wt_return = wts * ret,
#                   opt_wt_return = opt_wts * ret)
#
#   ptf_ret <- real_data %>%
#     dplyr::group_by(date) %>%
#     dplyr::summarise(Portfolio_Return = sum(wt_return),
#                      Opt_Portfolio_Return = sum(opt_wt_return))
#
# }
#
# sav_real_dt <- data.frame(date = index(log_ret_xts), coredata(log_ret_xts))
#
# # Merge data
# real_data <- dplyr::left_join(sav_real_dt, ptf_ret, by = 'date')
#
# # Calcul du risque et du Sharpe ratio pour les vraies données
# width <- switch(type_return,
#                 "daily" = 252,
```

```
#               "weekly" = 52,
#               "monthly" = 12,
#               "yearly" = 1)
#
#  real_data <- real_data %>%
#    dplyr::mutate(
#      Portfolio_Risk = rollapply(Portfolio_Return, width = width, FUN = sd, align = "right", fill = NA)
#      # ,
#      # Sharpe_Ratio = (Portfolio_Return - rf_return) / Portfolio_Risk
#    )
#
#  window = 5
#  portfolio_returns = real_data$Portfolio_Return
#  portfolio_risk = real_data$Portfolio_Risk
#  for (i in seq(window, nrow(real_data))) {
#    # Calcul de la volatilité sur une fenêtre mobile
#    portfolio_risk[i] <- sd(portfolio_returns[(i - window + 1):i], na.rm = TRUE)
#  }
#
#  portfolio_risk[which(is.na(portfolio_risk))] = 0
#
#  real_data$Portfolio_Risk = portfolio_risk
#
#  real_data <- real_data %>%
#    dplyr::mutate(
#      Sharpe_Ratio = (Portfolio_Return - rf_return) / Portfolio_Risk
#    )
#
#  # Création des graphiques
```

```
#  # [Code des graphiques inchangé, mais ajout de l'ESG dans les visualisations]
#  # Graphiques
#  min_var_plot <- min_var %>%
#   dplyr::select(-ESG)%>%
#   gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%
#   ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +
#   geom_bar(stat = "identity") +
#   theme_minimal() +
#   labs(x = 'Assets', y = 'Weights', title = "Minimum variance portfolio Weights") +
#   scale_y_continuous(labels = scales::percent)
#
#  max_sr_plot <- max_sr %>%
#   dplyr::select(-ESG)%>%
#   gather(key = "Asset", value = "Weights", -Return, -Risk, -SharpeRatio) %>%
#   ggplot(aes(x = fct_reorder(Asset, Weights), y = Weights, fill = Asset)) +
#   geom_bar(stat = "identity") +
#   theme_minimal() +
#   labs(x = 'Assets', y = 'Weights', title = "Tangency portfolio weights") +
#   scale_y_continuous(labels = scales::percent)
#
#  eff_frontier_plot <- portfolio_values %>%
#   ggplot(aes(x = Risk, y = Return, color = SharpeRatio)) +
#   geom_point() +
#   theme_classic() +
#   scale_y_continuous(labels = scales::percent) +
#   scale_x_continuous(labels = scales::percent) +
#     labs(x = 'Annualized Risk', y = 'Annualized Returns', title = "Portfolio Optimization & Efficient
Frontier") +
#   geom_point(aes(x = Risk, y = Return #, size = 3
```

```
#    ), shape = 22, fill = "red", data = min_var) +  # Point carré
#    geom_point(aes(x = Risk, y = Return #, size = 3
#    ), shape = 24, fill = "red", data = max_sr) +  # Point triangle
#    annotate('text', x = min_var$Risk, y = min_var$Return, label = "Minimum variance portfolio") +
#    annotate('text', x = max_sr$Risk, y = max_sr$Return, label = "Tangency Portfolio")+
#    # Ajout de la droite tangente
#    geom_abline(
#      intercept = rf_return,
#      slope = (max_sr$Return - rf_return) / max_sr$Risk,
#      color = "red",
#      linetype = "dashed"
#    )
#
#
#  # Création du graph 3 D
#  # Calcul de la pente de la droite tangente
#  slope <- (max_sr$Return - rf_return) / max_sr$Risk
#
#  # Création des grilles pour la surface
#  # Important: utiliser les mêmes plages que dans le graphique 2D
#  risk_grid <- seq(0, max(portfolio_values$Risk), length.out = 50)
#  esg_grid <- seq(min(portfolio_values$ESG), max(portfolio_values$ESG), length.out = 50)
#
#  # Création de la matrice de rendements pour la surface tangente
#  # Utiliser exactement la même équation que dans le graphique 2D
#  return_matrix <- matrix(
#    rf_return + slope * risk_grid,
#    nrow = length(risk_grid),
#    ncol = length(esg_grid),
```

```
#    byrow = FALSE
#  )
#
#
#  # Graphique de la frontière efficiente en 3D avec ESG
#  eff_frontier_3d <- plot_ly() %>%
#    # Points des portefeuilles
#    add_trace(
#      data = portfolio_values,
#      type = 'scatter3d',
#      mode = 'markers',
#      x = ~Risk,
#      y = ~Return,
#      z = ~ESG,
#      marker = list(
#        size = 3,
#        color = ~SharpeRatio,
#        colorscale = 'Viridis'
#      ),
#      name = "Portfolios"
#    ) %>%
#    # Surface tangente
#    add_surface(
#      x = risk_grid,
#      y = return_matrix[,1],
#      z = matrix(rep(esg_grid, each = length(risk_grid)),
#                 nrow = length(risk_grid),
#                 ncol = length(esg_grid)),
#      opacity = 0.3,
```

```
#     colorscale = list(c(0,1), c("red", "red")),
#     showscale = FALSE,
#     name = "Tangent Surface"
#   ) %>%
#   # Point du minimum de variance
#   add_trace(
#     x = min_var$Risk,
#     y = min_var$Return,
#     z = min_var$ESG,
#     type = "scatter3d",
#     mode = "markers",
#     marker = list(
#       size = 8,
#       color = "red",
#       symbol = "square"  # Correspond au shape=22 du graphique 2D
#     ),
#     name = "Minimum Variance Portfolio"
#   ) %>%
#   # Point du portefeuille tangent
#   add_trace(
#     x = max_sr$Risk,
#     y = max_sr$Return,
#     z = max_sr$ESG,
#     type = "scatter3d",
#     mode = "markers",
#     marker = list(
#       size = 8,
#       color = "red",
#       symbol = "triangle-up"  # Correspond au shape=24 du graphique 2D
```

```
#     ),
#     name = "Tangency Portfolio"
#   ) %>%
#   layout(
#     scene = list(
#       camera = list(
#         eye = list(x = 1.5, y = 1.5, z = 1.5),
#         center = list(x = 0, y = 0, z = 0)
#       ),
#       xaxis = list(
#         title = "Risk",
#         tickformat = ".1%"
#       ),
#       yaxis = list(
#         title = "Returns",
#         tickformat = ".1%"
#       ),
#       zaxis = list(title = "ESG Score"),
#       aspectmode = "manual",
#       aspectratio = list(x = 1, y = 1, z = 0.8)
#     ),
#     title = "Portfolio Optimization & Efficient Frontier (3D)"
#   )
#
#  # Retour des résultats
#  return(list(
#    table = real_data,
#    table_simul = portfolio_values,
#    Optimal_weight = optimal_w,
```

```
#    min_var_port_w = ggplotly(min_var_plot),
#    max_var_port_w = ggplotly(max_sr_plot),
#    # max_esg_portfolio = max_esg,
#    pf_opt_eff_front = ggplotly(eff_frontier_plot), #2d plot
#    efficient_frontier_2d = ggplotly(eff_frontier_plot), #2d plot
#    efficient_frontier_3d = eff_frontier_3d,
#    esg_scores = esg_scores
#  ))
# }



risk_free_rate = c("OAT 1y",
        'Bund 1y',
        'T-Note 1y',
        "OAT 5y",
        'Bund 5y',
        'T-Note 5y',
        "OAT 10y",
        'Bund 10y',
        'T-Note 10y'
)


#         https://github.com/t-emery/sais-susfin_data/blob/main/datasets/etf_comparison-2022-10-03.csv
#         esg_etf_comparison_url <- "https://raw.githubusercontent.com/t-emery/sais-susfin_data/main/datasets/etf_comparison-2022-10-03.csv"


# esg_etf_comparison_url <- "etf_comparison-2022-10-03.csv"
esg_etf_comparison_url <- "data/etf_comparison-2022-10-03.csv"
```

```
esg_etf_comparison <- esg_etf_comparison_url %>%
 read_csv()



##
# fonction pour les tests statistiques


# Fonction pour générer le commentaire de normalité
displayNormalityComment <- function(testResult, alpha, ticker) {
 if (testResult$p.value < alpha) {
  # comment <- paste0("Au seuil de ", "<b>", alpha*100, " %</b>", ", on rejette l'hypothèse nulle de normalité de la distribution. Les rendements du titre ", "<b>", ticker , "</b>", " ne suivent pas une loi Normale.")
  # comment <- paste0("Au seuil de <b>", alpha*100, " %</b>, on rejette l'hypothèse nulle de normalité de la distribution. Les rendements du titre ", "<b>", ticker , "</b>", " ne suivent pas une loi Normale.")

  comment <- paste0("Au seuil de ", alpha*100, " %, on rejette l'hypothèse nulle de normalité de la distribution. Les rendements du titre ", ticker , " ne suivent pas une loi Normale.")

 } else {
  # comment <- paste0("Au seuil de ", "<b>", alpha*100, " %</b>", ", l'hypothèse nulle de normalité de la distribution est acceptée. Les rendements du titre ", "<b>", ticker , "</b>", " suivent une loi Normale.")
  # comment <- paste0("Au seuil de <b>", alpha*100, " %</b>, l'hypothèse nulle de normalité de la distribution est acceptée. Les rendements du titre ", "<b>", ticker , "</b>", " suivent une loi Normale.")

  comment <- paste0("Au seuil de ", alpha*100, " %, l'hypothèse nulle de normalité de la distribution est acceptée. Les rendements du titre ", ticker , " suivent une loi Normale.")
 }
 return(comment)
```

```
}

###
# Fonction pour formater les dataframe dans DT table

render_dt_table = function(dt){
 if(is.null(dt)){
  return(NULL)
 }
 clm_format = list(list(className = "dt-center", targets = c(1)),
             list(className = "dt-left", targets = 0))

 dt_frm = DT::datatable(dt, rownames = FALSE,
                 extensions = 'Responsive', options = list(
                  ordering = TRUE,  # Conserver l'option de tri
                  paging = FALSE,  # Désactiver la pagination
                  info = FALSE,  # Désactiver le message d'information
                  searching = FALSE,
                  showPageSizeOptions = FALSE,
                  columnDefs = clm_format
                  # columnDefs = list(list(targets = "_all", className = "dt-center",
                  #                 width = "50px"))
                 )
 )

 return(dt_frm)
}

###
```

INVESTING

```
# Définir la fonction INV_base_url
INV_base_url <- function(language_id = "en") {
 # Vérifier que l'identifiant de langue est bien de deux caractères
 if (nchar(language_id) != 2) {
  stop("L'intitulé de la langue doit être exactement deux caractères.")
 }

 # Si la langue par défaut est "en", retourner l'URL de base
 if (language_id == "en") {
  return("https://investing.com")
 } else if (language_id == "fr") {
  return("https://fr.investing.com")
 }

 # Définir l'URL de la page web
 start_url <- "https://investing.com"

 # Lire le contenu de la page web
 start_page <- read_html(start_url)

 # Les URLs selon la langue choisie
 Languages_hrefs <- start_page %>%
  rvest::html_nodes(css = '#__next > header > div.flex.justify-center.xxl\\:px-\\[160px\\].xxxl\\:px-\\[300px\\].header_top-row-wrapper__7SAiJ > section > div.EditionSelector_editions__ayDVY > div > div > ul a') %>%
  html_attr("href")
```

```
# Les textes des langues dans lesquelles le site est disponible

Languages_texts <- start_page %>%

  rvest::html_nodes(css = '#__next > header > div.flex.justify-center.xxl\\:px-\\[160px\\].xxxl\\:px-\\[300px\\].header_top-row-wrapper__7SAiJ > section > div.EditionSelector_editions__ayDVY > div > div > ul a') %>%

  rvest::html_text()


# extract_between_slashes_and_dot <- function(url) {

#   sub("^//([^.]+).*", "\\1", url)

# }


# Extraire les identifiants de langue des URLs

languages <- sub("^//([^.]+).*", "\\1", Languages_hrefs)


# Créer un data frame avec les identifiants de langue, les URLs de base et les textes des langues

Languages_df <- data.frame(Language_id = languages,

                           Base_url = paste0("https:", Languages_hrefs),

                           Language = Languages_texts,

                           stringsAsFactors = FALSE)


# # Ou remplir directement la dataframe

# Languages_df = structure(list(Language_id = c("uk", "in", "ca", "au", "za", "ph", "ng", "de", "es", "mx", "fr", "it", "nl", "pt", "pl", "br", "ru", "tr", "sa", "gr", "se", "fi", "il", "jp", "kr", "cn", "hk", "id", "ms", "th", "vn", "hi"),

#                               Base_url = c("//uk.investing.com", "//in.investing.com", "//ca.investing.com", "//au.investing.com", "//za.investing.com", "//ph.investing.com", "//ng.investing.com", "//de.investing.com", "//es.investing.com", "//mx.investing.com",

#                                            "//fr.investing.com", "//it.investing.com", "//nl.investing.com", "//pt.investing.com", "//pl.investing.com", "//br.investing.com", "//ru.investing.com", "//tr.investing.com", "//sa.investing.com", "//gr.investing.com", "//se.investing.com", "//fi.investing.com",
```

```
 #                                                "//il.investing.com", "//jp.investing.com", "//kr.investing.com",
"//cn.investing.com",       "//hk.investing.com",       "//id.investing.com",       "//ms.investing.com",
"//th.investing.com", "//vn.investing.com", "//hi.investing.com"),

 #                  Language = c("English (UK)", "English (India)", "English (Canada)", "English (Australia)",
"English (South Africa)", "English (Philippines)", "English (Nigeria)", "Deutsch", "Español (España)",
"Español (México)", "Français",

 #                                "Italiano", "Nederlands", "Português (Portugal)", "Polski", "Português (Brasil)",
"Русский", "Türkçe", "العربية", "Ελληνικά", "Svenska", "Suomi", "עברית", "日本語", "한국어", "简体中文
", "繁體中文", "Bahasa Indonesia",

 #                                  "Bahasa Melayu", "ไทย", "Tiếng Việt", "हिंदी")

 #                     ),

 #                  class = "data.frame",

 #                  row.names = c(NA, -32L))



 # Chercher l'identifiant de langue dans le data frame

 result <- subset(Languages_df, Language_id == language_id)

 # result <- Languages_df %>% dplyr::filter(Language_id == language_id)


 # Vérifier si l'identifiant de langue existe

 if (nrow(result) == 0) {

  stop("L'identifiant de langue n'existe pas.")

 } else {

  return(result$Base_url)

 }

}



### Get url and ISIN

# Après préciser le type d'exchange
```

```
INV_get_info <- function(info, language="en", exchange = NULL) {

 if (!is.character(info)) {
  # stop(glue::glue("{info} must be a character"))
  stop(paste0(info, " must be a character"))
 }

 # Supprimer le dernier espace si nécessaire
 if (substr(info, nchar(info), nchar(info)) == " ") {
  info <- substr(info, 1, nchar(info) - 1)
 }

 # Encoder l'info pour l'URL
 info <- URLencode(gsub(" ", "+", info))

 info <- URLencode(gsub("&", "%26", info))

 base_url = INV_base_url(language)

 # Construire l'URL de recherche
 # search_url <- glue::glue("https://api.investing.com/api/search/v2/search?q={info}")
 search_url <- paste0("https://api.investing.com/api/search/v2/search?q=", info)

 # Effectuer la requête GET
 search_info <- httr::GET(search_url)

 # Vérifier le statut de la réponse
 if (status_code(search_info) == 200) {
  search_info_json <- httr::content(search_info, as = "text", encoding = "UTF-8")
```

```
# Gérer les erreurs de conversion JSON
tryCatch({
 json_info <- jsonlite::fromJSON(search_info_json)
}, error = function(e) {
 stop("Erreur lors de la conversion JSON: ", e$message)
})

if ("quotes" %in% names(json_info)) {
 search_quotes <- json_info$quotes

 search_quotes$url = paste0(base_url, search_quotes$url)
 if (length(search_quotes) != 0) {
  # Après chercher à faire des filtre sur les type de stock
  # search_quotes = search_quotes[1,]
  all_isin = NULL
  all_ytd = NULL

  for(elm in search_quotes$url){
   html_page = read_html(elm)

   the_isin = html_page%>%
    rvest::html_nodes("dd[data-test='isin']")%>%
    rvest::html_text()

   the_ytd = html_page%>%
    rvest::html_nodes("dd[data-test='oneYearReturn']")%>%
    rvest::html_text()
```

```
if(length(the_isin)!=0){
 # Supprimer le dernier espace si nécessaire
 if (substr(the_isin, nchar(the_isin), nchar(the_isin)) == " ") {
  the_isin <- substr(the_isin, 1, nchar(the_isin) - 1)
 } #else{
 #   the_isin <- the_isin
 # }

} else{
 the_isin <- NA
}

if(length(the_ytd)!=0){
 # Supprimer le dernier espace si nécessaire
 if (substr(the_ytd, nchar(the_ytd), nchar(the_ytd)) == " ") {
  the_ytd <- substr(the_ytd, 1, nchar(the_ytd) - 1)



 } else{
  the_ytd <- the_ytd
 }

 # if(language == "en"){
 the_ytd <- gsub(",", ".", the_ytd)
 the_ytd <- as.numeric(gsub("%", "", the_ytd))
 # }else{
 #   the_ytd <- gsub(",", ".", the_ytd)
 #   the_ytd <- as.numeric(gsub("%", "", the_ytd))
 # }
```

```
  } else{
    the_ytd <- NA
  }

  all_isin = append(all_isin, the_isin)
  all_ytd = append(all_ytd, the_ytd)

}

 search_quotes$isin <- all_isin
 search_quotes$ytd <- all_ytd

 if (!is.null(exchange) && is.character(exchange)) {
  exchange = toupper(exchange)

  index_exchange <- grep(exchange, search_quotes$exchange)

  if (length(index_exchange)!=0) {
    search_quotes <- as_tibble(search_quotes[index_exchange,])

  }

 }

 return(search_quotes)

} else {
```

```
      return("")
    }
   } else {
    # stop(glue::glue("{info} has no ID. Please check your input."))
    stop(paste0(info, " has no ID. Please check your input."))
   }
  } else {
   # Gérer les erreurs de requête
   # stop(glue::glue("Erreur {status_code(search_info)}: {content(search_info, as = 'text')}"))
   stop(paste0("Erreur ", status_code(search_info), " : ",content(search_info, as = 'text')))
  }
}


INV_get_id <- function(info, language="en", exchange = NULL) {

 elem_info = INV_get_info(info = info,
                          language= language,
                          exchange = exchange)

 if (is.null(dim(elem_info))) {
  # stop(glue::glue("{info} does not exist. Please check the information provided."))
  stop(paste0(info, " does not exist. Please check the information provided."))
 }else{
  the_id = elem_info$id
  return(the_id)
 }

}
```

```
# Obtenir l'élément de la carrière, du temps et l'url
get_carrier_and_time <- function(url) {
  # Lisez le contenu HTML de la page
  page <- read_html(url)

  # Extrait l'élément iframe
  iframe <- page %>% html_node(xpath = '//*[@id="__next"]/div[2]/div[2]/div[2]/div[1]/div[3]/div[1]/iframe')

  # Obtenez l'attribut "src" de l'iframe
  src <- iframe %>% html_attr("src")

  # Extrait les valeurs "carrier" et "time" de l'attribut "src"
  carrier <- sub(".*carrier=([^&]+).*", "\\1", src)
  time <- sub(".*time=([^&]+).*", "\\1", src)

  # Retourne les valeurs extraites
  return(list(carrier = carrier, time = time, url = url))
}


#################
# Get data from investing.com
fetch_inv_prices <- function(ticker_info,
                             from = Sys.Date() - 365*2,
                             to = Sys.Date(),
                             time_frame = 'Daily',
                             max_retries = 5, retry_delay = 1) {
```

```
 time_frame = str_to_title(time_frame)

 # httr::set_config(httr::user_agent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"))

 # UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"

 if(!(time_frame %in% c("Daily", "Weekly", "Monthly"))){
   stop("time_frame can only be 'Daily', 'Weekly' or 'Monthly' ")
 }

 the_info = INV_get_info(ticker_info)
 ticker_id = the_info$id

 # url <- paste0("https://api.investing.com/api/financialdata/historical/",
 #           ticker_id)

 # params <- list(`start-date` = from,
 #            `end-date` = to,
 #            `time-frame` =  time_frame,
 #            # `add-missing-rows` = FALSE)
 #            `add-missing-rows` = 'false')

 ## url <- paste0("https://api.investing.com/api/financialdata/historical/", ticker_id, "?start-date=", from, "&end-date=", to, "&time-frame=", time_frame, "&add-missing-rows=false")

 # url <- glue::glue("https://api.investing.com/api/financialdata/historical/{ticker_id}?start-date={from}&end-date={to}&time-frame={time_frame}&add-missing-rows=false")
```

```
 url <- paste0("https://api.investing.com/api/financialdata/historical/", ticker_id, "?start-date=", from, "&end-date=", to, "&time-frame=", time_frame, "&add-missing-rows=false")


 # url <- glue::glue("https://api.investing.com/api/financialdata/historical/{ticker_id}")


 headers <- c(
  "content-type" = "application/json",
  "domain-id" = "www",
  "sec-ch-ua-mobile" = "?0",
  "sec-ch-ua-platform" = "\"Windows\"",
  "Referer" = "https://www.investing.com/",
  "Referrer-Policy" = "strict-origin-when-cross-origin"
 )


 for (retry in 1:max_retries) {
  tryCatch({


    response = httr::GET(url, httr::add_headers(headers))


    # response <- httr::GET(url, add_headers(`Connection` = "keep-alive", `User-Agent` = UA), query = params)


    content <- httr::content(response, "text", encoding = "UTF-8")


    json <- jsonlite::fromJSON(content)


    # json <- jsonlite::fromJSON(content, flatten = TRUE
    # )
```

```
  data <- as_tibble(json$data)[, c("rowDateTimestamp",
                                   "last_open",
                                   "last_max",
                                   "last_min",
                                   "last_close",
                                   # "volume",
                                   "volumeRaw",
                                   "change_precent")]

  names(data) = c("Date", "Open", "High", "Low", "Close",
                  "Volume", "change_precent")

  data$Open <- as.numeric(gsub(",", "", data$Open))
  data$High <- as.numeric(gsub(",", "", data$High))
  data$Low <- as.numeric(gsub(",", "", data$Low))
  data$Close <- as.numeric(gsub(",", "", data$Close))
  data$Volume <- as.numeric(gsub(",", "", data$Volume))
  data$change_precent <- as.numeric(gsub(",", "", data$change_precent))

  data$Date = as.Date(data$Date)



  return(data)
 }, error = function(e) {
  if (retry < max_retries) {
    Sys.sleep(retry_delay)
  } else {
    print('Max retries reached. Please check the ticker symbol or try again later.')
```

```
      # stop("Failed to retrieve data after multiple retries. Please check your network connection and try again.")
      return(NULL)
    }
  }, warning = function(w) {
    if (retry < max_retries) {
      Sys.sleep(retry_delay)
    } else {
      warning("Warnings encountered and data could not be retrieved. Please check your network connection and try again.")
    }
  })
 }

}

# ticker_info = "FR5YT=RR"

# Exemple d'utilisation
# ticker_info <- "23769"
# from <- "2010-01-03"
# to <- "2024-07-04"
#
# daily_prices_df <- fetch_inv_prices(ticker_info,
#                         from,
#                         to,
#                         time_frame = 'Daily')
#
# daily_prices_df <- fetch_inv_prices(ticker_info,
```

```
#                         from,
#                         to,
#                         time_frame = 'Monthly')
#
# head(daily_prices_df)
# tail(daily_prices_df)

####################################



# Get stock data
# This function receives only the id
# ready_hcDt <- function(ticker_id, interval = NULL, period = NULL, max_retries = 5) {
#   period <- period
#   interval <- interval
#
#   acceptable_interval <- c("PT1M", "PT5M", "PT15M","PT30M", # For minutes
#                    "PT1H", "PT5H", # For hours
#                    "P1D", # one day
#                    "P1W", # one week
#                    "P1M" # one month
#   )
#
#   acceptable_period <- c("P1Y", "P5Y", "MAX",
#                    "PT1M","PT5M", "PT15M", "PT30M",
#                    "PT1H", "PT5H",
#                    "P1D",
#                    "P1W",
#                    "P1M")
```

```
#
#   pointscount <- 160
#
#   if (is.null(period)) {
#     if (is.null(interval)) {
#       period <- 'MAX'
#       interval <- 'P1M'
#                                                                                 url <-
glue::glue("https://api.investing.com/api/financialdata/{ticker_id}/historical/chart/?interval={interv
al}&period={period}&pointscount={pointscount}")
#     } else {
#       if (interval %in% acceptable_interval) {
#                                                                                   url <-
glue::glue("https://api.investing.com/api/financialdata/{ticker_id}/historical/chart/?interval={interv
al}&pointscount={pointscount}")
#       } else {
#         stop(glue::glue("{interval} is not a valid interval"))
#       }
#     }
#   } else {
#     if (period %in% acceptable_period) {
#       if (period == 'P1D') {
#         interval <- 'PT5M'
#       } else if (period == 'P1W') {
#         interval <- 'PT30M'
#       } else if (period == 'P1M') {
#         interval <- 'PT5H'
#       } else if (period == 'P3M') {
#         interval <- 'P1D'
#       } else if (period == 'P6M') {
```

```
#       interval <- 'P1D'
#     } else if (period == 'P1Y') {
#       interval <- 'P1W'
#     } else if (period == 'P5Y') {
#       interval <- 'P1M'
#     } else if (period == 'MAX') {
#       interval <- 'P1M'
#     }
#                                                                                 url <-
glue::glue("https://api.investing.com/api/financialdata/{ticker_id}/historical/chart/?interval={interv
al}&period={period}&pointscount={pointscount}")
#   } else {
#     stop(glue::glue("{period} is not a valid period"))
#   }
# }
#
#  for (attempt in 1:max_retries) {
#   tryCatch({
#     try_get_data <- read_html(url)
#     Sys.sleep(2)
#     try_get_data <- xml_text(try_get_data %>% rvest::html_nodes('p')) %>% fromJSON()
#
#     if (is_empty(try_get_data$data)) {
#       stop('No data available for the ticker... Be sure to provide good parameters')
#     } else {
#       ticker_dt <- try_get_data$data
#
#       if (all(ticker_dt[,7] == 0)) {
#         ticker_dt <- ticker_dt[,-7]
```

```
#       names(ticker_dt) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume')
#     } else {
#       names(ticker_dt) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Extra_info')
#     }
#
#     ticker_dt <- as_tibble(ticker_dt)
#     names(ticker_dt) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume')
#     ticker_dt$Date <- as.POSIXct((ticker_dt$Date+0.1)/1000, origin = "1970-01-01")
#
#     return(ticker_dt)
#   }
#  },
#  error = function(e) {
#   message(glue::glue("Attempt {attempt} failed: {e$message}"))
#   if (attempt == max_retries) {
#         stop("Max retries reached. Make sure you have an active internet connection and valid parameters.")
#   }
#  },
#  warning = function(w) {
#   message(glue::glue("Attempt {attempt} warning: {w$message}"))
#  })
#
#  Sys.sleep(1) # Pause before the next attempt
# }
# }


# From bourso
# Fonction pour récupérer les données de Boursorama
```

```
get_boursorama_data <- function(ticker_id = "2xFRABM5A", length = "3650", period = "0", guid = "",
max_retries = 5) {
 if(is.numeric(ticker_id) || is.character(ticker_id)){
  if(is.numeric(ticker_id)){
   ticker_id <- paste0(ticker_id, "")
   ticker_id <- switch(ticker_id,
             "23769" = "2xFRABM1A",
             "23684" = "2xDEUBM1A",
             "23700" = "2xUSABM1A",
             "23773" = "2xFRABM5A",
             "23688" = "2xDEUBM5A",
             "23703" = "2xUSABM5A",
             "23778" = "2xFRABM10A",
             "23693" = "2xDEUBM10A",
             "23705" = "2xUSABM10A")
  }else{
   ticker_id <- ticker_id
  }

 }
 # Définition des cookies
 cookies <- c(
  didomi_portal =
"eyJ1c2VyX2lkIjoiMTkzM2E5N2UtZThhYS02NjQzLThlN2YtZjQ2OTdkYzhhYjIyIiwiY3JlYXRlZCI6IjIwMj
QtMTEtMTdUMTQ6NDc6MzUuODE4WiIsInVwZGF0ZWQiOiIyMDI0LTExLTE3VDE0OjQ3OjM4LjI0N1
oiLCJ2ZW5kb3JzIjp7ImVuYWJsZWQiOlsiZ29vZ2xlIiwiYzpuZXh0LXBlcmZvcm1hbmNlIiwiYzpicnMtd
HJ1c3RwaWxvdCIsImM6YnJzLXVsdGltZWRpYSIsImM6YnJzLWdhbSIsImM6YnJzLWlhcyIsImM6YnJz
LTFwbHVzeCIsImM6YnJzLXBvb29sIiwiYzpicnMtcGlhbm8iLCJjOm1pY3Jvc29mdCIsImM6YnJzLW91
dGJyYWluIiwiYzpicnMtc21hcnQiXX0sInB1cnBvc2VzIjp7ImVuYWJsZWQiOlsiZGV2aWNlX2NoYXJhY3
RlcmlzdGljcyIsImJycy1hbmFseXNlLWV0LXBlcmYiLCJwdWJsaWNpdGVzLWVVWHp4V1VwIl19LCJ2
ZW5kb3JzX2xpIjp7ImVuYWJsZWQiOlsiZ29vZ2xlIl19LCJ2ZXJzaW9uIjoyfQ==",
```

```
   `euconsent-v2` = "CQIO_QAQIO_QAAHABBENBPFkAP_gAEPgAAqII7QGgAFgAVAA2ACAAFoATYAuAC6AF8AMIAegBBgCkAIgAVcArIBdQDCQGcAZ0A0AB1QFNALzAYCAywCMwEdgMsgGAALAAqACCALgAugB6AD8ARYAq4CmgF5gMEAZYAAA.f_wACHwAAAAA",
   didomi_cookies = "essential,analytics,marketing,social"
  )

  UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"
  params <- list(symbol = ticker_id,
                 length = length,
                 period = period,
                 guid = guid)

  attempt <- 1
  success <- FALSE
  data <- NULL

  while (attempt <= max_retries && !success) {
   response <- httr::GET(
    url = "https://www.boursorama.com/bourse/action/graph/ws/GetTicksEOD",
    httr::add_headers("user-agent" = UA),
    query = params,
    httr::set_cookies(.cookies = cookies)
   )

   if (httr::status_code(response) == 200) {
    success <- TRUE
    data <- httr::content(response, as = "parsed", type = "application/json")
   } else {
    Sys.sleep(1)  # Pause entre les essais
```

```
   attempt <- attempt + 1
  }
}

if (!success) stop("Échec après ", max_retries, " tentatives.")

# Transformer les données
if (!is.null(data$d$QuoteTab)) {
  df <- as.data.frame(do.call(rbind, data$d$QuoteTab))
  df <- df %>%
    # dplyr::mutate(
    #   d = as.Date(as.numeric(d), origin = "1970-01-01"),
    #   d = format(d, "%d/%m/%Y")  # Formatage en "26/01/2015"
    # ) %>%
    dplyr::rename(
      Date = d,
      Open = o,
      High = h,
      Low = l,
      Close = c,
      Volume = v
    )

  df$Date <- as.Date(as.numeric(df$Date))
  # df$Date <- as.Date(df$Date, format = "%d/%m/%Y")


  # Convertir correctement en tibble
  df_tibble <- as_tibble(df)
```

```
  # Vérifier si les colonnes sont des listes, et les unnest si nécessaire
  # df_tibble <- df_tibble %>%
  #   dplyr::mutate(across(everything(), ~ if (is.list(.)) unlist(.) else .))

  # Remplacer les colonnes liste par des vecteurs dans une boucle
  for (col in names(df_tibble)) {
    if (is.list(df_tibble[[col]])) {
      df_tibble[[col]] <- unlist(df_tibble[[col]])
    }
  }

  return(df_tibble)
 } else {
  return(NULL)
 }
}

# Exemple d'utilisation
# # France
# 2xFRABM1A
# 2xFRABM5A
# 2xFRABM10A
# # Allemagne ie Deutch
# 2xDEUBM1A
# 2xDEUBM5A
# 2xDEUBM10A
# ..
# # USA
```

```
# 2xUSABM1A

# 2xUSABM5A

# 2xUSABM10A


# Verifier que length est compris entre 365 et 7300 inclu et divisible par 365 avec reste = 0

# 10 ans ie 365*10

# result <- get_boursorama_data(ticker_id = "2xFRABM1A", length = "3650", period = "0")

# # 20 ans

# result <- get_boursorama_data(ticker_id = "2xFRABM5A", length = "7300", period = "0")

# print(head(result))



# Dernière version de la fonction en utilisant source

## Nouvelle Version prime

ready_hcDt <- function(ticker_id, interval = NULL, period = NULL, source = "Boursorama", max_retries = 5) {


 if (!source %in% c("Investing", "Boursorama")) {

  stop("Source invalide. Veuillez choisir 'Investing' ou 'Boursorama'.")

 }


 # Fonction pour la source "Boursorama"

 if (source == "Boursorama") {

  # Sur 20 ans

  data = get_boursorama_data(ticker_id = ticker_id, length = "7300", period = "0", max_retries = max_retries)

  # print('head(data) from bourso')

  # print(head(data))

  # print('tail(data) from bourso')
```

```
 # print(tail(data))

 return(data)
}

period <- period
interval <- interval

acceptable_interval <- c("PT1M", "PT5M", "PT15M","PT30M", # For minutes
                         "PT1H", "PT5H", # For hours
                         "P1D", # one day
                         "P1W", # one week
                         "P1M" # one month
)

acceptable_period <- c("P1Y", "P5Y", "MAX",
                       "PT1M","PT5M", "PT15M", "PT30M",
                       "PT1H", "PT5H",
                       "P1D",
                       "P1W",
                       "P1M")

pointscount <- 160

UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"

# url <- glue::glue("https://api.investing.com/api/financialdata/{ticker_id}/historical/chart/")
url <- paste0("https://api.investing.com/api/financialdata/", ticker_id, "/historical/chart/")
```

```
 if (is.null(period)) {
  if (is.null(interval)) {
   # period <- 'MAX'
   # interval <- 'P1D'
   params = list(
    interval = 'P1D',
    period = 'MAX',
    pointscount = pointscount
   )

  } else {
   if (interval %in% acceptable_interval) {
    params = list(
     interval = interval,
     # period = period,
     pointscount = pointscount
    )
    # url <- glue::glue("https://api.investing.com/api/financialdata/{ticker_id}/historical/chart/?interval={interval}&pointscount={pointscount}")

   } else {
    # stop(glue::glue("{interval} is not a valid interval"))
    stop(paste0(interval, " is not a valid interval"))
   }
  }
 } else {
  if (period %in% acceptable_period) {
   if (period == 'P1D') {
```

```
    interval <- 'PT5M'
  } else if (period == 'P1W') {
    interval <- 'PT30M'
  } else if (period == 'P1M') {
    interval <- 'PT5H'
  } else if (period == 'P3M') {
    interval <- 'P1D'
  } else if (period == 'P6M') {
    interval <- 'P1D'
  } else if (period == 'P1Y') {
    interval <- 'P1W'
  } else if (period == 'P5Y') {
    interval <- 'P1M'
  } else if (period == 'MAX') {
    interval <- 'P1M'
  }

  params = list(
    interval = interval,
    period = period,
    pointscount = pointscount
  )

  # url <- glue::glue("https://api.investing.com/api/financialdata/{ticker_id}/historical/chart/?interval={interval}&period={period}&pointscount={pointscount}")
 } else {
  # stop(glue::glue("{period} is not a valid period"))
  stop(paste0(period, " is not a valid period"))
```

```
  }
 }


 # #Nouveaux cookies
 cookies = c(
  udid = "90bcf29a6f7c3f0cb2e96e9a74c74923",
  adBlockerNewUserDomains = "1712775760",
  `_hjSessionUser_174945` = "eyJpZCI6ImU1YWIzMTFjLWI1MGEtNTg0MS1hMjFmLTZhNWNjOTNhZGFmNCIsImNyZWF0ZWQiOjE3MTI3NzU3NjA4NjcsImV4aXN0aW5nIjp0cnVlfQ==",
  `__eventn_id` = "90bcf29a6f7c3f0cb2e96e9a74c74923",
  `__eventn_uid` = "234271601",
  video_location_variant = "0",
  `user-browser-sessions` = "4",
  cf_clearance = "caGVVGdlBCtbOB0QXP9HhLQ535OOZFcvtEop5gGHOu8-1723687873-1.0.1.1-9VFlEWqoeqoagk60pQ9xhb0ugtI_4CqPu5x.9njV7Ba7avUnJgPgQ7KO4IE8c1yQwISWCdmXsqWrWs69o60Sng",
  `__eventn_id_usr` = '{"insightsPremiumUser":"No"}',
  ses_id = "NHozcmdoNDw+emttN2ZlZDFkYTg/P2FiMzQ3NDUzMCZlcTE/NGNjJWFubCJhYjIuM2c3YmUyM2QxYWBuNWVvPjQ2MzVnZzQ8Pm5rbzdhZWUxY2E4P2phNzM3NzI1NTA9ZWUxYzQ3Y2NhNWw4YWsybzMhNytlITMiMWNgMDV0byg0OzNyZzc0bj46azM3YmVkMTVhOz8wYTIzMzcxNTAwKGUu",
  `_ga` = "GA1.1.1642232810.1712775762",
  lifetime_page_view_count = "14",
  `_ga_C4NDLGKVMK` = "GS1.1.1723687904.11.1.1723687943.21.0.0",
  `__cflb` = "02DiuEaBtsFfH7bEbN4priA7DVv4KqV9WrBvRE9Vxza5a",
  `__cf_bm` = "sucP9QdhSG0_Ed46Hg_fnEfn8DlhoxPkslVsy_vJQxM-1737636455-1.0.1.1-oIXnA561Zhp0mLfCH7cb..3HFFwqxjqhwcsjFjYf6ZCEBZsyvAleoB6bSLggZFywZk4IgsPjpNSO7EnACv1NRi5oWnymzu3oRtHgqDpsXXc"
 )
```

```
for (attempt in 1:max_retries) {
 tryCatch({

  response <- httr::GET(
   url,
   httr::add_headers(
    `User-Agent` = UA,
    `Accept-Language` = "en-US,en;q=0.5",
    `Accept-Encoding` = "gzip, deflate",  # Retirer br (brotli)
    `Connection` = "keep-alive",
    `Upgrade-Insecure-Requests` = "1",
    `DNT` = "1",  # Do Not Track Header
    `Referer` = "https://www.investing.com"
   ),
   query = params ,httr::set_cookies(.cookies = cookies)
  )

  content <- httr::content(response, as = "text", encoding = "UTF-8")

  json_data <- jsonlite::fromJSON(content)

  # try_get_data <- read_html(url)
  # Sys.sleep(2)
  # try_get_data <- xml_text(try_get_data %>% rvest::html_nodes('p')) %>% fromJSON()

  if (is_empty(json_data$data)) {
   # if (is_empty(try_get_data$data)) {
   stop('No data available for the ticker... Be sure to provide good parameters')
  } else {
```

```
    # ticker_dt <- try_get_data$data
    ticker_dt <- json_data$data

    if (all(ticker_dt[,7] == 0)) {
     ticker_dt <- ticker_dt[,-7]
     names(ticker_dt) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume')
    } else {
     names(ticker_dt) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Extra_info')
    }

    ticker_dt <- as_tibble(ticker_dt)
    names(ticker_dt) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume')
    ticker_dt$Date <- as.POSIXct((ticker_dt$Date+0.1)/1000, origin = "1970-01-01")

    return(ticker_dt)
   }
  },
  error = function(e) {
   # message(glue::glue("Attempt {attempt} failed: {e$message}"))
   message(paste0("Attempt ", attempt, " failed: ", e$message))
   if (attempt == max_retries) {
    stop("Max retries reached. Make sure you have an active internet connection and valid parameters.")
   }
  },
  warning = function(w) {
   # message(glue::glue("Attempt {attempt} warning: {w$message}"))
   message(paste0("Attempt ", attempt, " warning: ", w$message))
  })
```

```
   Sys.sleep(1) # Pause before the next attempt
 }
}

# # Get stock data
# # Dernière version (version 1 prime, la version 1 fonctionne en local mais pas en déploiement)
# ready_hcDt <- function(ticker_id, interval = NULL, period = NULL, max_retries = 5) {
#
#   # Initialisation
#   to_date <- floor(as.numeric(Sys.time()))
#   from_date <- 533520000  # 1986-11-28
#   all_data <- list()
#   UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"
#
#   while(from_date < to_date) {
#    tryCatch({
#     params = list(
#       symbol = as.character(ticker_id),
#       resolution = "D",
#       from = as.character(from_date),
#       to = as.character(to_date)
#     )
#
#    print(paste("Fetching data from:", as.POSIXct(as.numeric(params$from), origin="1970-01-01")))
#
#                                                                    base_url                =
sprintf("https://tvc4.investing.com/829f2c650c7ce28678e0c877ec83942e/%d/1/1/8/history",
#                   to_date)
```

```
#
#     response <- httr::GET(
#       url = base_url,
#       httr::add_headers(`User-Agent` = UA),
#       query = params
#     )
#
#     Sys.sleep(2)  # Pour éviter trop de requêtes rapides
#
#     httr::stop_for_status(response)
#
#     content <- httr::content(response, as = "text", encoding = "UTF-8")
#     json_data <- jsonlite::fromJSON(content)
#
#     if (!is.null(json_data) && length(json_data) > 0) {
#       ticker_dt <- data.frame(
#         Date = as.POSIXct(json_data$t, origin = "1970-01-01"),
#         Open = json_data$o,
#         High = json_data$h,
#         Low = json_data$l,
#         Close = json_data$c,
#         Volume = json_data$v
#       )
#
#       all_data[[length(all_data) + 1]] <- ticker_dt
#
#       # Mettre à jour from_date pour la prochaine itération
#       if(length(json_data$t) > 0) {
#         from_date <- max(json_data$t) + 86400  # Ajouter un jour en secondes
```

```
#       } else {
#         break  # Sortir si pas de nouvelles données
#       }
#     } else {
#       break  # Sortir si pas de données
#     }
#
#   }, error = function(e) {
#     message("Error: ", e$message)
#     Sys.sleep(5)  # Attendre plus longtemps en cas d'erreur
#   })
# }
#
#  # Combiner tous les résultats et supprimer les doublons
#  if (length(all_data) > 0) {
#   final_data <- bind_rows(all_data) %>%
#     distinct(Date, .keep_all = TRUE) %>%
#     arrange(Date)
#
#   final_data$Volume <- 0
#
#   return(final_data)
#  } else {
#   stop("No data was retrieved")
#  }
# }



## This function receives only the id
```

```
## Version 1. Fonctionne en local mais pas après déploiement
# ready_hcDt <- function(ticker_id, interval = NULL, period = NULL, max_retries = 5) {
#   period <- period
#   interval <- interval
#
#   acceptable_interval <- c("PT1M", "PT5M", "PT15M","PT30M", # For minutes
#                            "PT1H", "PT5H", # For hours
#                            "P1D", # one day
#                            "P1W", # one week
#                            "P1M" # one month
#   )
#
#   acceptable_period <- c("P1Y", "P5Y", "MAX",
#                          "PT1M","PT5M", "PT15M", "PT30M",
#                          "PT1H", "PT5H",
#                          "P1D",
#                          "P1W",
#                          "P1M")
#
#   pointscount <- 160
#
#   UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"
#
#   # url <- glue::glue("https://api.investing.com/api/financialdata/{ticker_id}/historical/chart/")
#   url <- paste0("https://api.investing.com/api/financialdata/", ticker_id, "/historical/chart/")
#
#   if (is.null(period)) {
#     if (is.null(interval)) {
#       # period <- 'MAX'
```

```
#     # interval <- 'P1D'
#     params = list(
#       interval = 'P1D',
#       period = 'MAX',
#       pointscount = pointscount
#     )
#
#   } else {
#     if (interval %in% acceptable_interval) {
#       params = list(
#         interval = interval,
#         # period = period,
#         pointscount = pointscount
#       )
#                                                                                 # url <-
glue::glue("https://api.investing.com/api/financialdata/{ticker_id}/historical/chart/?interval={interv
al}&pointscount={pointscount}")
#
#     } else {
#       # stop(glue::glue("{interval} is not a valid interval"))
#       stop(paste0(interval, " is not a valid interval"))
#     }
#   }
#  } else {
#    if (period %in% acceptable_period) {
#     if (period == 'P1D') {
#       interval <- 'PT5M'
#     } else if (period == 'P1W') {
#       interval <- 'PT30M'
```

```
#     } else if (period == 'P1M') {
#       interval <- 'PT5H'
#     } else if (period == 'P3M') {
#       interval <- 'P1D'
#     } else if (period == 'P6M') {
#       interval <- 'P1D'
#     } else if (period == 'P1Y') {
#       interval <- 'P1W'
#     } else if (period == 'P5Y') {
#       interval <- 'P1M'
#     } else if (period == 'MAX') {
#       interval <- 'P1M'
#     }
#
#     params = list(
#       interval = interval,
#       period = period,
#       pointscount = pointscount
#     )
#
#                                                                                 # url <-
glue::glue("https://api.investing.com/api/financialdata/{ticker_id}/historical/chart/?interval={interv
al}&period={period}&pointscount={pointscount}")
#   } else {
#     # stop(glue::glue("{period} is not a valid period"))
#     stop(paste0(period, " is not a valid period"))
#   }
# }
#
```

#Ancien cookies

cookies = c(

udid = "a16bc43e37a197b2cd5af511799d2897",

invpro_promote_variant = "2",

`_fbp` = "fb.1.1708940869985.1116663043",

adBlockerNewUserDomains = "1708940870",

`_hjSessionUser_174945` = "eyJpZCI6Ijc3MDIwNDFiLWJlYjktNTg3ZS04N2YxLWIzNzk0ZDI3ZmI4MSIsImNyZWF0ZWQiOjE3MDg5NDA4NzE3MDAsImV4aXN0aW5nIjp0cnVlfQ==",

`_im_session_id` = "1708943371.1614021848",

session_id = "1708943371.1614021848",

user_id = "1708943371.1614021848",

`_im_session_data` = "",

`_im_fb` = "eyJmYmMiOiIiLCJmYnAiOiJmYi4xLjE3MDg5NDA4Njk5ODUuMTExNjY2MzA0MyJ9",

`__eventn_id` = "a16bc43e37a197b2cd5af511799d2897",

`__eventn_uid` = "234271601",

`_ga_REX3620224` = "GS1.1.1716603467.2.1.1716603649.0.0.0",

`__eventn_id_usr` = '{"adFreeUser":0,"investingProUser":0,"investingProPremiumUser":0}',

video_location_variant = "0",

`user-browser-sessions` = "11",

`_im_last_pv` = "1725746873.10874561097",

OptanonAlertBoxClosed = "2024-09-07T22:07:56.000Z",

`eupubconsent-v2` = "CQEktHAQEktHAAcABBENBGFgAAAAAAAAAChQAAAU1gJAA4AM-AjwBKoDfAHbAO5AgoBIgCSgEowJkgTSAn2BRQCi0FGgUcApqAAA.YAAAAAAAAAAA",

OTAdditionalConsentString = "1~",

OptanonConsent = "isGpcEnabled=0&datestamp=Sun Sep 08 2024 00:07:56 GMT+0200 (heure d’été d’Europe centrale)&version=202405.1.0&browserGpcFlag=0&isIABGlobal=false&hosts=&consentId=9aeb478c-9867-4389-a59e-c76e9e498435&interactionCount=2&landingPath=NotLandingPage&groups=C0001:1,C0002:0,C0003:0,C0004:0,V2STACK42:0&AwaitingReconsent=false&isAnonUser=1&intType=2",

ses_id = "ZSs1dG5hZm5kIGttZTRjYjRhMmtmZjIxYmUzMDI0MyU0IDc5NGM0cmRraSdmZTUpZ25lYT9qNTQzMmNuYzNgMWUzNWZuMGZsZDJrb2VjY2g0YzJtZmYyOWJmMzQyYDM7NDE3MDQxNGdkYmk1Zm41Mmd1ZXk/ezUkM2FjM2MiYCdlajV0bj5mPGRnazFlP2MyNDMyPWZmMjNiazM1MjAzKzR/",

cf_clearance = "S922rqRUy.4BrN1BsTpScqoa15pFRt_pHg2mYDQtuuQ-1728503519-1.2.1.1-DQ1BXhUb259Q3fesWDtO.ddez7zqMpQmL5KpDUr.A.0zQ7sL5Ei1TsAz.jeqcRF2KlgsXT_VfRbb3GYnvtlIpQMIic8mh_UuiJgv3wsDghYM.ZzVYehrLhGtjvdalOmVNQ7kWleyvJ1bjHx_q5UdKPONAqHUzh4tNQWfpdGE_jXgPlgCvBblZSdkNhcoos5YG.Mx5ZyVbcrmnVa3Jt3dYVQFW_oJtE2KpjiZOjW.w3m076R4eKhEs_mogQUms0sSFMDHZ4yDP0UkQ2lrR0zDtTujirg3TzSoUaY_sCEgj1303hiNHAFkMydIdwBi_X.YSuEz7ytDBn3w5IjZ0FZEsMptzRCGPdDxXRTIee20W_u5AX6L5GWRxUnyROyM4vFqRBONbXOAop4ZfyZ3wNuZkw",

`_ga_9XQG87E8PF` = "GS1.2.1728503520.9.1.1728503535.0.0.0",

`_ga_C4NDLGKVMK` = "GS1.1.1728503773.59.0.1728503773.0.0.0",

`_ga` = "GA1.1.210617884.1708940873",

lifetime_page_view_count = "217",

`__cflb` = "02DiuEaBtsFfH7bEbN4priA7DVv4KqV9Wwv5hm1ukhJdA",

`__cf_bm` = "WjrxVAp_uM8Lbo5rePjr_idMhjZU6pVQfwdMNVNPAdo-1728615061-1.0.1.1-l3.JXUuI2McSiGexemCVD5FzlRiglngth44EZ5v9ZLosdlsWWiPAAjIdLq5N8EGSYVRyMDcYUGrJGoTO9mSodYxjnKvhYHkzKvbu5ic9BsY"

)

#

#Nouveaux cookies

cookies = c(

udid = "90bcf29a6f7c3f0cb2e96e9a74c74923",

adBlockerNewUserDomains = "1712775760",

`_hjSessionUser_174945` = "eyJpZCI6ImU1YWIzMTFjLWI1MGEtNTg0MS1hMjFmLTZhNWNjOTNhZGFmNCIsImNyZWF0ZWQiOjE3MTI3NzU3NjA4NjcsImV4aXN0aW5nIjp0cnVlfQ==",

`__eventn_id` = "90bcf29a6f7c3f0cb2e96e9a74c74923",

`__eventn_uid` = "234271601",

video_location_variant = "0",

`user-browser-sessions` = "4",

```
#    cf_clearance = "caGVVGdlBCtbOB0QXP9HhLQ535OOZFcvtEop5gGHOu8-1723687873-1.0.1.1-9VFlEWqoeqoagk60pQ9xhb0ugtI_4CqPu5x.9njV7Ba7avUnJgPgQ7KO4IE8c1yQwISWCdmXsqWrWs69o60Sng",

#    `__eventn_id_usr` = '{"insightsPremiumUser":"No"}',

#                                                                                    ses_id                =
"NHozcmdoNDw+emttN2ZlZDFkYTg/P2FiMzQ3NDUzMCZlcTE/NGNjJWFubCJhYjIuM2c3YmUyM2QxYWBuNWVvPjQ2MzVnZzQ8Pm5rbzdhZWUxY2E4P2phNzM3NzI1NTA9ZWUxYzQ3Y2NhNWw4YWsybzMhNytlITMiMWNgMDV0byg0OzNyZzc0bj46azM3YmVkMTVhOz8wYTIzMzcxNTAwKGUu",

#    `_ga` = "GA1.1.1642232810.1712775762",

#    lifetime_page_view_count = "14",

#    `_ga_C4NDLGKVMK` = "GS1.1.1723687904.11.1.1723687943.21.0.0",

#    `__cflb` = "02DiuEaBtsFfH7bEbN4priA7DVv4KqV9WrBvRE9Vxza5a",

#           `__cf_bm`  =  "sucP9QdhSG0_Ed46Hg_fnEfn8DlhoxPkslVsy_vJQxM-1737636455-1.0.1.1-oIXnA561Zhp0mLfCH7cb..3HFFwqxjqhwcsjFjYf6ZCEBZsyvAleoB6bSLggZFywZk4IgsPjpNSO7EnACv1NRi5oWnymzu3oRtHgqDpsXXc"

#  )

#

#

#  for (attempt in 1:max_retries) {

#    tryCatch({

#

#     response <- httr::GET(

#       url,

#       httr::add_headers(

#         `User-Agent` = UA,

#         `Accept-Language` = "en-US,en;q=0.5",

#         `Accept-Encoding` = "gzip, deflate",  # Retirer br (brotli)

#         `Connection` = "keep-alive",

#         `Upgrade-Insecure-Requests` = "1",

#         `DNT` = "1",  # Do Not Track Header

#         `Referer` = "https://www.investing.com"
```

```
#       ),
#       query = params ,httr::set_cookies(.cookies = cookies)
#     )
#
#     content <- httr::content(response, as = "text", encoding = "UTF-8")
#
#     json_data <- jsonlite::fromJSON(content)
#
#     # try_get_data <- read_html(url)
#     # Sys.sleep(2)
#     # try_get_data <- xml_text(try_get_data %>% rvest::html_nodes('p')) %>% fromJSON()
#
#     if (is_empty(json_data$data)) {
#       # if (is_empty(try_get_data$data)) {
#       stop('No data available for the ticker... Be sure to provide good parameters')
#     } else {
#       # ticker_dt <- try_get_data$data
#       ticker_dt <- json_data$data
#
#       if (all(ticker_dt[,7] == 0)) {
#         ticker_dt <- ticker_dt[,-7]
#         names(ticker_dt) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume')
#       } else {
#         names(ticker_dt) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Extra_info')
#       }
#
#       ticker_dt <- as_tibble(ticker_dt)
#       names(ticker_dt) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume')
#       ticker_dt$Date <- as.POSIXct((ticker_dt$Date+0.1)/1000, origin = "1970-01-01")
```

```
#
#       return(ticker_dt)
#     }
#   },
#   error = function(e) {
#     # message(glue::glue("Attempt {attempt} failed: {e$message}"))
#     message(paste0("Attempt ", attempt, " failed: ", e$message))
#     if (attempt == max_retries) {
#           stop("Max retries reached. Make sure you have an active internet connection and valid parameters.")
#     }
#   },
#   warning = function(w) {
#     # message(glue::glue("Attempt {attempt} warning: {w$message}"))
#     message(paste0("Attempt ", attempt, " warning: ", w$message))
#   })
#
#   Sys.sleep(1) # Pause before the next attempt
#  }
# }



# fetch_inv_prices remplace  ready_hcDt_new


# Cette proposition est l'amélioration faite par l'IA


# https://cdn.investing.com/x/7329009/_next/static/chunks/pages/_app-06a6516b5dbd96c0.js
ready_hcDt_new <- function(ticker_info, resolution = "D",
                   from, to) {
```

```
 the_info = INV_get_info(ticker_info)
 ticker_id = the_info$id
 ticker_url = paste0(the_info$url, "-streaming-chart")
 from_timestamp <- as.numeric(as.POSIXct(from, origin="1970-01-01"))
 to_timestamp <- as.numeric(as.POSIXct(to, origin="1970-01-01"))
 needed_elm = get_carrier_and_time(ticker_url)
 the_carrier = needed_elm$carrier
 the_time = needed_elm$time

 params = list(
  symbol = ticker_id,
  resolution = "D",
  from = from_timestamp,
  to = to_timestamp
 )

 tryCatch({
  # url <- glue::glue("https://tvc4.investing.com/{the_carrier}/{the_time}/1/1/8/history?symbol={ticker_id}&resolution={resolution}&from={from_timestamp}&to={to_timestamp}")

  # url <- glue::glue("https://tvc4.investing.com/{the_carrier}/{the_time}/1/1/8/history")
  url <- paste0("https://tvc4.investing.com/", the_carrier, "/", the_time, "/1/1/8/history")

  # "https://tvc4.investing.com/dadb0c5d26296296a592501a2d7f87d4/1728205416/1/1/8/history"

  UA <- "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0"
```

```
  # Set headers
  # headers <- c(
  #   "User-Agent" = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36",
  #   "Referer" = ticker_url
  # )

  # Make the request with headers
  # response <- httr::GET(url, httr::add_headers(.headers = headers))
  # response <- httr::GET(url, add_headers(`Connection` = "keep-alive", `User-Agent` = UA))

  response <- httr::GET(url, add_headers(`Connection` = "keep-alive", `User-Agent` = UA), query = params)

  if (httr::status_code(response) == 200) {
    content <- httr::content(response, as = "text", encoding = "UTF-8")
    json_data <- jsonlite::fromJSON(content)
    df <- data.frame(
      Date = as.POSIXct(json_data$t, origin="1970-01-01", tz="UTC"),
      Open = json_data$o,
      High = json_data$h,
      Low = json_data$l,
      Close = json_data$c
    )
    return(df)
  } else {
    stop("Failed to fetch data. Status code: ", httr::status_code(response))
  }
}, error = function(e) {
```

```
   print(paste("Erreur lors de la récupération des données :", e))
   return(NULL)
 })
}



# Ancienne fonction
# Cette fonction travaille mais beug par moment
# ready_hcDt_new <- function(ticker_info, resolution = "D",
#                    from, # from = Sys.Date() - 365*2
#                    to #to = Sys.Date()
#                    ) {
#
#   the_info = INV_get_info(ticker_info)
#   ticker_id = the_info$id
#
#   ticker_url = paste0(the_info$url, "-streaming-chart")
#
#   # Convertir les dates en format numérique (timestamp)
#   from_timestamp <- as.numeric(as.POSIXct(from, origin="1970-01-01"))
#
#   to_timestamp <- as.numeric(as.POSIXct(to, origin="1970-01-01"))
#
#   # get_carrier_and_time <- function(url) {
#   #   # Lisez le contenu HTML de la page
#   #   page <- read_html(url)
#   #
#   #   # Extrait l'élément iframe
```

```
# # iframe <- page %>% html_node(xpath = '//*[@id="__next"]/div[2]/div[2]/div[2]/div[1]/div[3]/div[1]/iframe')
# #
# # # Obtenez l'attribut "src" de l'iframe
# # src <- iframe %>% html_attr("src")
# #
# # # Extrait les valeurs "carrier" et "time" de l'attribut "src"
# # carrier <- sub(".*carrier=([^&]+).*", "\\1", src)
# # time <- sub(".*time=([^&]+).*", "\\1", src)
# #
# # # Retourne les valeurs extraites
# # return(list(carrier = carrier, time = time, url = url))
# # }
#
# needed_elm = get_carrier_and_time(ticker_url)
#
# the_carrier = needed_elm$carrier
# # print(the_carrier)
# the_time = needed_elm$time
# # print(the_time)
#
# tryCatch({
#
#  # Construire l'URL de la requête
# url <- glue::glue("https://tvc4.investing.com/{the_carrier}/{the_time}/1/1/8/history?symbol={ticker_id}&resolution={resolution}&from={from_timestamp}&to={to_timestamp}")
#  # print(url)
#  # Lire le contenu de l'URL
#  response <- read_html(url) %>% html_node("body") %>% rvest::html_text()
```

```
#   Sys.sleep(1)
#   response1 <- read_html(url)
#
#   # print(response1)
#   # response <- read_html(url)
#
#
#   # if( status_code(response1) == 200){
#   #   print("Cool")
#   # }else{
#   #   print("je ne sais pas")
#   # }
#
#   # content <- content(response1, "text")
#   # content1 <- httr::content(response1, as="parsed")
#   # print(content)
#   # print(content1)
#
#   # Convertir le texte JSON en liste R
#   json_data <- jsonlite::fromJSON(response)
#
#   # Extraire les données et les transformer en dataframe
#   df <- data.frame(
#     Date = as.POSIXct(json_data$t, origin="1970-01-01", tz="UTC"),
#     Open = json_data$o,
#     High = json_data$h,
#     Low = json_data$l,
#     Close = json_data$c
#   )
```

```
#
#    return(df)
#
#    # return(list(df= df,
#    #          response = response,
#    #          # c = content,
#    #          # c_2 = content1,
#    #          json = json_data
#    # ))
#   }, error = function(e) {
#    print(paste("Erreur lors de la récupération des données :", e))
#    return(NULL)
#   })
# }



# Example of usage
# Exemple d'utilisation
# ISIN OF OAT FRANCE
# FR0127921338 # 1Year FR1YT=RR
# FR0120473253 # 5Year FR5YT=RR
# FR0011196856 # 10Year FR10YT=RR

# Utiliser ce site plutôt
# 1 year
# https://fr.tradingeconomics.com/france/52-week-bill-yield
# 5years
# https://fr.tradingeconomics.com/france/5-year-note-yield
# 10 years
```

```
# https://fr.tradingeconomics.com/france/government-bond-yield

# https://fr.tradingeconomics.com/united-states/52-week-bill-yield
# https://fr.tradingeconomics.com/united-states/5-year-note-yield
# https://fr.tradingeconomics.com/united-states/government-bond-yield

# https://fr.tradingeconomics.com/germany/52-week-bill-yield
# https://fr.tradingeconomics.com/germany/5-year-note-yield
# https://fr.tradingeconomics.com/germany/government-bond-yield

# SUR yahoo
# Treasury Note 5 year '^FVX'
# benchmark_data = ready_hcDt_new(ticker_info = "FR1YT=RR",
#                         "D",
#                         from = "2021-01-01",
#                         to = "2024-08-01")

# ready_hcDt_new <- function(ticker_info, resolution = "D", from, to) {
#
#   the_info = INV_get_info(ticker_info)
#   ticker_id = the_info$id
#
#   ticker_url = paste0(the_info$url, "-streaming-chart")
#
#   # Convertir les dates en format numérique (timestamp)
#   from_timestamp <- as.numeric(as.POSIXct(from, origin="1970-01-01"))
#
#   to_timestamp <- as.numeric(as.POSIXct(to, origin="1970-01-01"))
#
```

```
#  # get_carrier_and_time <- function(url) {
#  #  # Lisez le contenu HTML de la page
#  #  page <- read_html(url)
#  #
#  #  # Extrait l'élément iframe
#  #  iframe <- page %>% html_node(xpath = '//*[@id="__next"]/div[2]/div[2]/div[2]/div[1]/div[3]/div[1]/iframe')
#  #
#  #  # Obtenez l'attribut "src" de l'iframe
#  #  src <- iframe %>% html_attr("src")
#  #
#  #  # Extrait les valeurs "carrier" et "time" de l'attribut "src"
#  #  carrier <- sub(".*carrier=([^&]+).*", "\\1", src)
#  #  time <- sub(".*time=([^&]+).*", "\\1", src)
#  #
#  #  # Retourne les valeurs extraites
#  #  return(list(carrier = carrier, time = time, url = url))
#  # }
#
#  needed_elm = get_carrier_and_time(ticker_url)
#
#  the_carrier = needed_elm$carrier
#  the_time = needed_elm$time
#  # Construire l'URL de la requête
#  url <- glue::glue("https://tvc4.investing.com/{the_carrier}/{the_time}/1/1/8/history?symbol={ticker_id}&resolution={resolution}&from={from_timestamp}&to={to_timestamp}")
#
#  # Lire le contenu de l'URL
#  content <- read_html(url) %>% html_node("body") %>% rvest::html_text()
```

```
#
#   # Convertir le texte JSON en liste R
#   json_data <- jsonlite::fromJSON(content)
#
#   # Extraire les données et les transformer en dataframe
#   df <- data.frame(
#     Date = as.POSIXct(json_data$t, origin="1970-01-01", tz="UTC"),
#     Open = json_data$o,
#     High = json_data$h,
#     Low = json_data$l,
#     Close = json_data$c
#   )
#
#   return(df)
# }

cards_key.stats = list(
  # Card 1
  card(full_screen = TRUE,
      card_header("Current Valuation Measures"),
      htmlOutput("V_Stats_cmp_table")),
  # Card 2
  card(full_screen = TRUE,
      card_header("Financial Highlights"),
      htmlOutput("F_Stats_cmp_table")),
  # Card 3
  card(full_screen = TRUE,
      card_header("Trading Information"),
      htmlOutput("T_Stats_cmp_table")),
```

```
  # Card 4
  card(full_screen = TRUE,
     card_header("Current Valuation Measures"),
     highchartOutput("cur_v")),
  # Card 5
  card(full_screen = TRUE,
     card_header("Financial Highlights"),
     highchartOutput("prof")),
  # Card 6
  card(full_screen = TRUE,
     card_header("Financial Highlights : Income"),
     highchartOutput("INC_st")),
  # Card 7
  card(full_screen = TRUE,
     card_header("Financial Highlights : Balance Sheet"),
     highchartOutput("BS")),
  # Card 8
  card(full_screen = TRUE,
     card_header("Financial Highlights : Cash flow"),
     highchartOutput("Cashfl")),
  # Card 9
  card(full_screen = TRUE,
     card_header("Other information"),
     DTOutput("ticker_and_long.names"))

)

cards <- list(
  card(
```

```
  # full_screen = TRUE,
  full_screen = FALSE,
  card_header("Total Score evolution"),

  # Ce code aussi fonctionne à merveille
  # highchartOutput("esg_score_chart",
  #             width = "80%",
  #             height = "80%")

  # Le tooltipFormat blocque la vue
  # sparklineOutput("vbox_esg_note_spark")

  # Cette partie fonctionne mais trouver comment ajuster lorsque input$esg_dt =""
  valueBoxOutput("vbox_esg_note",
              width = "70%"
              # width = "auto" #,

              # height = "auto"
  )

  # plotOutput("esg_score_chart")
),
card(
  full_screen = TRUE,
  card_header("Recent Score"),
  # dataTableOutput("esg_recent_score")
  # tableOutput("esg_recent_score")
  DTOutput("esg_recent_score_DT") #Ceci fonctionne bien
```

```
   # plotOutput("esg_recent_score")
  ),
  card(
   full_screen = TRUE,
   card_header("Closing price"),
   plotlyOutput("esg_cl_price_chart")
  )
)

cards_tests <- list(
  card(
   full_screen = FALSE,
   card_header("Shapiro-Wilk Test"),
   htmlOutput("pvalue1"),
   uiOutput("testSWComment")
  ),
  card(
   full_screen = FALSE,
   card_header("Jarque-Bera Test"),
   htmlOutput("pvalue2"),
   uiOutput("testJBComment")
  )
)



##For esg_full data
cards_data <- list(
  card(
```

```
  full_screen = TRUE,
  card_header("ESG prices"),
  DTOutput("esg_prices")

 ),
 card(
  full_screen = TRUE,
  card_header("Risk Rating evolution by period"),
  DTOutput("esg_recent_score_DT_full") #Ceci fonctionne bien

 )
)

cards_return_data <- list(
 # card 1
 card(
  full_screen = TRUE,
  card_header("Stock return"),
  DTOutput("returns")

 ),
 # card 2
 card(
  full_screen = TRUE,
  # card_header("Benchmark return"),
  card_header("Risk free rate return"),
  DTOutput("rf_returns")

 ),
```

```
# card 3
card(
 full_screen = TRUE,
 # Avant la mise à jour
 # card_header("Stock and risk free rate returns (in %) by date"),
 # DTOutput("St_and_rf_returns")

 card_header("Stock, Benchmark and risk free rate returns (in %) by date"),
 DTOutput("St_BNC_and_rf_returns")

),
# card 4
card(
 full_screen = TRUE,
 card_header("Returns evolution"),
 # plotlyOutput("graphchart"),
 # highchartOutput("graphchart_hc", #ancien graph
 highchartOutput("graphchart_hc_", #nouveau graph
          width = "100%",
          height = "100%")

),
# card 5
card(
 full_screen = TRUE,
 card_header("Statistics table of returns"),
 DTOutput("statistics")

),
```

```
# card 6
card(
 full_screen = TRUE,
 card_header("Stock return and aggregated Risk free rate"),
 # DTOutput("St_and_rf_returns_new"),
 DTOutput("St_and_rf_returns_new2")

),
# card 7
card(
 full_screen = TRUE,
 card_header("Data used to calculate the Sharpe Ratio"),
 DTOutput("St_rf_ESG_data")

),
# card 8
card(
 full_screen = TRUE,
 card_header("Sharpe Ratio and Modified Sharpe Ratio"),
 DTOutput("Sharpe__result")

),
# card 9
card(
 full_screen = TRUE,
 card_header("3D plot"),
 plotlyOutput("plotly_3D")
),
# card 10
```

```
 card(
  full_screen = FALSE,
  card_header("Selected date"),
  verbatimTextOutput("selected_RF_id")
 ),
 # card 11
 card(full_screen = TRUE,
    card_header("Control Panel"),
    radioButtons("na_method", "Choose NA handling method:",
            choices = c("Omit NA" = "omit",
                    "Replace by mean" = "mean",
                    "Replace by zero" = "zero",
                    "Replace by previous value" = "previous"),
            selected = "omit"),
    DTOutput("summary_stats_new")
    # DTOutput("summary_stats")

 ),
 # card 12
 card(
  full_screen = TRUE,
  card_header("Returns : Stock, Benchmark and aggregated Risk free rate"),
  DTOutput("St_rf_BNC_returns_new2")

 )
)

testing_cards <- list(
 card(
```

```
  full_screen = FALSE,
  card_header("Print 1"),
  verbatimTextOutput("print_1")
 ),
 card(
  full_screen = FALSE,
  card_header("Print 2"),
  verbatimTextOutput("print_2")
 ),
 card(
  full_screen = FALSE,
  card_header("Print 3"),
  verbatimTextOutput("print_3")
 ),
 card(
  full_screen = FALSE,
  card_header("Print 4"),
  verbatimTextOutput("print_4")
 ),
 card(
  full_screen = FALSE,
  card_header("Print 5"),
  verbatimTextOutput("print_5")
 ))


# cards indicators
cards_indicators <- list(
 # card 1
```

```
  card(
   full_screen = TRUE,
   card_header("Absolute Stock Performance Indicator"),
   DTOutput("performances_stock")

  ),
  # card 2
  card(
   full_screen = TRUE,
   card_header("Relative Stock Performance Indicator"),
   DTOutput("performances_rf")

  ),
  # card 3
  card(
   full_screen = TRUE,
   card_header("Stock Risk Indicator"),
   DTOutput("risks")

  )
)

available_size = seq(5,50, by = 5)

##For esg_full data
portfolio_data <- list(
  # Card 1
  card(
    full_screen = FALSE,
```

```
  card_header("Simulation parameters"),
  selectInput(
    "nb_simulation", strong("Number of simulation"),
    vector_simulation,
    multiple = FALSE,
    selected = 100,
    selectize =TRUE),
  # br(),
  selectInput(
    "type_simulation", strong("Simulation type"),
    c("Uniform" = "uniform", "Montecarlo" = "montecarlo", "Dirichlet" = "dirichlet"),
    multiple = FALSE,
    selected = "montecarlo",
    selectize =TRUE),

  selectInput(
    "size_type", strong("Size of each point in the 3d Plot"),
    c(available_size),
    multiple = FALSE,
    selected = 20,
    selectize =TRUE),
),
# Card 2
card(
  full_screen = FALSE,
  card_header("Weights (in %)"),
  uiOutput("dynamic_weights")
  # ,  # Conteneur pour les champs de poids dynamiques
  # textOutput("total_weight_text"), # Affichage du total des poids
```

```
),
# Card 3
card(
 full_screen = TRUE,
 card_header("Annualized performance summary"),
 DTOutput("performance_data_tbl")
),
# Card 4
card(
 full_screen = TRUE,
 card_header("Tickers performance by date"),
 DTOutput("performance_data_tbl_")
),
# Card 5
card(
 full_screen = TRUE,
 card_header("Portfolio performance 3D"),
 plotlyOutput("performance_3d_plot")
),
# Card 6
card(
 full_screen = TRUE,
 card_header("Variance covariance table"),
 DTOutput("variance_cov")
),
# Card 7
card(
 # full_screen = FALSE,
 full_screen = TRUE,
```

```
   card_header("Variance covariance Heatmap"),
   highchartOutput("variance_cov_heatmap")
  )

)



# Sparkline function
hc_theme_sparkline_vb <- function(...) {
  theme <- list(
   chart = list(
    backgroundColor = NULL,
    margins = c(0, 0, 0, 0),
    spacingTop = 0,
    spacingRight = 0,
    spacingBottom = 0,
    spacingLeft = 0,
    plotBorderWidth = 0,
    borderWidth = 0,
    style = list(overflow = "visible")
   ),
   xAxis = list(
    visible = FALSE,
    endOnTick = FALSE,
    startOnTick = FALSE
   ),
   yAxis = list(
    visible = FALSE,
    endOnTick = FALSE,
```

```
  startOnTick = FALSE
),
tooltip = list(
  outside = FALSE,
  shadow = FALSE,
  borderColor = "transparent",
  botderWidth = 0,
  backgroundColor = "transparent",
  style = list(textOutline = "5px white"),
  formatter = JS(
    "function() {
       return '' + Highcharts.numberFormat(this.y, 2);
     }"
  )
),
plotOptions = list(
  series = list(
    marker = list(enabled = FALSE),
    lineWidth = 2,
    shadow = FALSE,
    fillOpacity = 0.25,
    color = "#FFFFFFBF",
    fillColor = list(
      linearGradient = list(x1 = 0, y1 = 1, x2 = 0, y2 = 0),
      stops = list(
        list(0.00, "#FFFFFF00"),
        list(0.50, "#FFFFFF7F"),
        list(1.00, "#FFFFFFFF")
      )
```

```
      )
    )
  ),
  credits = list(
    enabled = FALSE,
    text = ""
  )
)

 theme <- structure(theme, class = "hc_theme")

 if (length(list(...)) > 0) {
  theme <- hc_theme_merge(
    theme,
    hc_theme(...)
  )
 }

 theme
}

# En utilisant les données de notation
# Récupérer le nom du ticker de l'ESG
# ticker = toupper(ticker)
createVBx_esg <- function(esg_dt, ticker) {

 if(is.null(esg_dt) || ticker == ""){
  return(NULL)
```

```
} else{

  # Get stock data
  esg_dt = esg_dt
  names(esg_dt) = c("Date", "Total", "E", "S", "G")



  # Convert xts to data frame
  # df <- as.data.frame(Cl(df))
  df <- esg_dt[, c("Date", "Total")]

  # # Set the Date column as the index
  # row.names(df) <- index(df)

  # Turn rowname to column
  # df <- rownames_to_column(df, var = "Date")

  # Rename dataframe
  # names(df) <- c("Date", "Total")
  # df = esg_dt[, c("Date", "Total")]

  # Order the dataframe to be sure one not getting errors
  df <- df[order(as.Date(df$Date, format="%m/%d/%Y")),]
  # The chart
  hc <- hchart(df,
           type = "area",
           hcaes(Date, Total),
           name = paste0(toupper(ticker), " ESG Score")) %>%
   hc_size(height = 35)%>%
```

```
  hc_add_theme(hc_theme_sparkline_vb()) %>%
  hc_credits(enabled = FALSE) %>%
  hc_chart(backgroundColor = "")



valueBox3 <- function(value, title, sparkobj = NULL, subtitle, icon = NULL,
                      color = "aqua", width = 1, href = NULL){
  # color = "aqua", width = 4, href = NULL){



  shinydashboard:::validateColor(color)


  if(is.null(icon)){
    icon = NULL
  } else{ shinydashboard:::tagAssert(icon, type = "i") }


  # Cette partie travaille si les input n'est pas null
  # if (!is.null(icon))
  #   shinydashboard:::tagAssert(icon, type = "i")


  boxContent <- div(
    class = paste0("small-box bg-", color),
    div(
      class = "inner",
      tags$small(title),
      h3(value),
      if (!is.null(sparkobj)) sparkobj,
      p(subtitle)
    ),
```

```
  if (!is.null(icon)) div(class = "icon-large", icon, style = "z-index; 0")
 )

 if (!is.null(href))
  boxContent <- a(href = href, boxContent)

 div(
  class = if (!is.null(width)) paste0("col-sm-", width),
  boxContent
 )
}

# Trouver la valeur du cours de clôture qui correspond à la récente date
row.names(df) = NULL
the_max = df[df$Date == max(df$Date), 2]

# the_max = round(as.numeric(the_max[[2]]),2)
the_max = round(as.numeric(the_max),2)



prev_recent_day = df[NROW(df)-1,1]

# The previous day closing price
prev_note = df[df$Date == prev_recent_day, 2]

prev_note = round(as.numeric(prev_note),2) #Rounding it

# Calculating diff between ESG note
diff_from_past = the_max - prev_note
```

```
diff_from_past = round(as.numeric(diff_from_past),2) #Rounding it

if (diff_from_past > 0) {
  diff_from_past = paste0("+",
                          diff_from_past #, "%"
  )
  arrow_type = "&uarr;"
  graph_color = "green"

} else if (diff_from_past < 0) {
  diff_from_past = paste0(diff_from_past #, "%"
  )
  arrow_type = "&darr;"
  graph_color = "red"

} else {
  diff_from_past = paste0(diff_from_past #, "%"
  )
  arrow_type = ""
  # graph_color ="gray"
  graph_color = "yellow"
}



# print(the_max)

vb <- valueBox3(
  # value = the_max,
```

```
    value = HTML(paste0("<b>", the_max, "</b>")), # now in bold
    # title = toupper(paste0("   ", ticker, '')),
    title = HTML(paste0("<b>", toupper(paste0(" ", ticker, '')), "</b>")), # now in bold with a blank space before the ticker name
    sparkobj = hc,
    # subtitle = tagList(HTML("&uarr;"), diff_from_past, "since last day"),
    # subtitle = tagList(HTML(arrow_type), diff_from_past, "since last day"),
    # subtitle = paste0("(",tagList(HTML(arrow_type), diff_from_past, ")")),
    # subtitle = HTML(paste0("<b>", arrow_type, "(", diff_from_past, ")</b>")), # now in bold
    subtitle = tagList(HTML(arrow_type), "(", diff_from_past, ")"),
    # icon = icon("code"),
    width = 4,
    # color = "yellow",
    color = graph_color,
    href = NULL)


  }



}



# Classer par ordre ascendant
# https://investir.lesechos.fr/cours/actions/euronext-access-paris/a-z?sort_by=name:asc
# Et descendant
# https://investir.lesechos.fr/cours/actions/euronext-access-paris/a-z?sort_by=name:desc

link_Github <- tags$a(
```

```
  shiny::icon("github"), "Github",
  href = "https://github.com/Koffi-Fredysessie/BRVM"
  # target = "_blank"
)
link_Linkdin <- tags$a(
  shiny::icon("linkedin"), "Linkedin",
  href = "https://www.linkedin.com/in/sk-fred"
  # target = "_blank"
)
link_Gmail <- tags$a(
  shiny::icon("envelope"), "Gmail",
  href = "mailto:koffisessie@gmail.com"
  # target = "_blank"
)



ui <- page_navbar(
  # theme = bs_theme(version = 5, bootswatch = "cosmo"),
  theme = bs_theme(version = 5,
                   bootswatch = "cosmo",
                   primary = "#2E8B57"),
  title = "ESG Funds and stocks Dashboard",
  position = "fixed-top",

  id = "navbar",  # Ajout d'un id pour le navbar

  header = div(
    useShinyjs(),
    tags$style(type="text/css", "body {padding-top: 40px;}"),
```

```
tags$head(
 tags$link(rel = "manifest", href="public/manifest.json", crossorigin = "use-credentials"),
 tags$style(HTML("
#      .tab-content {
#   margin-top: 20px;
# }
.justified-text {
 text-align: justify;
 margin-left: 15px;
 margin-right: 15px;
}
.custom-mainPanel {
 width: 95%;  /* Adjust width percentage as needed */
   margin: 0 auto;  /* Center the main panel horizontally */
}
.custom-navset {
padding-top: 5px;  /* Adjust padding as needed */
}
.stats-container {
   display: grid;
   grid-template-columns: repeat(2, 1fr);
   gap: 20px;
   margin-bottom: 20px;
 }
 .stat-card {
   text-align: center;
   padding: 20px;
   background-color: #f8f9fa;
   border-radius: 10px;
```

```
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}
.stat-number {
  font-size: 24px;
  font-weight: bold;
  color: #2E8B57;
}
.carousel-container {
  position: relative;
  width: 100%;
  height: 300px;
  overflow: hidden;
  border-radius: 10px;
}
.carousel-image {
  width: 100%;
  height: 300px;
  object-fit: cover;
  border-radius: 10px;
}
  .tab-content { margin-top: 20px; }
  .justified-text {
    text-align: justify;
    line-height: 1.6;
    padding: 20px;
  }
  .slick-slide img {
    width: 100%;
    height: 300px;
```

```
  object-fit: cover;
  border-radius: 8px;
}
.stats-container {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  gap: 20px;
  margin-bottom: 20px;
}
.stat-card {
  text-align: center;
  padding: 20px;
  background-color: #f8f9fa;
  border-radius: 10px;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}
.stat-number {
  font-size: 24px;
  font-weight: bold;
  color: #2E8B57;
}
.big-letter {
  font-size: 1.5em;
  font-weight: bold;
  color: #2E8B57;
}
.slick-prev, .slick-next {
  z-index: 1;
  width: 40px;
```

```
  height: 40px;
}
.slick-prev {
  left: 10px;
}
.slick-next {
  right: 10px;
}
.slick-prev:before, .slick-next:before {
  font-size: 40px;
  opacity: 0.8;
  color: #2E8B57;
}
.carousel-container {
  position: relative;
  margin: 0 20px;
}
.creator-info {
  margin-top: 20px;
  padding: 15px;
  background-color: #f8f9fa;
  border-radius: 10px;
  border-left: 4px solid #2E8B57;
}
.video-container {
  position: relative;
  padding-bottom: 56.25%;
  height: 0;
  overflow: hidden;
```

```
      margin: 20px 0;
    }
    .video-container iframe {
      position: absolute;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
    }
  "))
 )
),
# useShinyjs(),


# tags$style(type="text/css", "body {padding-top: 25px;}"),
# Avant j'utilisais aussi cette partie
# tags$style(type="text/css", "body {padding-top: 40px;}"),

# tags$head(tags$style(HTML("
# .tab-content {
#   margin-top: 20px;
# }"))),

# tags$head(tags$style(HTML("
#   .tab-content {
#     margin-top: 20px;
#   }
#   .justified-text {
```

```
#    text-align: justify;
#    margin-left: 15px;
#    margin-right: 15px;
#  }
#  .custom-mainPanel {
#    width: 95%;  /* Adjust width percentage as needed */
#     margin: 0 auto;  /* Center the main panel horizontally */
#  }
#  .custom-navset {
#  padding-top: 5px;  /* Adjust padding as needed */
#  }
#  .stats-container {
#     display: grid;
#     grid-template-columns: repeat(2, 1fr);
#     gap: 20px;
#     margin-bottom: 20px;
#    }
#    .stat-card {
#     text-align: center;
#     padding: 20px;
#     background-color: #f8f9fa;
#     border-radius: 10px;
#     box-shadow: 0 2px 4px rgba(0,0,0,0.1);
#    }
#    .stat-number {
#     font-size: 24px;
#     font-weight: bold;
#     color: #2E8B57;
#    }
```

```
#    .carousel-container {
#      position: relative;
#      width: 100%;
#      height: 300px;
#      overflow: hidden;
#      border-radius: 10px;
#    }
#    .carousel-image {
#      width: 100%;
#      height: 300px;
#      object-fit: cover;
#      border-radius: 10px;
#    }
# "))),

# fillable = FALSE,
nav_spacer(), # push nav items to the right
lang = "en",

# Quand j'utilise nav_panel, le nom s'affiche mais pas les tabpanel
# nav_panel("Stock Markets Summary",
nav_panel(
 "INTRODUCTION",
 value = "Introduction",
 icon = icon("info-circle"),

 div(
  style = "padding: 2rem; margin-top: 60px;",
  fluidRow(
```

```
# Left column (5 columns)
column(
 width = 5,
 card(
  class = "h-100",
  card_body(
   class = "justified-text",
   p(HTML("<span class='big-letter'>E</span>nvironmental, <span class='big-letter'>S</span>ocial, and <span class='big-letter'>G</span>overnance (ESG) investment funds have gained significant traction over the past few decades. These funds focus on investing in companies that adhere to certain environmental, social, and governance criteria, making them an attractive option for investors who are conscious about sustainability and ethical considerations.")),
   p("The history of ESG investing can be traced back to the 1960s when socially responsible investing (SRI) began to take shape. Today, ESG funds encompass a wide range of investment strategies, from screening and shareholder advocacy to impact investing and thematic investing."),
   p("By integrating ESG factors into their investment processes, ESG funds aim to promote sustainable business practices, mitigate risks, and identify opportunities that traditional financial analysis might overlook."),

   # Video section
   div(
    class = "video-container",
    tags$iframe(
     src = "https://www.canal-u.tv/chaines/canal-aunege/embed/100089?t=0",
     width = "560",
     height = "315",
     frameborder = "0",
     allow = "accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture",
     allowfullscreen = TRUE
    )
   ),
```

```
# Creator information
# Creator information
# Creator information
div(
 class = "creator-info",
 h4("À propos du créateur", style = "color: #2E8B57; margin-bottom: 15px;"),
 div(
  style = "display: flex; align-items: start; gap: 20px;",
  # Image de l'auteur
  div(
   style = "flex-shrink: 0;",
   img(
    # Utiliser l'ID du fichier Google Drive dans l'URL
    # src = "https://lh3.googleusercontent.com/drive-storage/AJQWtBOM2SyP_AiLwJUsLCs-KQjNLFpEwC_1O4dlCHy5p98EFpjopkW6R3IyW4SF98jajCg6VzdrEcYjRjGur8YW0xJPnhnfJPf2Cy49B7cQEEwvgw=w2048-h1200",
    src = "Fredy_sessie.JPG",
    style = "width: 150px; height: 150px; border-radius: 10px; object-fit: cover;"
   )
  ),
  # Informations textuelles
  div(
   p(
    style = "margin-bottom: 10px;",
    strong("Créateur et concepteur de la plateforme :"),
    br(),
    "Koffi Frederic SESSIE",
    br(),
```

```
  # Ajout de la tabulation (padding-left) et réduction de la taille de la police (font-size)
  span(
    "• Analyste & ingénieur financier,",
    style = "padding-left: 12px; font-size: 0.8em; display: block;"
  ),
  span(
    "• Analyste crédit & Data analyst,",
    style = "padding-left: 12px; font-size: 0.8em; display: block;"
  ),
  span(
    "• Auteur des packages financiers R Euronext et Brvm",
    style = "padding-left: 12px; font-size: 0.8em; display: block;"
  )
  # h6("> Analyste et ingénieur financier"),
  # "- Analyste et ingénieur financier",
  # br(),
  # h6("> Analyste crédit et Data analyst"),
  # "- Analyste crédit et Data analyst",
  # br(),
  # "Auteur des Packages R nommé Euronext et Brvm"
),
p(
  style = "margin-bottom: 0;",
  strong("Sous la supervision de :"),
  br(),
  "Phillippe Gillet",
  br(),
  # "Maître de Conférences HC – HDR"
  span(
```

```
            "• Maître de Conférences HC – HDR",
            style = "padding-left: 12px; font-size: 0.8em; display: block;"
          )
        )
      )
    )
  )
 )
),

# Right column (7 columns)
column(
 width = 7,
 card(
  full_screen = TRUE,
  card_header(
   class = "bg-primary text-white",
   h2("ESG Investment Overview", class = "text-center")
  ),

  card_body(
   # Statistics cards
   div(
    class = "stats-container",
    div(
     class = "stat-card",
     p(class = "stat-number", "$17.1T"),
     p("Global ESG Assets", style = "color: #666;")
```

```
    ),
    div(
      class = "stat-card",
      p(class = "stat-number", "+42%"),
      p("YoY Growth", style = "color: #666;")
    ),
    div(
      class = "stat-card",
      p(class = "stat-number", "3,000+"),
      p("ESG Funds Worldwide", style = "color: #666;")
    ),
    div(
      class = "stat-card",
      p(class = "stat-number", "89%"),
      p("Of Millennials Interested in ESG", style = "color: #666;")
    )
  ),
  # Ajouter la source des statistiques
  div(
    style = "text-align: right; font-style: italic; font-size: 0.8em; margin-top: -15px; margin-bottom: 15px; color: #666;",
    "Sources: ",
    tags$a(href = "https://www.bloomberg.com/professional/blog/esg-assets-may-hit-53-trillion-by-2025-a-third-of-global-aum/",
           "Bloomberg", target = "_blank"), " (Global ESG Assets & YoY Growth), ",
    # tags$a(href = "https://www.morningstar.com/news/marketwatch/20210202476/sustainable-investment-funds-more-than-doubled-in-2020",
    #        "Morningstar", target = "_blank"), " (ESG Funds Worldwide), ",
    tags$a(href = "https://www.morningstar.com/",
```

```
            "Morningstar", target = "_blank"), " (ESG Funds Worldwide), ",
          #   tags$a(href   =   "https://www.morganstanley.com/ideas/sustainable-investing-growing-popularity",
          #      "Morgan Stanley", target = "_blank"), " (Millennials Interest)"
          tags$a(href = "https://www.morganstanley.com/",
            "Morgan Stanley", target = "_blank"), " (Millennials Interest)"
        ),

        # Carousel using slickR
        div(
          class = "carousel-container",
          style = "margin-top: 20px;",
          slickROutput("slick_carousel", width = "100%", height = "300px")
        ),
        # Dans la partie card_body du côté droit, après le carousel
        div(
          style = "margin-top: 30px;",
          h3("Top Performing Funds 2024",
            class = "text-center",
            style = "color: #2E8B57; margin-bottom: 20px;"),
          DTOutput("fund_table")
        )
       )
      )
     )
    )
  )
 ),
 # nav_panel(
```

```
#  "INTRODUCTION",
#  value = "Introduction",
#  icon = icon("info-circle"),
#
#  layout_column_wrap(
#   width = 1,
#   style = css(padding = "2rem"),
#
#   card(
#    full_screen = TRUE,
#    card_header(
#     class = "bg-primary text-white",
#     h2("Introduction to ESG Investment Funds", class = "text-center")
#    ),
#
#    card_body(
#     layout_column_wrap(
#      width = 1/2,
#
#      # Left column with text
#      div(
#       class = "justified-text",
#        p("Environmental, Social, and Governance (ESG) investment funds have gained significant traction over the past few decades. These funds focus on investing in companies that adhere to certain environmental, social, and governance criteria, making them an attractive option for investors who are conscious about sustainability and ethical considerations."),
#        p("The history of ESG investing can be traced back to the 1960s when socially responsible investing (SRI) began to take shape. Today, ESG funds encompass a wide range of investment strategies, from screening and shareholder advocacy to impact investing and thematic investing."),
```

```
#           p("By integrating ESG factors into their investment processes, ESG funds aim to promote sustainable business practices, mitigate risks, and identify opportunities that traditional financial analysis might overlook.")
#         ),
#
#         # Right column
#         div(
#           # Statistics cards
#           div(
#             class = "stats-container",
#             div(
#               class = "stat-card",
#               p(class = "stat-number", "$17.1T"),
#               p("Global ESG Assets", style = "color: #666;")
#             ),
#             div(
#               class = "stat-card",
#               p(class = "stat-number", "+42%"),
#               p("YoY Growth", style = "color: #666;")
#             ),
#             div(
#               class = "stat-card",
#               p(class = "stat-number", "3,000+"),
#               p("ESG Funds Worldwide", style = "color: #666;")
#             ),
#             div(
#               class = "stat-card",
#               p(class = "stat-number", "89%"),
#               p("Of Millennials Interested in ESG", style = "color: #666;")
```

```
#           )
#         ),
#
#         # Carousel
#         div(
#           class = "carousel-container",
#           uiOutput("image_carousel")
#         )
#       )
#     )
#   )
# ),


# nav_panel("INTRODUCTION", value = "Introduction", icon = icon("info-circle"),
#       fluidRow(
#         column(2),
#         column(8, align = "center",
#             tags$br(),
#             tags$br(),
#             # h3("Introduction to ESG Investment Funds"),
#                 h2(HTML("<span style='font-weight:bold; color:darkgreen;'>Introduction to ESG Investment Funds</span>")),
#             tags$br(),
#             # h3(""),
#             div(class = "justified-text",
#                 p("Environmental, Social, and Governance (ESG) investment funds have gained significant traction over the past few decades. These funds focus on investing in companies that
```

```
adhere to certain environmental, social, and governance criteria, making them an attractive option for investors who are conscious about sustainability and ethical considerations."),

#                 # tags$br(),

#               p("The history of ESG investing can be traced back to the 1960s when socially responsible investing (SRI) began to take shape. Investors started to avoid companies involved in activities such as tobacco production, apartheid practices, and environmental pollution. Over the years, the criteria expanded to include broader ESG factors, and the approach evolved from merely excluding negative entities to actively seeking out positive ones."),

#                 # tags$br(),

#               p("In recent years, ESG funds have grown exponentially, driven by increasing awareness and demand from investors who want their investments to align with their values. Regulatory changes and an emphasis on corporate transparency have also contributed to the rise of ESG investing. Today, ESG funds encompass a wide range of investment strategies, from screening and shareholder advocacy to impact investing and thematic investing."),

#                 # tags$br(),

#               p("By integrating ESG factors into their investment processes, ESG funds aim to promote sustainable business practices, mitigate risks, and identify opportunities that traditional financial analysis might overlook. This approach not only benefits the environment and society but also has the potential to enhance long-term financial performance.")

#               )

#         ),

#         column(2)

#       )

# ),

nav_panel("ESG ANALYSIS",

      # "Stock Analysis content",

      icon = icon("leaf"),

      class = "custom-navset",

      navset_card_underline(


       # sidebarLayout(

       sidebar = sidebar(

        # sidebarPanel(
```

```
# id = "sidebar",
align = "center",
width = 320,
# actionButton("toggleSidebar", "Toggle Sidebar"),
# varSelectInput(
# Nouvelle version pour tester avec M. Gillet
# selectInput("esg_sbl", "Choose Stock for ESG Analysis",
#           ""),

radioButtons(
 inputId = "unik_stock_elm",
 label = "Stock or Fund input method",
 choices = c("Select stock directly"= "Select",
          "Use ISIN" = "Isin",
          "Search Stock" = "Entrance"),
 inline = FALSE # Afficher les boutons radio verticalement
),
uiOutput("unik_esg_sbl_ui"),
actionButton("unik_run_button", "Submit"),

# Ancienne version test
# selectInput(
#   inputId = "esg_sbl", label = strong("Choose a stock"), c("", esg_etf_comparison$ticker)),

# br(),

selectInput(inputId = "Rf_rate", label = strong("Select a risk free rate"), c("", risk_free_rate),
         # selected = "OAT 5y",
         selectize = TRUE),
```

```
        # "ticker", "Choose an ESG", c("AAPL", "MSFT", "GOOGL", "AMZN", "TSLA", "NVDA", "V", "PYPL", "INTC")),

        # dateRangeInput("date", strong("Analysis horizon"),

        #           start = "2020-01-06", end = (Sys.Date()-1)),

        # Désactiver le choix des dates futures



        # # J'ai voulu permettre à l'utilisateur de choisir la source de ses données mais cette option sera utilisée dans le futur

        # selectInput(inputId = "BNC_source", label = strong("Select data source"), c("Yahoo", "Euronext"),

        #           selected = "Yahoo",

        #           selectize = TRUE),


        # # Dans l'UI, remplacer le selectInput statique par :

        # uiOutput("benchmark_selector"),


        selectInput(inputId = "Benchmarks", label = strong("Select a benchmark"),

                c("", benchmark_list$Name),

                # options = list(maxOptions = 1700),

                selectize = TRUE),


        dateRangeInput(inputId = "date", # Ajout de l'id

                  label = strong("Analysis horizon"),

                  start = "2020-01-06",

                  end = Sys.Date(),

                  max = Sys.Date(),

                  # end = "2024-10-13",

                  # max = "2024-10-13",

                  min = "1990-01-01",
```

```
            language = "en-CA",
            startview = "month"),

        # br(),

        # sliderInput(inputId = "bins", label = strong("No. of histogram"), min = 1, max = 100, value=50),
        colorPickr(inputId = "colour1", strong("Chart Colour"),
            selected = "#27AD81",
            swatches = c(
             scales::viridis_pal()(9),
             scales::brewer_pal(palette = "Blues")(9),
             scales::brewer_pal(palette = "Reds")(9)
            ),
            update = "change",
            opacity = TRUE,
            preview = FALSE,
            hue = FALSE,
            interaction = list(
             hex= FALSE,
             rgba = FALSE,
             input = FALSE,
             save = FALSE,
             clear = FALSE
            ),
            pickr_width = "245px"),
        # tags$br(),
        selectInput(inputId = "retfreq", label = strong("Frequency of returns"), c("Daily" = "daily",
                                                                                    "Weekly" = "weekly",
```

```
                                        "Monthly" = "monthly",
                                        "Yearly" = "yearly"),
        # freq_list,
        selected = "daily"),
    # radioButtons("retfreq", "Frequency of returns", c("daily" = "daily",
    #                                 "weekly" = "weekly",
    #                                 "monthly" = "monthly",
    #                                 "yearly" = "yearly"), inline = T),
    selectInput(inputId = "log", label = strong("Returns"), c("Arithmetic" = "arithmetic",
                                        "Logarithmic" = "log"), selected = "arithmetic"),
    # radioButtons("log", "Returns",
    #          choices = c("Arithmetic" = "arithmetic",
    #                  "Logarithmic" = "log"), inline = T),
    # tags$br(),



    # numericInput("rfrate", "Risk Free Rate", value = 0.02, min = -100, max = 100, step = 0.01
    #          # , align = center
    #          ),

    checkboxInput(inputId = "benchmark", label = "Use risk free rate to calculate Sharpe ratio?",
value = FALSE),


    conditionalPanel(
     condition = "input.benchmark == false",
     numericInput(inputId = "rfrate_new",
            label = "Daily Risk Free Rate",
            value = 0.02, min = -100, max = 100, step = 0.01)
     # , align = center
```

```
        ),

        checkboxInput(inputId = "sc_question", label = "Use the optimised smooth coefficient?", value = TRUE),


        conditionalPanel(

         condition = "input.sc_question == false",

         numericInput(inputId = "sc",label = "Smooth coefficient", value = 0.05, min = 0, max = 1, step = 0.01)

         # , align = center

        ),


        sliderInput(inputId = "alpha",

                label = strong("Significance level"),

                min = 0.01,

                max = 0.2,

                value = 0.10),


        numericInput(inputId = "minaccret",label = strong("Minimum Acceptable Return"), value = 0, min = 0, max = 100, step = 0.01)

        # ,

        # selectInput("Up_Color", strong("Up Color"), str_to_title(my_color),selected = "Darkgreen", selectize = TRUE),

        #  selectInput("Down_Color", strong("Down Color"), str_to_title(my_color), selected = "Red", selectize = TRUE)


       ), # fin sidebar

       mainPanel(

        class = "custom-mainPanel",

        # navset_card_underline(

        # navset_pill_list(
```

```
navset_card_pill(
 # id ="Main",
 title = "Single ESG Analysis",
 placement = "above",

 # well = TRUE,
 # width = "100%",
 # br(),
 # fluidRow(align = "center",
 #        selectInput("hideorshow", label = strong("Disposition du Sidebar"),
 #                    choices = c("Show", "Hide"), selected = "Show")),
 nav_panel("ESG Summary",
         # br(),
         layout_columns(
          col_widths = c(6,6),
          layout_columns(
           col_widths = c(12, 12),
           row_heights = c("200px", "150"),
           cards[[1]], cards[[2]]),
          row_heights = c("380px"),
          cards[[3]]
         )

         # ,
         # plotlyOutput("graphCours"),
         # column(width = 12,
         #       plotOutput("histo", width = "auto", height = "400px")
         # ),
         # column(width = 12,
```

```
        #       plotOutput("qqplot", width = "auto", height = "500px")
        # ),
        # column(width = 12,
        #       plotOutput("checkresid", width = "auto", height = "500px")
        # ),
        # br(), br(),
        # column(width = 12,
        #       highchartOutput("stockChart", width = "auto", height = "500px")
        # )
),# Fin ESG summary nav_panel
nav_panel("Data",
        layout_columns(
         col_widths = c(12,12),
         # row_heights = c("200px", "150"),
         cards_data[[2]], cards_data[[1]]

        )
        # ,
        # DTOutput("BNC_dt"),
        # DTOutput("BNC_dt_return")
        ), #end nav_panel Data
nav_panel("Return",
        # layout_columns(
        #   col_widths = c(6,6),
        #   # row_heights = c("200px", "150"),
        #   cards_return[[1]], cards_return[[2]]
        #
        # ),
        # ,DTOutput
```

```
        # box(title = "Stock Return", width = 6, DTOutput("returns")),
        # cards_return_data[[4]],
        layout_columns(
         col_widths = c(5,7),
         # row_heights = c("200px", "150"),
         cards_return_data[[11]], cards_return_data[[4]]

        ),
        layout_columns(
         col_widths = c(6,6),
         # row_heights = c("200px", "150"),
         cards_return_data[[1]], cards_return_data[[2]]

        ),
        cards_return_data[[3]],
        # cards_return_data[[6]], # retirer Stock return and aggregated Risk free rate
        cards_return_data[[12]]
        # ,
        # cards_return_data[[10]]

        # box(title = "Stock Return ",width = 6, DT::dataTableOutput("returns"))
),#end nav_panel Return
nav_panel("Statistics",
        # layout_columns(
        #   col_widths = c(6),
        #   # row_heights = c("200px", "150"),
        #   cards_return_data[[5]],
        #   box(title="Shapiro-Wilk Test",
        #     width = 12,
```

```
#     # textOutput("pvalue1"),
#     htmlOutput("pvalue1"),
#     # textOutput("testSW")
#     htmlOutput("testSWComment")
#   ),
#   box(title="Jarque-Bera Test",
#     width = 12,
#     # textOutput("pvalue2"),
#     # textOutput("testJB"),
#     htmlOutput("pvalue2"),
#     htmlOutput("testJBComment")
#   )
#
# )# Fin layout
layout_columns(
 col_widths = c(6,6),
 row_heights = c("440px", "440px"),
 cards_return_data[[5]],
 layout_columns(
  col_widths = c(12, 12),
  row_heights = c("220px", "220px"),
  cards_tests[[1]],
  cards_tests[[2]]
 )

)

), #Fin navpanel
nav_panel("Indicator",
```

```
            layout_columns(
              col_widths = c(6,6),
              row_heights = c("200px", "200"),
              cards_indicators[[1]], cards_indicators[[2]]
            ),
            row_heights = c("auto"),
            cards_indicators[[3]]
            #,
            # box(title="Shapiro-Wilk Test",width = 12,textOutput("pvalue1"),textOutput("testSW")),
            # box(title="Jarque-Bera Test",width = 12,textOutput("pvalue2"),textOutput("testJB")),

          ), # Fin nav_panel
          nav_panel("Model",
            # row_heights = c("100%"),
            # cards_return_data[[7]]
            row_heights = c("100%", "min-content"),
            cards_return_data[[9]],
            layout_columns(
              col_widths = c(12,12),
              row_heights = c("auto", "auto"),
              cards_return_data[[7]], cards_return_data[[8]]
              # ,
              # testing_cards[[1]], testing_cards[[2]],
              # testing_cards[[3]], testing_cards[[4]],
              # testing_cards[[5]]
            )
          ) # Fin nav_panel
        ) #end navset_card_pill
```

```
        ) #end main panel

      ) # Fin navset_card_underline
),# Fin nav_panel ESG ANALYSIS

nav_panel("FUNDS COMPARISON",
      #"Stock Comparison Content",
      icon = icon("line-chart"),

      tags$style(HTML("
  .shiny-input-radiogroup {
    text-align: left !important;
  }
  .radio label {
    display: block;
  }
")),

      # Ajoutez du CSS personnalisé pour aligner les radio buttons à gauche
      #       tags$style(HTML("
      #  .radio-inline, .checkbox-inline {
      #    text-align: left !important;
      #    margin-left: 0px !important;
      #  }
      #  .shiny-input-container {
      #    display: inline-block;
      #    vertical-align: middle;
      #  }
```

```
# )),
navset_card_underline(

 # sidebarLayout(
 sidebar = sidebar(
  # sidebarPanel(
  # id = "sidebar",
  align = "center",
  width = 320,

  # selectInput(
  #   "cmp_esg_sbl", strong("Choose stock(s)"),
  #   c("", esg_etf_comparison$ticker),
  #   multiple = TRUE,
  #   selected = '',
  #   selectize =TRUE),
  radioButtons(
   # checkboxInput(
   inputId = "stock_elm",
   label = "Choose stock input method",
   # size = 'lg',
   choices = c("Select stock(s) directly"= "Select",
               "Use ISIN" = "Isin",
               "Search Stock(s)" = "Entrance"),
   # inline = TRUE # Ajout de l'option inline
   inline = FALSE # Afficher les boutons radio verticalement
  ),
  uiOutput("cmp_esg_sbl_ui"),
  actionButton("run_button", "Run"),
```

```
# selectInput("cmp_Rf", strong("Select a risk free rate"), c("", risk_free_rate),
#           # selected = "OAT 5y",
#           selectize = TRUE),



# # Désactiver le choix des dates futures
# dateRangeInput(inputId = "cmp_date", # Ajout de l'id
#              label = strong("Analysis horizon"),
#              start = "2020-01-06",
#              end = Sys.Date(),
#              max = Sys.Date(),
#              min = "1990-01-01",
#              language = "en-CA",
#              startview = "month"),

# br(),

colorPickr(inputId = "t_colour", strong("Overview table Colour"),
          selected = "#27AD81", #1B64AE #I can use also this colour
          swatches = c(
            scales::viridis_pal()(9),
            scales::brewer_pal(palette = "Blues")(9),
            scales::brewer_pal(palette = "Reds")(9)
          ),
          update = "change",
          opacity = TRUE,
          preview = FALSE,
          hue = FALSE,
```

```
        interaction = list(
          hex= FALSE,
          rgba = FALSE,
          input = FALSE,
          save = FALSE,
          clear = FALSE
        ),
        pickr_width = "245px"),

      tags$br() #,
      # selectInput("cmp_retfreq",
      #       strong("Frequency of returns"),
      #       c("Daily" = "daily",
      #         "Weekly" = "weekly",
      #         "Monthly" = "monthly",
      #         "Yearly" = "yearly"),
      #       # freq_list,
      #       selected = "daily"),

      # selectInput("cmp_log", strong("Returns"), c("Arithmetic" = "arithmetic",
      #                             "Logarithmic" = "log"), selected = "arithmetic"),

      # checkboxInput("cmp_benchmark", "Use benchmark returns as risk free rate?", value = FALSE),

      # conditionalPanel(
      #   condition = "input.cmp_benchmark == false",
      #   #align = center not working,
      #   numericInput("cmp_rfrate_new",
```

```
        #           "Daily Risk Free Rate",
        #           value = 0.02,
        #           min = -100,
        #           max = 100,
        #           step = 0.01)
        #
        # ),
        # checkboxInput("cmp_sc_question", "Use the optimised smooth coefficient?", value = TRUE),
        #
        # conditionalPanel(
        #   condition = "input.cmp_sc_question == false",
        #   numericInput("cmp_sc", "Smooth coefficient", value = 0.05, min = 0, max = 1, step = 0.01)
        #   # , align = center not working
        # ),

        # sliderInput("cmp_alpha",
        #           strong("Significance level :"),
        #           min = 0,
        #           max = 0.2,
        #           value = 0.10),

        # numericInput("cmp_minaccret", "Minimum Acceptable Return", value = 0, min = 0, max = 200, step = 0.01)

      ), # Fin sidebar portofolio
      mainPanel(
       class = "custom-mainPanel",
       navset_card_pill(
```

```
title = "Stocks comparison tool",
placement = "above",
nav_panel("Sustainability", #"Sustainability  Overview"

      # layout_columns(
      #   layout_columns(
      #     col_widths = c(3, 9),
      #     row_heights = c("200px", "200"),
      #     portfolio_data[[2]], portfolio_data[[3]])
      #
      # )
      # br(),
      htmlOutput("funds_cmp_table")  # Kable output for displaying bonds

), #Fin navpanel "Sustainability"
nav_panel("Keys Stats",
      # htmlOutput("K_Stats_cmp_table"),
      cards_key.stats[[1]],
      cards_key.stats[[2]],
      cards_key.stats[[3]]


      # portfolio_data[[4]],
      # portfolio_data[[6]]

), #Fin navpanel "Performance and Risks"
nav_panel("Graph tool",

      # fluidRow(
```

```
#   column(3,selectInput(
#    "nb_simulation", strong("Number of simulation"),
#    c("", vector_simulation),
#    multiple = FALSE,
#    selected = '',
#    selectize =TRUE)),
#   column(3, selectInput(
#    "type_simulation", strong("Simulation type"),
#    c("uniform", "montecarlo", "dirichlet"),
#    multiple = FALSE,
#    selected = "montecarlo",
#    selectize =TRUE))
#
# ), #Fin fluidrow

# layout_columns(
#   #   col_widths = c(12,12),
#   col_widths = c(4, 8),
#   row_heights = c("500px", "500px"),
#   portfolio_data[[1]],
#   portfolio_data[[5]]
# ),

# plotlyOutput("perf_plot_min_var"),
# plotlyOutput("perf_plot_max_var"),
# plotlyOutput("perf_plot_front_eff")
# br(),
# htmlOutput("f_table")  # Kable output for displaying bonds
cards_key.stats[[4]],
```

```
                cards_key.stats[[5]],
                cards_key.stats[[6]],
                cards_key.stats[[7]],
                cards_key.stats[[8]],
                cards_key.stats[[9]]



            ) #Fin navpanel "Graph tool"
            # ,
            # nav_panel("Portfolio optimization",
            #         DTOutput("weights_table__")
            # ) #Fin navpanel 4

          ) #fin naset_car_pill
        ) #Fin main page
      ) # Fin navset_card_underline

), #Fin nav_panel "FUNDS COMPARISON"



nav_panel("PORTFOLIO",
      # "Dashboard content",
      icon = icon("briefcase"),
      value = "Portfolio",  # Ajout d'une valeur pour le nav_panel

      # tags$script(HTML("
      #    Shiny.addCustomMessageHandler('toggleSidebar', function(expand) {
      #      var sidebar = document.querySelector('.collapse-toggle');
      #      var isExpanded = sidebar.getAttribute('aria-expanded') === 'true';
```

```
#      if (expand && !isExpanded) {
#        sidebar.click();
#      } else if (!expand && isExpanded) {
#        sidebar.click();
#      }
#    });
#   ")),
# useShinyjs(),

navset_card_underline(
 # sidebarLayout(
 sidebar = sidebar(
  # sidebarPanel(
  id = "sidebar",
  open = TRUE,
  align = "center",
  width = 320,

  # selectInput(
  #   "db_esg_sbl", strong("Choose stock(s)"),
  #   c("", esg_etf_comparison$ticker),
  #   multiple = TRUE,
  #   selected = '',
  #   selectize =TRUE),

  radioButtons(
   inputId = "pf_stock_elm",
   label = "Choose stock input method",
   choices = c("Select stock(s) directly"= "Select",
```

```
          "Use ISIN(s)" = "Isin",
          "Search Stock(s)" = "Entrance"),
 inline = FALSE # Afficher les boutons radio verticalement
),
uiOutput("db_esg_sbl_ui"),
actionButton("pf_run_button", "Submit"), # Ajouter un bouton d'action


selectInput("gp_Rf", strong("Select a risk free rate"), c("", risk_free_rate),
        # selected = "OAT 5y",
        selectize = TRUE),
# Désactiver le choix des dates futures
dateRangeInput(inputId = "gp_date", # Ajout de l'id
          label = strong("Analysis horizon"),
          start = "2010-01-06",
          end = Sys.Date(),
          max = Sys.Date(),
          min = "1990-01-01",
          language = "en-CA",
          startview = "month"),


br(),
# tags$br(),
selectInput("gp_retfreq", strong("Frequency of returns"), c("Daily" = "daily",
                                    "Weekly" = "weekly",
                                    "Monthly" = "monthly",
                                    "Yearly" = "yearly"),
        # freq_list,
        selected = "daily"),
```

```
selectInput("gp_log", strong("Returns"), c("Arithmetic" = "arithmetic",
                                           "Logarithmic" = "log"), selected = "arithmetic"),

checkboxInput("gp_benchmark", "Use benchmark returns as risk free rate?", value = FALSE),

conditionalPanel(
 condition = "input.gp_benchmark == false",
 #align = center not working,
 numericInput("gp_rfrate_new",
          "Daily Risk Free Rate",
          value = 0.02,
          min = -100,
          max = 100,
          step = 0.01)

),
checkboxInput("gp_sc_question", "Use the optimised smooth coefficient?", value = TRUE),

conditionalPanel(
 condition = "input.gp_sc_question == false",
 numericInput("gp_sc", "Smooth coefficient", value = 0.05, min = 0, max = 1, step = 0.01)
 # , align = center not working
),

sliderInput(inputId = "gp_alpha",
        label = strong("Significance level :"),
        min = 0,
        max = 0.2,
        value = 0.10),
```

```
          numericInput("gp_minaccret", "Minimum Acceptable Return", value = 0, min = 0, max = 100,
step = 0.01)


        ), # Fin sidebar portofolio
        mainPanel(
         class = "custom-mainPanel",
         navset_card_pill(
          title = "Portfolio Performance Analysis",
          placement = "above",
          id = "portfolioNavset",  # Ajout d'un id pour le navset
          nav_panel("Portfolio Summary",
                # uiOutput("dynamic_weights"),  # Conteneur pour les champs de poids dynamiques
                # textOutput("total_weight_text"), # Affichage du total des poids
                # DTOutput("performance_data_tbl") #,
                layout_columns(
                 #col_widths = c(6,6),
                 layout_columns(
                  col_widths = c(3, 9),
                  row_heights = c("200px", "200"),
                  portfolio_data[[2]], portfolio_data[[3]])


                )

          ), #Fin navpanel 1
          nav_panel("Data",
                value = "Data",
                # layout_columns(
                #   col_widths = c(12,12),
```

```
        #   # row_heights = c("200px", "150"),
        #   cards_data[[2]],
        portfolio_data[[4]],
        portfolio_data[[6]]

), #Fin navpanel 2
nav_panel("Modelization",
        value = "Modelization",

        # fluidRow(
        #   column(3,selectInput(
        #    "nb_simulation", strong("Number of simulation"),
        #    c("", vector_simulation),
        #    multiple = FALSE,
        #    selected = '',
        #    selectize =TRUE)),
        #   column(3, selectInput(
        #    "type_simulation", strong("Simulation type"),
        #    c("uniform", "montecarlo", "dirichlet"),
        #    multiple = FALSE,
        #    selected = "montecarlo",
        #    selectize =TRUE))
        #
        # ), #Fin fluidrow
        DTOutput("weights_table"),

        # layout_columns(
        # #   col_widths = c(12,12),
        # col_widths = c(4, 8),
```

```
        # row_heights = c("500px", "500px"),
        # portfolio_data[[1]],
        # portfolio_data[[5]]
        # ),
        layout_columns(
         #   col_widths = c(12,12),
         col_widths = c(12),
         row_heights = c("500px"),
         portfolio_data[[5]]
        ),
        portfolio_data[[7]],
        plotlyOutput("perf_plot_min_var"),
        plotlyOutput("perf_plot_max_var"),
        plotlyOutput("perf_plot_front_eff")
), #Fin navpanel 3 Modelisation

nav_panel("Portfolio comparison",
        value = "PortfolioComparison",
        fluidRow(align = 'center',
              column(3,
                  dateRangeInput(inputId = "pf_date", # Ajout de l'id
                              label = strong("Analysis horizon"),
                              start = "2020-01-06",
                              end = Sys.Date(),
                              max = Sys.Date(),
                              min = "1990-01-01",
                              language = "en-CA",
                              startview = "month")
```

```
                ),
                column(3, selectInput("pf_retfreq", strong("Data frequency"), c("Daily" = "daily",
                                                                               "Weekly" = "weekly",
                                                                               "Monthly" = "monthly",
                                                                               "Yearly" = "yearly"),
                                      selected = "daily")),
                column(3, selectInput("pf_log", strong("Returns"), c("Arithmetic" = "arithmetic",
                                                                     "Logarithmic" = "log"), selected = "arithmetic")),
                # column(3, numericInput("pf_sc", "Smooth coefficient", value = 0.05, min = 0, max = 1, step = 0.01)),
                column(3, selectInput("pf_sc", "Smooth coefficient", seq(0, 1, 0.01), multiple = FALSE, selected = 0.05))
                # ) #first fluidrow
                ,
                # br(),
                # fluidRow(align = 'center',
                # column(4, numericInput("pf_minaccret", "Minimum Acceptable Return", value = 0, min = 0, max = 200, step = 0.01)),
                # column(3, selectInput("pf_minaccret", "Min. Acceptable Return (in %)", seq(-100, 100, 0.01), multiple = FALSE, selected = 0)),
                column(3,
                    selectizeInput("pf_minaccret",
                                   "Min. Acceptable Return (in %)",
                                   choices = NULL,  # Important : initialiser avec NULL
                                   options = list(maxOptions = 20000),
                                   selected = 0)),
                column(3, selectInput("pf_minaccESG", "Min. Acceptable ESG [0 ; 100]", c(0:100), multiple = FALSE, selected = 10)),
                column(3, selectInput("nb_portfolio", strong("Number of portfolio"), c(1:20), multiple = FALSE, selected = 1)),
```

```
                # column(3, selectInput(inputId = "alpha_level", label = strong("Significance level
(%)"), seq(0, 10, 0.01), multiple = FALSE, selected = 5))
                column(3,
                    selectizeInput(inputId = "alpha_level",
                                label = strong("Significance level (%)"),
                                choices = NULL,  # Important : initialiser avec NULL
                                options = list(maxOptions = 20000),
                                selected = 5))


            ), #second fluidrow
            # fluidRow(
            #
            #   column(12,
            #       dateRangeInput("pf_date", strong("Analysis horizon"),
            #                   start = "2020-01-06", end = Sys.Date() - 1, max = Sys.Date()),
            #       selectInput("pf_retfreq", strong("Data frequency"), c("Daily" = "daily",
            #                                   "Weekly" = "weekly",
            #                                   "Monthly" = "monthly",
            #                                   "Yearly" = "yearly"),
            #                selected = "yearly"),
            #       selectInput("pf_log", strong("Returns"), c("Arithmetic" = "arithmetic",
            #                                   "Logarithmic" = "log"), selected = "arithmetic"),
            #       numericInput("pf_sc", "Smooth coefficient", value = 0.05, min = 0, max = 1, step
= 0.01)
            #
            #       )
            # ),
            # numericInput("pf_minaccret", "Minimum Acceptable Return", value = 0, min = 0, max
= 200, step = 0.01),
```

```
                # numericInput("pf_minaccESG", "Minimum Acceptable ESG", value = 0, min = 0, max = 100, step = 0.01),
                # selectInput("nb_portfolio", strong("Number of portfolio"), c(1:20), multiple = FALSE, selected = 1),
                # actionButton("pf_construc_btn", "Generate empty portfolio"),
                # fluidRow(align = 'center',
                #       column(4),
                #       column(4, actionButton("pf_construc_btn", "Generate empty portfolio")),
                #       column(4)),

                uiOutput("dynamic_ui"),

                fluidRow(align = 'center',
                     column(4),
                     column(4, actionButton("pf_create_user_w", "Generate User Weights")),
                     column(4)),

                # actionButton("pf_create_user_w", "Generate User Weights"),
                # conditionalPanel(
                #   condition = "input.pf_construc_btn > 0",
                #   actionButton("pf_create_user_w", "SUBMIT")
                # ),
                # card(uiOutput("dynamic_weights_ui")),

                # card(
                #   full_screen = FALSE,
                #   card_header("Porfolio Weights (in %)"),
                #       uiOutput("dynamic_weights_ui")   # Conteneur pour les champs de poids dynamiques
                # ),
```

```
                uiOutput("dynamic_weights_ui"),

                # "portfolio_table" fonctionne avec l'ancien code

                # tabPanel("Portfolio comparison", DTOutput("portfolio_table")),

                uiOutput("portfolio_comparison_ui"),
                br(),
                uiOutput("detailportfolio_comparison_ui")
                # nav_panel("Portfolio optimization",

                # DTOutput("weights_table__")

            ) #Fin navpanel 4 "Portfolio comparison",

          ) #fin naset_car_pill
        ) #Fin main page
      ) # Fin navset_card_underline

), #Fin nav_panel "PORTOFOLIO"

# nav_panel("FORECAST",
#         # "Stock Forecast content",
#         icon = icon("cloud")),

nav_menu(
 title = "GET IN TOUCH!",
 align = "right",
 nav_item(link_Gmail),
```

```
  nav_item(link_Github),
  nav_item(link_Linkdin)
 ),
 nav_item(
  input_dark_mode(id = "dark_mode", mode = "light")
 )
) #fin page_navbar



server <- function(input, output, session) {

 output$slick_carousel <- renderSlickR({
  slickR(
   obj = esg_images,
   height = 300,
   width = "100%"
  ) +
   settings(
    autoplay = TRUE,
    autoplaySpeed = 5000,
    dots = TRUE,
    arrows = TRUE,
    centerMode = FALSE,
    centerPadding = "0",
    adaptiveHeight = TRUE,
    infinite = TRUE
   )
 })
```

```
 output$fund_table <- renderDT({
  fund_data <- get_fund_data()

  # print("fund_data")
  # print(head(fund_data))

  # Nettoyer et préparer les données
  fund_data_clean <- fund_data %>%
   dplyr::mutate(
    Fund_URL = stringr::str_extract(Fund, "(?<=\\().*(?=\\))"),
    Fund_Name = stringr::str_extract(Fund, "(?<=\\[).*(?=\\])"),
    Fund = base::sprintf('<a href="%s" target="_blank">%s</a>', Fund_URL, Fund_Name)
   ) %>%
   dplyr::select(-Fund_URL, -Fund_Name) %>%
   dplyr::select(Fund, Category, Rating, YTD_Return, Month12_Return, Year5_Return)

  # Convertir les ratings en images HTML
  fund_data <- fund_data_clean %>%
   dplyr::mutate(
    Rating = case_when(
     Rating == "Bronze" ~ '<img src="https://www.dupageseniorcouncil.org/wp-content/uploads/2018/02/Bronze-Icon-1.jpg" height="60">',
     Rating == "Gold" ~ '<img src="https://www.dupageseniorcouncil.org/wp-content/uploads/2018/02/Gold-Icon.jpg" height="60">',
     Rating == "Silver" ~ '<img src="https://www.dupageseniorcouncil.org/wp-content/uploads/2018/02/Silver-Icon-298x300.jpg" height="60">',
     TRUE ~ Rating
    )
   )
```

```
datatable(
 fund_data,
 options = list(
  pageLength = 5,
  dom = 'ft<"bottom"p>',
  order = list(list(3, 'desc')),
  columnDefs = list(
   list(className = 'dt-center', targets = c(1, 3:5)),
   list(
    targets = 2,
    className = 'dt-center rating-column',
    sortable = FALSE
   ),
   list(
    targets = 3:5,
    render = JS(
     "function(data, type, row) {
      return type === 'display' && data != null ? data.toFixed(2) + '%' : data;
     }"
    )
   )
  ),
  initComplete = JS(
   "function(settings, json) {
    $(this.api().table().container()).addClass('table-container');
   }"
  )
 ),
```

```
    escape = FALSE,
    rownames = FALSE,
    class = 'cell-border',  # Retiré 'stripe'
    caption = htmltools::tags$caption(
      style = 'caption-side: bottom; text-align: left; font-style: italic; padding-top: 10px;',
      'Source : ',
      htmltools::a(
        href  =  'https://www.morningstar.co.uk/uk/news/256091/10-funds-outperforming-their-peers-in-2024-so-far.aspx',
        'Morningstar UK - 10 funds outperforming their peers in 2024 so far',
        target = '_blank'
      )
    )
  ) %>%
    formatStyle(
      columns = c("YTD_Return", "Month12_Return", "Year5_Return"),
      backgroundColor = styleInterval(
        c(10, 20),
        c('#ffecec', '#fff9ec', '#ecffec')
      )
    ) %>%
    formatStyle(
      'Rating',
      backgroundColor = 'white'
    )
 })



 ############################
```

```
observe({
 # Pour pf_minaccret
 updateSelectizeInput(session,
                "pf_minaccret",
                choices = seq(-100, 100, 0.01),
                selected = 0,
                server = TRUE)
})

observe({
 # Pour alpha_level
 updateSelectizeInput(session,
                "alpha_level",
                choices = seq(0, 10, 0.01),
                selected = 5,
                server = TRUE)
})



##############################
######### METRICS ############
##############################

# Create reactive values for each portfolio's tickers and weights
portfolio_tickers <- reactiveValues()
portfolio_weights <- reactiveValues()

observeEvent(input$nb_portfolio, {
```

```
################### NOUVEAUTE AJOUTEE ############
# Réinitialiser les poids

portfolio_weights_rv <- reactiveValues()
# Réinitialiser l'état d'initialisation

weights_initialized(FALSE)

# Réinitialiser les UI des tableaux
output$portfolio_comparison_ui <- renderUI({ NULL })
output$detailportfolio_comparison_ui <- renderUI({ NULL })

################### FIN NOUVEAUTE ############

num_portfolios <- as.numeric(input$nb_portfolio)
output$dynamic_ui <- renderUI({
 rows <- lapply(1:num_portfolios, function(i) {
   column(
    width = 3,
    wellPanel(
     h4(paste("Portfolio", i)),
     selectInput(
      inputId = paste0("cmp_pf_stock_elm_", i),
      label = "Stock input method",
      choices = c("Select stock(s) directly" = "Select",
                  "Use ISIN(s)" = "Isin",
                  "Search Stock(s)" = "Entrance")
     ),
     uiOutput(paste0("db_esg_sbl_ui_", i)),
```

```
        selectInput(paste0("pf_Rf_", i), strong("Select a risk free rate"), c("", risk_free_rate)),
        selectInput(inputId = paste0("pf_BNC_", i), label = strong("Select a benchmark"), c("", benchmark_list$Name),
                    selectize = TRUE),
        checkboxInput(paste0("pf_benchmark_", i), "Use benchmark returns as risk free rate?", value = FALSE),
        conditionalPanel(
          condition = paste0("input.pf_benchmark_", i, " == false"),
          numericInput(paste0("pf_rfrate_new_", i),
                       "Yearly Risk Free Rate",
                       value = 0.02, min = -100, max = 100, step = 0.01)
        )
       )
      )
    })

    rows_with_spaces <- list()
    for (i in seq_along(rows)) {
      rows_with_spaces[[length(rows_with_spaces) + 1]] <- rows[[i]]
      if (i %% 4 == 0 && i != num_portfolios) {
        rows_with_spaces[[length(rows_with_spaces) + 1]] <- div(style = "margin-bottom: 20px;")
      }
    }

    fluidRow(align = "center", rows_with_spaces)
  })
 })

 # observeEvent(input$nb_portfolio, {
```

```
#   num_portfolios <- as.numeric(input$nb_portfolio)
#   output$dynamic_ui <- renderUI({
#    fluidRow(
#     align = "center",
#     lapply(1:num_portfolios, function(i) {
#      list(
#       column(
#        width = 3,
#        wellPanel(
#         h4(paste("Portfolio", i)),
#         selectInput(
#          inputId = paste0("cmp_pf_stock_elm_", i),
#          label = "Stock input method",
#          choices = c("Select stock(s) directly" = "Select",
#                      "Use ISIN" = "Isin",
#                      "Search Stock(s)" = "Entrance")
#         ),
#         uiOutput(paste0("db_esg_sbl_ui_", i)),
#         selectInput(paste0("pf_Rf_", i), strong("Select a risk free rate"), c("", risk_free_rate)),
#           checkboxInput(paste0("pf_benchmark_", i), "Use benchmark returns as risk free rate?",
value = FALSE),
#         conditionalPanel(
#          condition = paste0("input.pf_benchmark_", i, " == false"),
#          numericInput(paste0("pf_rfrate_new_", i),
#                       "Yearly Risk Free Rate",
#                       value = 0.02, min = -100, max = 100, step = 0.01)
#         )
#        )
#       ),
```

```
#         if (i %% 4 == 0) br() else NULL
#       )
#     })
#   )
#  })
# })



# Base qui fonctionne bien mais je veux ajuster les espaces
# observeEvent(input$nb_portfolio, {
#  # observeEvent(input$pf_construc_btn, {
#  num_portfolios <- as.numeric(input$nb_portfolio)
#
#  output$dynamic_ui <- renderUI({
#   fluidRow(
#     align = "center",
#     lapply(1:num_portfolios, function(i) {
#       column(
#         width = 3,
#         wellPanel(
#           h4(paste("Portfolio", i)),
#           selectInput(
#           # radioButtons(
#             inputId = paste0("cmp_pf_stock_elm_", i),
#             label = "Stock input method",
#             choices = c("Select stock(s) directly" = "Select",
#                         "Use ISIN" = "Isin",
#                         "Search Stock(s)" = "Entrance")
#             # ,
```

```
#            # inline = FALSE
#          ),
#          uiOutput(paste0("db_esg_sbl_ui_", i)),
#          selectInput(paste0("pf_Rf_", i), strong("Select a risk free rate"), c("", risk_free_rate)),
#            checkboxInput(paste0("pf_benchmark_", i), "Use benchmark returns as risk free rate?",
value = FALSE),
#          conditionalPanel(
#           condition = paste0("input.pf_benchmark_", i, " == false"),
#           numericInput(paste0("pf_rfrate_new_", i),
#                    "Yearly Risk Free Rate",
#                    # "Daily Risk Free Rate",
#                    value = 0.02, min = -100, max = 100, step = 0.01)
#          )
#         )
#       )
#     })
#   )
# })
# })


# Generate UI elements for each portfolio based on selected input method
observe({
  lapply(1:as.numeric(input$nb_portfolio), function(i) {
    local({
      portfolio_index <- i
      output[[paste0("db_esg_sbl_ui_", portfolio_index)]] <- renderUI({
        method <- input[[paste0("cmp_pf_stock_elm_", portfolio_index)]]
        if (is.null(method)) {
          return(NULL)
```

```
      }
      if (method == "Select") {
        selectInput(
          paste0("portf_esg_sbl_", portfolio_index),
          strong("Choose stock(s)"),
          c("", esg_etf_comparison$ticker),
          multiple = TRUE,
          selected = '',
          selectize = TRUE
        )
      } else if (method == "Isin") {
        textInput(
          paste0("portf_esg_sbl_", portfolio_index),
          strong("Enter ISIN(s)"),
          # placeholder = "Enter ISIN(s) separated by commas"
          placeholder = "ISIN(s) separated by ','"
        )
      } else if (method == "Entrance") {
        textInput(
          paste0("portf_esg_sbl_", portfolio_index),
          strong("Enter ticker(s)"),
          # placeholder = "Enter ticker(s) separated by commas"
          placeholder = "Ticker(s) separated by ','"
        )
      }
    })
  })
 })
})
```

```
# Event to handle tickers for each portfolio
observe({
 lapply(1:as.numeric(input$nb_portfolio), function(i) {
  local({
   portfolio_index <- i
   ticker_event <- eventReactive(input$pf_create_user_w, {
    method <- input[[paste0("cmp_pf_stock_elm_", portfolio_index)]]
    tickers <- input[[paste0("portf_esg_sbl_", portfolio_index)]]
    if (is.null(tickers)) {
     return(NULL)
    }
    if (method == "Select") {
     return(tickers)
    } else if (method == "Isin") {
     user_input <- tickers
     if (is.null(user_input) || user_input == "") {
      return(NULL)
     }
     isin_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
     test <- yf_isin_to_ticker(isin_vector)
     if (length(test) == 3) {
      return(test$tick)
     } else {
      return(NULL)
     }
    } else {
     user_input <- tickers
     if (is.null(user_input) || user_input == "") {
```

```
        return(NULL)
      }
      ticker_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
      return(ticker_vector)
    }
   })
   assign(paste0("pf_tickers_to_analyze_", portfolio_index), ticker_event, envir = .GlobalEnv)
  })
 })
})

# # Observe each portfolio's tickers to handle weight inputs
# observe({
#   lapply(1:as.numeric(input$nb_portfolio), function(i) {
#    local({
#     portfolio_index <- i
#     observeEvent(get(paste0("pf_tickers_to_analyze_", portfolio_index))(), {
#      tickers <- get(paste0("pf_tickers_to_analyze_", portfolio_index))()
#      if (is.null(tickers) || length(tickers) == 0 || (length(tickers) == 1 && tickers == "")) {
#       return(NULL)
#      }
#      new_labels <- tickers
#      weights <- paste0("w_", portfolio_index, "_", str_to_title(tickers))
#      # Initialize weight values
#      for (ticker in tickers) {
#       if (is.null(portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]])) {
#        portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]] <- 0
#       }
#      }
```

```
#       removeUI(selector = paste0("#dynamic_weights_ui_", portfolio_index), multiple = TRUE)
#       for (j in seq_along(tickers)) {
#         insertUI(
#           selector = paste0("#dynamic_weights_ui_", portfolio_index),
#           ui = tags$div(
#             fluidRow(
#                                         column(12, numericInput(weights[j], new_labels[j], value = portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", tickers[j])]], min = 0, max = 100, step = 0.001, width = '400px'))
#             ),
#             class = "pf_dynamic-weight-input"
#           ),
#           multiple = TRUE
#         )
#       }
#     })
#   })
# })
#
# # Observe and update weights for each ticker in each portfolio
# observe({
#   lapply(1:as.numeric(input$nb_portfolio), function(i) {
#     local({
#       portfolio_index <- i
#       observe({
#         lapply(get(paste0("pf_tickers_to_analyze_", portfolio_index))(), function(ticker) {
#           observeEvent(input[[paste0("w_", portfolio_index, "_", str_to_title(ticker))]], {
#                         portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]] <- input[[paste0("w_", portfolio_index, "_", str_to_title(ticker))]]
```

```
#         }, ignoreNULL = FALSE)
#       })
#     })
#   })
# })
#
# # Render UI for dynamic weights input
# observeEvent(input$pf_create_user_w, {
#   output$dynamic_weights_ui <- renderUI({
#     lapply(1:as.numeric(input$nb_portfolio), function(i) {
#       div(id = paste0("dynamic_weights_ui_", i))
#     })
#   })
# })

######################Ancienne version ######################
#####################################################################

# output$dynamic_weights_ui <- renderUI({
#   num_portfolios <- as.numeric(input$nb_portfolio)
#
#   # Créer une liste pour stocker les colonnes
#   columns <- lapply(1:num_portfolios, function(i) {
#     column(
#       width = 3,  # Chaque colonne aura une largeur de 3
#       # width = 12,
#       # column(12,
#       card(
```

```
#       full_screen = TRUE,
#       # fill = TRUE,
#       # full_screen = FALSE,
#       card_header(paste("Weights for Portfolio", i, "(in %)")),
#      div(id = paste0("dynamic_weights_ui_", i)) # Conteneur pour les champs de poids dynamiques
#     )
#     # )
#    )
#  })
#
#  # Créer des lignes pour regrouper les colonnes
#  rows <- list()
#  for (i in seq(1, length(columns), by = 4)) {
#    rows[[length(rows) + 1]] <- fluidRow(
#      # column(12, #retirer après ça
#      columns[i:min(i + 3, length(columns))] # Regrouper les colonnes par groupes de 4
#    )
#    # columns[i:min(i + 3, length(columns))] # Regrouper les colonnes par groupes de 4
#    # ) #à retirer après
#  }
#
#  # Retourner toutes les lignes en un seul UI
#  tagList(rows)
# })


###################### Fin Ancienne version ###################
#############################################################
```

```
###################### Nouvelle version ######################
#############################################################

# Ajouter cette variable réactive en dehors de la fonction renderUI
selected_portfolios <- reactiveVal(0)

# Modifier le renderUI
output$dynamic_weights_ui <- renderUI({
 # Mettre à jour le nombre de portefeuilles sélectionnés
 isolate({
  if (selected_portfolios() != input$nb_portfolio) {
   selected_portfolios(input$nb_portfolio)
   # Réinitialiser les poids et les tickers ici
   portfolio_weights <- reactiveValues()
   for(i in 1:50) {
    removeUI(
     selector = paste0("#dynamic_weights_ui_", i, " .pf_dynamic-weight-input"),
     multiple = TRUE
    )
   }
  }
 })

 num_portfolios <- as.numeric(input$nb_portfolio)

 # Créer une liste pour stocker les colonnes
 columns <- lapply(1:num_portfolios, function(i) {
  column(
   width = 3,
```

```
    card(
     full_screen = TRUE,
     card_header(paste("Weights for Portfolio", i, "(in %)")),
     div(
      id = paste0("dynamic_weights_ui_", i),
      # Ajouter une classe unique pour faciliter la réinitialisation
      class = "portfolio-weights-container"
     )
    )
   )
  })

  # Créer des lignes pour regrouper les colonnes
  rows <- list()
  for (i in seq(1, length(columns), by = 4)) {
   rows[[length(rows) + 1]] <- fluidRow(
    columns[i:min(i + 3, length(columns))]
   )
  }

  # Retourner toutes les lignes en un seul UI
  tagList(rows)
 })

 # Ajouter cet observeEvent pour gérer la réinitialisation
 observeEvent(input$nb_portfolio, {
  # Forcer la réinitialisation des conteneurs de poids
  removeUI(selector = ".portfolio-weights-container .pf_dynamic-weight-input", multiple = TRUE)
```

```
 # Réinitialiser les valeurs réactives
 portfolio_weights <- reactiveValues()
 weights_initialized(FALSE)

 # Nettoyer les variables globales des tickers
 for(i in 1:50) {
  if(exists(paste0("pf_tickers_to_analyze_", i), envir = .GlobalEnv)) {
   rm(list = paste0("pf_tickers_to_analyze_", i), envir = .GlobalEnv)
  }
 }
}, priority = 1000)

###################### Fin nouvelle version ###################
################################################################

# Initialiser les conteneurs d'UI pour chaque portefeuille
# output$dynamic_weights_ui <- renderUI({
#   num_portfolios <- as.numeric(input$nb_portfolio)
#
#   # Utilisation de layout_columns pour organiser les cards en colonnes
#   fluidRow(
#    layout_columns(
#    # col_widths = 3,
#    col_widths = rep(3, num_portfolios), # Définir la largeur de chaque colonne à 3
#    lapply(1:num_portfolios, function(i) {
#     card(
#      full_screen = FALSE,
#      card_header(paste("Weights for Portfolio", i, "(in %)")),
#      div(id = paste0("dynamic_weights_ui_", i)) # Conteneur pour les champs de poids dynamiques
```

```
#     )
#   })
#  )
#  ) #après retirer le fluidRow
# })
```

```
######################################################################
################## Ancienne version qui conction #################
## La partie à corriger en pour mettre à jour les UI weight
## Observer chaque portefeuille pour gérer les entrées de poids

observe({
 lapply(1:as.numeric(input$nb_portfolio), function(i) {
  local({
   portfolio_index <- i

   observeEvent(get(paste0("pf_tickers_to_analyze_", portfolio_index))(), {
    tickers <- get(paste0("pf_tickers_to_analyze_", portfolio_index))()

    if (is.null(tickers) || length(tickers) == 0 || (length(tickers) == 1 && tickers == "")) {
     return(NULL)
    }

    new_labels <- tickers
    weights <- paste0("w_", portfolio_index, "_", str_to_title(tickers))
```

```
      # Initialiser les valeurs de poids si elles ne sont pas encore définies
      for (ticker in tickers) {
       if (is.null(portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]])) {
        portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]] <- 0
       }
      }

      # Supprimer l'UI existante pour ce portefeuille
      removeUI(selector = paste0("#dynamic_weights_ui_", portfolio_index, " .pf_dynamic-weight-input"), multiple = TRUE)

      # Ajouter les nouvelles entrées de poids
      for (j in seq_along(tickers)) {
       insertUI(
        selector = paste0("#dynamic_weights_ui_", portfolio_index),
        ui = tags$div(
         fluidRow(
          # Obtenir de grosse écriture
          column(12, numericInput(weights[j], new_labels[j], value = portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", tickers[j])]], min = 0, max = 100, step = 0.001, width = '400px'))
         ),
         class = "pf_dynamic-weight-input"
        ),
        multiple = TRUE
       )
      }
     })
    })
   })
```

```
})

# Observer et mettre à jour les poids pour chaque ticker dans chaque portefeuille
observe({
  lapply(1:as.numeric(input$nb_portfolio), function(i) {
    local({
      portfolio_index <- i
      observe({
        lapply(get(paste0("pf_tickers_to_analyze_", portfolio_index))(), function(ticker) {
          observeEvent(input[[paste0("w_", portfolio_index, "_", str_to_title(ticker))]], {
            portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]] <- input[[paste0("w_", portfolio_index, "_", str_to_title(ticker))]]
          }, ignoreNULL = FALSE)
        })
      })
    })
  })
})

##################### Fin Ancienne version ##################



#################################################################
######################### Nouvelle version ######################

# # Créer un reactiveValues pour stocker les poids des portefeuilles
# portfolio_weights <- reactiveValues()
#
# # Observer le changement du nombre de portefeuilles
```

```
# observeEvent(input$nb_portfolio, {
#   # Réinitialiser le reactiveValues des poids
#   for (key in names(portfolio_weights)) {
#     portfolio_weights[[key]] <- NULL
#   }
#
#   # Nettoyer les UI des poids pour tous les portefeuilles
#   for(i in 1:50) { # nombre arbitraire suffisamment grand
#     removeUI(
#       selector = paste0("#dynamic_weights_ui_", i, " .pf_dynamic-weight-input"),
#       multiple = TRUE
#     )
#   }
# }, priority = 1000)
#
# # Observer les tickers et créer les inputs de poids
# observe({
#   lapply(1:as.numeric(input$nb_portfolio), function(i) {
#     local({
#       portfolio_index <- i
#
#       observeEvent(get(paste0("pf_tickers_to_analyze_", portfolio_index))(), {
#         tickers <- get(paste0("pf_tickers_to_analyze_", portfolio_index))()
#
#         if (is.null(tickers) || length(tickers) == 0 || (length(tickers) == 1 && tickers == "")) {
#           return(NULL)
#         }
#
#         new_labels <- tickers
```

```
#       weights <- paste0("w_", portfolio_index, "_", str_to_title(tickers))
#
#       # Initialiser les valeurs de poids
#       for (ticker in tickers) {
#         if (is.null(portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]])) {
#           portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]] <- 0
#         }
#       }
#
#       # Nettoyer l'UI existante pour ce portefeuille
#       removeUI(selector = paste0("#dynamic_weights_ui_", portfolio_index, " .pf_dynamic-weight-input"), multiple = TRUE)
#
#       # Ajouter les nouvelles entrées de poids
#       for (j in seq_along(tickers)) {
#         insertUI(
#           selector = paste0("#dynamic_weights_ui_", portfolio_index),
#           ui = tags$div(
#             fluidRow(
#               column(12, numericInput(weights[j],
#                                       new_labels[j],
#                                       value = portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", tickers[j])]],
#                                       min = 0, max = 100, step = 0.001,
#                                       width = '400px'))
#             ),
#             class = "pf_dynamic-weight-input"
#           ),
#           multiple = TRUE
```

```
#         )
#       }
#     })
#   })
# })
#
# # Observer et mettre à jour les poids
# observe({
#   lapply(1:as.numeric(input$nb_portfolio), function(i) {
#     local({
#       portfolio_index <- i
#       observe({
#         lapply(get(paste0("pf_tickers_to_analyze_", portfolio_index))(), function(ticker) {
#           observeEvent(input[[paste0("w_", portfolio_index, "_", str_to_title(ticker))]], {
#             portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]] <-
#               input[[paste0("w_", portfolio_index, "_", str_to_title(ticker))]]
#           }, ignoreNULL = FALSE)
#         })
#       })
#     })
#   })
# })


######################## Fin Nouvelle version #################
#####################################################################
# Fonctionne super bien
# Ensure metrics are calculated only after clicking the button
```

```
# Créer un reactive pour les poids qui sera mis à jour chaque fois qu'un poids change
# Reactive pour surveiller les changements de poids après leur création

# Créer un reactiveValues pour stocker les poids
portfolio_weights_rv <- reactiveValues()

# Créer un reactive pour stocker l'état des poids initialisés
weights_initialized <- reactiveVal(FALSE)

# Initialiser les poids lorsque le bouton est cliqué
observeEvent(input$pf_create_user_w, {
 weights_initialized(TRUE)

 lapply(1:as.numeric(input$nb_portfolio), function(i) {
  tickers <- get(paste0("pf_tickers_to_analyze_", i))()
  for(ticker in tickers) {
   portfolio_weights_rv[[paste0("pf_weights_values_", i, "_", ticker)]] <- 0
  }
 })
})

# Observer les changements de poids
observe({
 lapply(1:as.numeric(input$nb_portfolio), function(i) {
  tickers <- get(paste0("pf_tickers_to_analyze_", i))()
  lapply(tickers, function(ticker) {
   weight_id <- paste0("w_", i, "_", str_to_title(ticker))
   if(!is.null(input[[weight_id]])) {
    portfolio_weights_rv[[paste0("pf_weights_values_", i, "_", ticker)]] <- input[[weight_id]]
```

```
    }
   })
  })
})

# Modifier weights_reactive pour utiliser portfolio_weights_rv
weights_reactive <- reactive({
 req(weights_initialized())

 lapply(1:as.numeric(input$nb_portfolio), function(i) {
  tickers <- get(paste0("pf_tickers_to_analyze_", i))()
  sapply(tickers, function(ticker) {
   portfolio_weights_rv[[paste0("pf_weights_values_", i, "_", ticker)]]
  })
 })
})

calculate_metrics <- reactive({
# calculate_metrics <- eventReactive(input$pf_create_user_w, {
 req(weights_initialized())  # Vérifie que les poids ont été initialisés
 weights <- weights_reactive()

 lapply(1:as.numeric(input$nb_portfolio), function(i) {
  tickers <- get(paste0("pf_tickers_to_analyze_", i))()

  rf <- input[[paste0("pf_Rf_", i)]] # Old version
  # benchmark <- input[[paste0("pf_Rf_", i)]] # Old version
  rf_return <- input[[paste0("pf_rfrate_new_", i)]]
  alpha_level = as.numeric(input$alpha_level)
```

```
cat("Debug Info:\n")
cat("Tickers:", tickers, "\n")
cat("Benchmark:", rf, "\n")
cat("Risk-free Rate:", rf_return, "\n")
cat("MAR:", input$pf_minaccret, "\n")
cat("MAESG:", input$pf_minaccESG, "\n")
cat("Log Returns:", input$pf_log, "\n")

if(rf != ""){

 the_info = switch(rf,
                   # 1 year rf
                   "OAT 1y" = "FR1YT=RR",
                   'T-Note 1y' = 'US1YT=X',
                   'Bund 1y' = 'DE1YT=RR',
                   # 5 year rf
                   "OAT 5y" = "FR5YT=RR",
                   'T-Note 5y' = 'US5YT=X',
                   'Bund 5y' = 'DE5YT=RR',
                   # 10 year rf
                   "OAT 10y" = "FR10YT=RR",
                   'Bund 10y' = 'DE10YT=RR',
                   'T-Note 10y' = 'US10YT=X'
 )

 rf <- the_info
```

```
}else{
 # return(NULL)
 rf <- NULL
}

# Cas de benchmark ici car on a une idée sur l'existence du Rf qui est soit NULL ou non
benchmark <- input[[paste0("pf_BNC_", i)]] # New version

if(benchmark != ""){
 start_date <- input$pf_date[1]
 end_date <- input$pf_date[2]
 type_return <- input$pf_retfreq
 return_method <- input$pf_log

 # Utiliser ma fonction Get_bench_data pour vérifier l'existence de données
 benchmark_data  <- Get_bench_data(Name = benchmark,
                        # from = as.Date(start_date),
                        from = start_date,
                        na_method = "keep",
                        # na_method = "previous",
                        to = end_date
                        # to = as.Date(end_date)
 )

 # Vérifier s'il existe assez de données pour la data du benchmark (BNC_return)
 if (!is.null(benchmark_data) && nrow(benchmark_data) > 1) {

  colnames(benchmark_data)[2] <- 'adjusted' # Renommer par adjusted la seconde colonne
```

```
benchmark_ret <- benchmark_data %>%
  tq_transmute(select = adjusted,
               mutate_fun = periodReturn,
               period = type_return,
               col_rename = 'benchmark_ret',
               type = return_method) %>%
  tk_xts(date_var = "date", silent = TRUE)

# benchmark <- benchmark
benchmark <- list(df = benchmark_ret,
             benchmark = benchmark)

} else{

# Utiliser rf (risk_free comme benchamrk)
if(!is.null(rf)){
  benchmark <- rf

  RF_id_new <- switch(benchmark,
                      "FR1YT=RR" = 23769,
                      'DE1YT=RR' = 23684,
                      'US1YT=X' = 23700,
                      "FR5YT=RR" = 23773,
                      'DE5YT=RR' = 23688,
                      'US5YT=X' = 23703,
                      "FR10YT=RR" = 23778,
                      'DE10YT=RR' = 23693,
                      'US10YT=X' = 23705 )
```

```
    benchmark_data <- ready_hcDt(ticker_id = RF_id_new, max_retries = 30)
    benchmark_data$Date <- as.Date(benchmark_data$Date)

    # print('start_date')
    # print(start_date)
    # print(as.Date(start_date))



    benchmark_data <- dplyr::as_tibble(benchmark_data) %>%
     dplyr::filter(Date >= as.Date(start_date)) %>%
     dplyr::filter(Date <= as.Date(end_date))

    benchmark_ret <- benchmark_data %>%
     tq_transmute(select = Close,
                  mutate_fun = periodReturn,
                  period = type_return,
                  col_rename = 'benchmark_ret',
                  type = return_method) %>%
     tk_xts(date_var = "Date", silent = TRUE)

    benchmark <- list(df = benchmark_ret,
                      benchmark = benchmark)

   }else{
    benchmark <- NULL
   }
  }

} else{
```

```
      benchmark <- NULL

    }

    # tickers_ <- reactive({
    #   tickers
    # })

    # if (!is.null(tickers_()) && all(tickers_() != "")) {
    if (!is.null(tickers) && all(tickers != "")) {
      calculate_portfolio_metrics(
        # tickers = tickers_(),
        tickers = tickers,
        benchmark = benchmark, # corriger ici aussi
        # benchmark = "SPY",
        level_alpha = alpha_level,
        rf_return = rf_return,
        MAR = as.numeric(input$pf_minaccret)/100,
        MA_ESG = as.numeric(input$pf_minaccESG),
        start_date = input$pf_date[1],
        end_date = input$pf_date[2],
        type_return = input$pf_retfreq,
        return_method = input$pf_log,
        user_weights = unlist(weights[[i]])  # Utiliser les poids réactifs
        #           user_weights           =           unlist(sapply(tickers,           function(ticker)
portfolio_weights[[paste0("pf_weights_values_", i, "_", ticker)]]))
      )
    } else {
      NULL
```

```
  }
 })
})

# Combine and display metrics for all portfolios
output$portfolio_table <- renderDT({
 req(weights_initialized())
 req(weights_reactive())  # Dépendre explicitement des poids

 if(!weights_initialized()) {
  return(NULL)
 }

 metrics_list <- calculate_metrics()
 metrics_df <- do.call(rbind, lapply(seq_along(metrics_list), function(i) {
  if (!is.null(metrics_list[[i]])) {
   metrics <- metrics_list[[i]]
   df_ <- data.frame(
    Portfolio = paste("Portfolio", i),
    # tickers = paste0(),
    # Benchmark = paste0(),
    Annual_Return = metrics$pf_annual_return,
    Annual_Risk = metrics$pf_annual_risk,
    VaR = metrics$pf_var,
    Tracking_Error = metrics$pf_tracking_error,
    Beta = metrics$pf_beta,
    Beta_Bull = metrics$pf_beta_bull,
    Beta_Bear = metrics$pf_beta_bear,
    Sharpe_Ratio = metrics$pf_sharpe_ratio,
```

```
    Modified_SR = metrics$pf_Modified_SR[1],
    Info_Ratio = metrics$pf_info_ratio,
    Modified_Info_Ratio = metrics$pf_modified_info_ratio,
    Jensen_Alpha = metrics$pf_jensen_alpha,
    Modified_Jensen_Alpha = metrics$pf_modified_jensen_alpha,
    Sortino_Ratio = metrics$pf_sortino_ratio,
    Skewness = metrics$pf_skewness,
    Kurtosis = metrics$pf_kurtosis,
    JB_P.value = metrics$pf_jb.pvalue
    # ,
    # JB_Test.decision = metrics$pf_jb_test.decision
  )

  df_[, -1] <- round(df_[, -1], 6)

  # names(df_) = c("Portfolio", "Annual return", "Annual Risk",
  #               "VaR", "Tracking Error", "Beta",  "Sharpe Ratio",
  #               "Modified SR",    "Information ratio",
  #               "Modified Info ratio",      "Jensen Alpha",
  #               "Modified Jensen Alpha", "Sortino ratio")

  df_$JB.test_decision = ifelse(df_$JB_P.value > as.numeric(input$alpha_level), "Normal", "Not Normal")

  df_

 } else {
  return(NULL)
 }
```

```
 }))

 datatable(metrics_df,
        rownames = FALSE,
        options = list(scrollX = TRUE,
                columnDefs = list(list(targets = "_all",
                                className = "dt-center",
                                width = "50px"))))
})


# Combine and display detail metrics for all portfolios
output$detailed_pf_tb <- renderDT({
 req(weights_initialized())
 req(weights_reactive())  # Dépendre explicitement des poids

 if(!weights_initialized()) {
  return(NULL)
 }

 metrics_list <- calculate_metrics()
 metrics_df <- do.call(rbind, lapply(seq_along(metrics_list), function(i) {
  if (!is.null(metrics_list[[i]])) {
   metrics <- metrics_list[[i]]

   tick_list = rownames(t(metrics$metrics_by_ticker))
   # names(test) = colnames(test)
   # nrow(t(metrics$metrics_by_ticker))
```

```
    # test = (t(metrics$metrics_by_ticker))
    test = (t(metrics$metrics_by_ticker))%>%as_tibble()
    col_names_saved = names(test)
    # test$Portfolio <- rep("Portfolio1", length(tick_list))
    test$Portfolio <- paste("Portfolio", i)
    test$Ticker = tick_list

    df_ = test[,c("Portfolio", "Ticker",
                  col_names_saved)]

    names(df_) = c("Portfolio", "Ticker", "ESG",
                   "Annual_Return",    "Annual_Risk",
                   # "VaR",
                   "Beta", "Tracking_Error",     "Sharpe_Ratio",          "Modified_SR",              "Relative_SR",
#"Relative_Sharpe_Ratio"
                   "SR_with_MAESG", "Info_Ratio",     "Modified_Info_Ratio", "Jensen_Alpha",
                "Modified_Jensen_Alpha",       "Sortino_Ratio",          "Skewness",     "Kurtosis",
                "JB_P.value")

    df_$JB.test_decision = ifelse(df_$JB_P.value > as.numeric(input$alpha_level), "Normal", "Not
Normal")

    print('JB_P.value')
    print(df_$JB_P.value)

    df_

   } else {
    return(NULL)
   }
```

```
 }))

 datatable(metrics_df,
         rownames = FALSE,
         options = list(scrollX = TRUE,
                  columnDefs = list(list(targets = "_all",
                                  className = "dt-center",
                                  width = "50px")))) %>%
  formatStyle(
   'JB_P.value',
   valueColumns = 'JB_P.value',
   formatter = JS(
    "function(value) {",
    "return value.toExponential();",
    "}"
   )
  )
})



# # Display the "Portfolio comparison" section after pressing the button
# observe({
#   req(weights_initialized())  # Vérifie que les poids ont été initialisés
# # observeEvent(input$pf_create_user_w, {
#   output$portfolio_comparison_ui <- renderUI({
#    div(
#     h3("Portfolio Comparison"),
#     DTOutput("portfolio_table")
#    )
```

```
#   })
#
#   output$detailportfolio_comparison_ui <- renderUI({
#     div(
#       h3("Detailed portfolio table"),
#       DTOutput("detailed_pf_tb")
#     )
#   })
# })

# Display the "Portfolio comparison" section
observe({
 if(!weights_initialized()) {
  output$portfolio_comparison_ui <- renderUI({ NULL })
  output$detailportfolio_comparison_ui <- renderUI({ NULL })
  return()
 }

 output$portfolio_comparison_ui <- renderUI({
  div(
   h3("Portfolio Comparison"),
   DTOutput("portfolio_table")
  )
 })

 output$detailportfolio_comparison_ui <- renderUI({
  div(
   h3("Detailed portfolio table"),
   DTOutput("detailed_pf_tb")
```

```
   )
 })
})



# # TOUT CECI FOCTIONNAIT BIEN A 75%
#
# observeEvent(input$nb_portfolio, {
# # observeEvent(input$pf_construc_btn, {
#   num_portfolios <- as.numeric(input$nb_portfolio)
#
#   output$dynamic_ui <- renderUI({
#    fluidRow(
#     align = "center",
#     lapply(1:num_portfolios, function(i) {
#       column(
#         width = 3,
#         wellPanel(
#           h4(paste("Portfolio", i)),
#           radioButtons(
#             inputId = paste0("cmp_pf_stock_elm_", i),
#             label = "Stock input method",
#             choices = c("Select stock(s) directly" = "Select",
#                         "Use ISIN" = "Isin",
#                         "Search Stock(s)" = "Entrance"),
#             inline = FALSE
#           ),
#           uiOutput(paste0("pf_ui_", i)),
#           selectInput(paste0("pf_Rf_", i), strong("Select a risk free rate"), c("", risk_free_rate)),
```

```
#              checkboxInput(paste0("pf_benchmark_", i), "Use benchmark returns as risk free rate?", value = FALSE),
#           conditionalPanel(
#             condition = paste0("input.pf_benchmark_", i, " == false"),
#             numericInput(paste0("pf_rfrate_new_", i), "Daily Risk Free Rate", value = 0.02, min = -100, max = 100, step = 0.01)
#           )
#         )
#       )
#     })
#   )
# })
# # on a rendu dynamique le changement selon la sélection de "stock input method"
# observe({
#   lapply(1:as.numeric(input$nb_portfolio), function(i) {
#     # Obtenir le pf_ui pour chaque portfolio i
#     output[[paste0("pf_ui_", i)]] <- renderUI({
#       method <- input[[paste0("cmp_pf_stock_elm_", i)]]
#       if (method == "Select") {
#         selectInput(
#           paste0("portf_esg_sbl_", i), strong("Choose stock(s)"),
#           c("", esg_etf_comparison$ticker),
#           multiple = TRUE, selectize = TRUE
#         )
#       } else if (method == "Isin") {
#         textInput(
#           paste0("portf_esg_sbl_", i), strong("Enter ISIN(s)"),
```

```
#         placeholder = "Enter ISIN(s) separated by ','"
#       )
#     } else if (method == "Entrance") {
#       textInput(
#         paste0("portf_esg_sbl_", i), strong("Enter ticker(s)"),
#         placeholder = "Enter ticker(s) separated by ','"
#       )
#     }
#   })
# })
#
# # Create reactive values for each portfolio's tickers and weights
# portfolio_tickers <- reactiveValues()
# portfolio_weights <- reactiveValues()
#
# # Observe the number of portfolios to dynamically create event handlers
# observe({
#   nb_portfolio <- input$nb_portfolio
#
#   # Event to handle tickers for each portfolio
#   lapply(1:as.numeric(nb_portfolio), function(i) {
#     local({
#       portfolio_index <- i
#       ticker_event <- eventReactive(input$pf_create_user_w, {
#         method <- input[[paste0("cmp_pf_stock_elm_", portfolio_index)]]
#         tickers <- input[[paste0("portf_esg_sbl_", portfolio_index)]]
#
#         if (method == "Select") {
```

```
#         text <- tickers
#         return(text)
#       } else if (method == "Isin") {
#         user_input <- tickers
#         if (is.null(user_input) || user_input == "") {
#           return(NULL)
#         }
#         isin_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#         test <- yf_isin_to_ticker(isin_vector)
#         if (length(test) == 3) {
#           return(test$tick)
#         } else {
#           return(NULL)
#         }
#       } else {
#         user_input <- tickers
#         if (is.null(user_input) || user_input == "") {
#           return(NULL)
#         }
#         ticker_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#         return(ticker_vector)
#       }
#     })
#     assign(paste0("pf_tickers_to_analyze_", portfolio_index), ticker_event, envir = .GlobalEnv)
#   })
#  })
# })
#
# # Observe each portfolio's tickers to handle weight inputs
```

```
# observe({
#   nb_portfolio <- input$nb_portfolio
#
#   lapply(1:as.numeric(nb_portfolio), function(i) {
#     local({
#       portfolio_index <- i
#       observeEvent(get(paste0("pf_tickers_to_analyze_", portfolio_index))(), {
#         tickers <- get(paste0("pf_tickers_to_analyze_", portfolio_index))()
#
#         if (is.null(tickers) || length(tickers) == 0 || (length(tickers) == 1 && tickers == "")) {
#           return(NULL)
#         }
#
#         new_labels <- tickers
#         weights <- paste0("w_", portfolio_index, "_", str_to_title(tickers))
#
#         # Initialize weight values
#         for (ticker in tickers) {
#           if (is.null(portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]])) {
#             portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]] <- 0
#           }
#         }
#
#         removeUI(selector = paste0("#dynamic_weights_ui_", portfolio_index), multiple = TRUE)
#
#         for (j in seq_along(tickers)) {
#           insertUI(
#             selector = paste0("#dynamic_weights_ui_", portfolio_index),
#             ui = tags$div(
```

```
#           fluidRow(
#                                       column(3, numericInput(weights[j], new_labels[j], value = portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", tickers[j])]], min = 0, max = 100, step = 0.001, width = '400px'))
#           ),
#           class = "pf_dynamic-weight-input"
#         ),
#         multiple = TRUE
#       )
#     }
#   })
# })
# })
#
# # Observe and update weights for each ticker in each portfolio
# observe({
#   nb_portfolio <- input$nb_portfolio
#
#   lapply(1:as.numeric(nb_portfolio), function(i) {
#     local({
#       portfolio_index <- i
#       observe({
#         lapply(get(paste0("pf_tickers_to_analyze_", portfolio_index))(), function(ticker) {
#           observeEvent(input[[paste0("w_", portfolio_index, "_", str_to_title(ticker))]], {
#                        portfolio_weights[[paste0("pf_weights_values_", portfolio_index, "_", ticker)]] <- input[[paste0("w_", portfolio_index, "_", str_to_title(ticker))]]
#           }, ignoreNULL = FALSE)
#         })
#       })
```

```
#    })
#  })
# })
#
# # Render UI for dynamic weights input
# observeEvent(input$pf_create_user_w, {
#   output$dynamic_weights_ui <- renderUI({
#     lapply(1:as.numeric(input$nb_portfolio), function(i) {
#       div(id = paste0("dynamic_weights_ui_", i))
#     })
#   })
# })
#
# # Function to calculate metrics for each portfolio
# calculate_metrics <- reactive({
#   lapply(1:as.numeric(input$nb_portfolio), function(i) {
#     tickers <- get(paste0("pf_tickers_to_analyze_", i))()
#     benchmark <- input[[paste0("pf_Rf_", i)]]
#     rf_return <- input[[paste0("pf_rfrate_new_", i)]]
#
#     if (!is.null(tickers) && all(tickers != "")) {
#       calculate_portfolio_metrics(
#         tickers = tickers,
#         benchmark = "SPY", # For testing
#         rf_return = 0.5,   # For testing
#         MAR = input$pf_minaccret,
#         MA_ESG = input$pf_minaccESG,
#         start_date = input$pf_date[1],
#         end_date = input$pf_date[2],
```

```
#       type_return = input$pf_retfreq,
#                                                 user_weights = unlist(sapply(tickers, function(ticker) portfolio_weights[[paste0("pf_weights_values_", i, "_", ticker)]]))
#     )
#   } else {
#     NULL
#   }
#  })
# })
#
# # Combine and display metrics for all portfolios
# output$portfolio_table <- renderDT({
#   metrics_list <- calculate_metrics()
#
#   # Faire du rbind
#   metrics_df <- do.call(rbind, lapply(seq_along(metrics_list), function(i) {
#     if (!is.null(metrics_list[[i]])) {
#       metrics <- metrics_list[[i]]
#       data.frame(
#         Portfolio = paste("Portfolio", i),
#         Annual_Return = metrics$pf_annual_return,
#         Annual_Risk = metrics$pf_annual_risk,
#         VaR = metrics$pf_var,
#         Tracking_Error = metrics$pf_tracking_error,
#         Beta = metrics$pf_beta,
#         Sharpe_Ratio = metrics$pf_sharpe_ratio,
#         Modified_SR = metrics$pf_Modified_SR[1],
#         Info_Ratio = metrics$pf_info_ratio,
#         Modified_Info_Ratio = metrics$pf_modified_info_ratio,
```

```
#       Jensen_Alpha = metrics$pf_jensen_alpha,
#       Modified_Jensen_Alpha = metrics$pf_modified_jensen_alpha,
#       Sortino_Ratio = metrics$pf_sortino_ratio
#     )
#    } else {
#     return(NULL)
#    }
#  }))
#
#  datatable(metrics_df, options = list(scrollX = TRUE))
# })



# Le code ci-dessous est l'avant dernier code et il fonctionnait à 50%


# # Create reactive values for each portfolio's tickers and weights
# portfolio_tickers <- reactiveValues()
# portfolio_weights <- reactiveValues()
#
# observeEvent(input$pf_create_user_w, {
#  lapply(1:as.numeric(input$nb_portfolio), function(i) {
#   assign(paste0("pf_tickers_to_analyze_", i), eventReactive(input$pf_create_user_w, {
#     method <- input[[paste0("cmp_pf_stock_elm_", i)]]
#     tickers <- input[[paste0("portf_esg_sbl_", i)]]
#
#     if (method == "Select") {
#       text <- tickers
#       return(text)
#     } else if (method == "Isin") {
```

```
#       user_input <- tickers
#       if (is.null(user_input) || user_input == "") {
#         return(NULL)
#       }
#       isin_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#       test <- yf_isin_to_ticker(isin_vector)
#       if (length(test) == 3) {
#         return(test$tick)
#       } else {
#         return(NULL)
#       }
#     } else {
#       user_input <- tickers
#       if (is.null(user_input) || user_input == "") {
#         return(NULL)
#       }
#       ticker_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#       return(ticker_vector)
#     }
#   }), envir = .GlobalEnv)
#  })
# })
#
# observe({
#  lapply(1:as.numeric(input$nb_portfolio), function(i) {
#   if (exists(paste0("pf_tickers_to_analyze_", i), envir = .GlobalEnv)) {
#     observeEvent(get(paste0("pf_tickers_to_analyze_", i))(), {
#       tickers <- get(paste0("pf_tickers_to_analyze_", i))()
#
```

```
#       if (is.null(tickers) || length(tickers) == 0 || (length(tickers) == 1 && tickers == "")) {
#         return(NULL)
#       }
#
#       new_labels <- tickers
#       weights <- paste0("w_", i, "_", str_to_title(tickers))
#
#       for (ticker in tickers) {
#         if (is.null(portfolio_weights[[paste0("pf_weights_values_", i, "_", ticker)]])) {
#           portfolio_weights[[paste0("pf_weights_values_", i, "_", ticker)]] <- 0
#         }
#       }
#
#       removeUI(selector = paste0("#dynamic_weights_ui_", i), multiple = TRUE)
#
#       for (j in seq_along(tickers)) {
#         insertUI(
#           selector = paste0("#dynamic_weights_ui_", i),
#           ui = tags$div(
#             fluidRow(
#                                             column(3, numericInput(weights[j], new_labels[j], value =
portfolio_weights[[paste0("pf_weights_values_", i, "_", tickers[j])]], min = 0, max = 100, step = 0.001,
width = '400px'))
#             ),
#             class = "pf_dynamic-weight-input"
#           ),
#           multiple = TRUE
#         )
#       }
```

```
#     })
#   }
#  })
# })
#
# # observe({
# observeEvent(input$pf_create_user_w,{
#  lapply(1:as.numeric(input$nb_portfolio), function(i) {
#   if (exists(paste0("pf_tickers_to_analyze_", i), envir = .GlobalEnv)) {
#     lapply(get(paste0("pf_tickers_to_analyze_", i))(), function(ticker) {
#       observeEvent(input[[paste0("w_", i, "_", str_to_title(ticker))]], {
#         portfolio_weights[[paste0("pf_weights_values_", i, "_", ticker)]] <- input[[paste0("w_", i, "_",
str_to_title(ticker))]]
#       }, ignoreNULL = FALSE)
#     })
#   }
#  })
# })
#
# observeEvent(input$pf_create_user_w, {
#  output$dynamic_weights_ui <- renderUI({
#   lapply(1:as.numeric(input$nb_portfolio), function(i) {
#     div(id = paste0("dynamic_weights_ui_", i))
#   })
#  })
# })
#
# calculate_metrics <- reactive({
#  lapply(1:as.numeric(input$nb_portfolio), function(i) {
```

```
#    if (exists(paste0("pf_tickers_to_analyze_", i), envir = .GlobalEnv)) {
#     tickers <- get(paste0("pf_tickers_to_analyze_", i))()
#     benchmark <- input[[paste0("pf_Rf_", i)]]
#     rf_return <- input[[paste0("pf_rfrate_new_", i)]]
#
#     if(benchmark != ""){
#
#      the_info = switch(benchmark,
#                  # 1 year benchmark
#                  "OAT 1y" = "FR1YT=RR",
#                  'T-Note 1y' = 'US1YT=X',
#                  'Bund 1y' = 'DE1YT=RR',
#                  # 5 year benchmark
#                  "OAT 5y" = "FR5YT=RR",
#                  'T-Note 5y' = 'US5YT=X',
#                  'Bund 5y' = 'DE5YT=RR',
#                  # 10 year benchmark
#                  "OAT 10y" = "FR10YT=RR",
#                  'Bund 10y' = 'DE10YT=RR',
#                  'T-Note 10y' = 'US10YT=X'
#      )
#
#      benchmark <- the_info
#
#     }else{
#      # return(NULL)
#      benchmark <- NULL
#     }
#
```

```
#
#     print("The unlist ticker process")
#      print(unlist(sapply(tickers, function(ticker) portfolio_weights[[paste0("pf_weights_values_", i, "_", ticker)]])))
#
#     # if (!is.null(tickers)) {
#     if (!is.null(tickers) && all(tickers != "")) {
#       calculate_portfolio_metrics(
#         tickers = tickers,
#         benchmark = "SPY",
#         # benchmark = benchmark, #A corriger après
#         rf_return = 0.5,
#         MAR = input$pf_minaccret,
#         MA_ESG = input$pf_minaccESG,
#         start_date = input$pf_date[1],
#         end_date = input$pf_date[2],
#         type_return = input$pf_retfreq,
#                                             user_weights = unlist(sapply(tickers, function(ticker) portfolio_weights[[paste0("pf_weights_values_", i, "_", ticker)]]))
#       )
#     } else {
#       NULL
#     }
#   } else {
#     NULL
#   }
#  })
# })
#
```

```
# output$portfolio_table <- renderDT({
#   # A verifier et enler après
#   print("--longueur calculate_metrics() --")
#   print(length(calculate_metrics()))
#
#   print("-- la dimension de calculate_metrics() -- ")
#   print(dim(calculate_metrics()))
#   # if(!is.null(calculate_metrics())){
#   #   metrics_list <- calculate_metrics()
#   #
#   #   metrics_df <- do.call(rbind, lapply(seq_along(metrics_list), function(i) {
#   #     if (!is.null(metrics_list[[i]])) {
#   #       metrics <- metrics_list[[i]]
#   #       df_ = data.frame(
#   #         Portfolio = paste("Portfolio", i),
#   #         Annual_Return = metrics$pf_annual_return,
#   #         Annual_Risk = metrics$pf_annual_risk,
#   #         VaR = metrics$pf_var,
#   #         Tracking_Error = metrics$pf_tracking_error,
#   #         Beta = metrics$pf_beta,
#   #         Sharpe_Ratio = metrics$pf_sharpe_ratio,
#   #         Modified_SR = metrics$pf_Modified_SR[1],
#   #         Info_Ratio = metrics$pf_info_ratio,
#   #         Modified_Info_Ratio = metrics$pf_modified_info_ratio,
#   #         Jensen_Alpha = metrics$pf_jensen_alpha,
#   #         Modified_Jensen_Alpha = metrics$pf_modified_jensen_alpha,
#   #         Sortino_Ratio = metrics$pf_sortino_ratio
#   #       )
#   #
```

```
#  #     df_[, -1] = round(df_[, -1], 6)
#  #
#  #     df_
#  #
#  #    } #else {
#  #     #return(NULL)
#  #    #}
#  #    print(metrics_df)
#  #    datatable(metrics_df, rownames = FALSE, options = list(scrollX = TRUE))
#  #
#  #  }
#  #  ))
#  # }
#
#   metrics_list <- calculate_metrics()
#
#   metrics_df <- do.call(rbind, lapply(seq_along(metrics_list), function(i) {
#    if (!is.null(metrics_list[[i]])) {
#      metrics <- metrics_list[[i]]
#      df_ = data.frame(
#        Portfolio = paste("Portfolio", i),
#        Annual_Return = metrics$pf_annual_return,
#        Annual_Risk = metrics$pf_annual_risk,
#        VaR = metrics$pf_var,
#        Tracking_Error = metrics$pf_tracking_error,
#        Beta = metrics$pf_beta,
#        Sharpe_Ratio = metrics$pf_sharpe_ratio,
#        Modified_SR = metrics$pf_Modified_SR[1],
#        Info_Ratio = metrics$pf_info_ratio,
```

```
#       Modified_Info_Ratio = metrics$pf_modified_info_ratio,
#       Jensen_Alpha = metrics$pf_jensen_alpha,
#       Modified_Jensen_Alpha = metrics$pf_modified_jensen_alpha,
#       Sortino_Ratio = metrics$pf_sortino_ratio
#     )
#
#     df_[, -1] = round(df_[, -1], 6)
#
#     df_
#
#   } else {
#     return(NULL)
#   }
# }
# ))
#
# print(metrics_df)
# datatable(metrics_df, rownames = FALSE, options = list(scrollX = TRUE))
# })



##############################
##############################

# observe({
#  updateSelectInput(session, "exp_stock1",
#                label = "Choose Stock 1 from exported file:",
#                choices = get_items_list()[1])
```

```
# })

# observe({
#   if (input$navbar == "Portfolio") {
#     current_panel <- input$portfolioNavset
#     if (!is.null(current_panel) && current_panel == "PortfolioComparison") {
#       sidebar_toggle(id = "sidebar", open = FALSE)
#       # shinyjs::runjs("document.querySelector('.collapse-toggle').style.pointerEvents = 'none';")
#       shinyjs::runjs("document.querySelector('.collapse-toggle').style.visibility = 'hidden';")
#     } else {
#       sidebar_toggle(id = "sidebar", open = TRUE)
#       shinyjs::runjs("document.querySelector('.collapse-toggle').style.visibility = 'visible';")
#       # shinyjs::runjs("document.querySelector('.collapse-toggle').style.pointerEvents = 'auto';")
#     }
#   } else {
#     sidebar_toggle(id = "sidebar", open = TRUE)
#     shinyjs::runjs("document.querySelector('.collapse-toggle').style.visibility = 'visible';")
#     # shinyjs::runjs("document.querySelector('.collapse-toggle').style.pointerEvents = 'auto';")
#   }
# })

observe({
  if (input$navbar == "Portfolio") {
    current_panel <- input$portfolioNavset
    if (!is.null(current_panel) && current_panel == "PortfolioComparison") {
      sidebar_toggle(id = "sidebar", open = FALSE)
      shinyjs::runjs("
        document.querySelector('.collapse-toggle').style.pointerEvents = 'none';
        document.querySelector('.collapse-toggle').style.visibility = 'hidden';
```

```
    ")
  } else {
    sidebar_toggle(id = "sidebar", open = TRUE)
    shinyjs::runjs("
      document.querySelector('.collapse-toggle').style.pointerEvents = 'auto';
      document.querySelector('.collapse-toggle').style.visibility = 'visible';
    ")
  }
 } else {
  sidebar_toggle(id = "sidebar", open = TRUE)
  shinyjs::runjs("
    document.querySelector('.collapse-toggle').style.pointerEvents = 'auto';
    document.querySelector('.collapse-toggle').style.visibility = 'visible';
  ")
 }
})

# observe({
#   sidebar_toggle(
#     id = "sidebar",
#     open = input$navbar == "Portfolio comparison"
#   )
# })



# observe({
#   if (input$navbar == "Portfolio") {
#     current_panel <- input$portfolioNavset
#     if (!is.null(current_panel) && current_panel == "PortfolioComparison") {
```

```
#     sidebar_toggle(id = "sidebar", open = FALSE)
#     shinyjs::runjs("document.querySelector('.collapse-toggle').style.pointerEvents = 'none';")
#   } else {
#     sidebar_toggle(id = "sidebar", open = TRUE)
#     shinyjs::runjs("document.querySelector('.collapse-toggle').style.pointerEvents = 'auto';")
#   }
#  } else {
#   sidebar_toggle(id = "sidebar", open = TRUE)
#   shinyjs::runjs("document.querySelector('.collapse-toggle').style.pointerEvents = 'auto';")
#  }
# })

observeEvent(input$retfreq, {
 new_label <- paste0(str_to_title(input$retfreq), " Risk Free Rate")
 updateNumericInput(session, "rfrate_new", label = new_label)
})

observeEvent(input$dark_mode, {
 if (input$dark_mode == "dark") {
  showNotification("Welcome to the dark side!")
 }
})



####################
# Partie comparison

# Mettre à jour le choix de l'utilisateur
output$cmp_esg_sbl_ui <- renderUI({
```

```
 if (input$stock_elm == "Select") {
  selectInput(
   "cmp_esg_sbl", strong("Choose stock(s)"),
   c("", esg_etf_comparison$ticker),
   multiple = TRUE,
   selected = '',
   selectize = TRUE
  )
 } else if (input$stock_elm == "Isin") {
  textInput(
   "cmp_esg_sbl", strong("Enter ISIN(s)"),
   placeholder = "Enter ISIN(s) separated by commas"
  )
 } else if (input$stock_elm == "Entrance") {
  textInput(
   "cmp_esg_sbl", strong("Enter ticker(s)"),
   placeholder = "Enter ticker(s) separated by commas"
  )
 }
})

# reactive({})

# the_cmp_esg_sbl <- eventReactive(input$run_button, {
# # the_cmp_esg_sbl <- reactive({
# # observe({
#   if(input$stock_elm == "Select"){
#    text = input$cmp_esg_sbl
#    print(text)
```

```
#    return(text)
#
#  }else if (input$stock_elm == "Isin"){
#    user_input <- input$cmp_esg_sbl
#
#    print(user_input)
#
#    if (is.null(user_input) || user_input == "") {
#      return("")
#    }
#
#
#    # Divisez la chaîne d'entrée par des virgules et supprimez les espaces en début et fin
#    isin_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#
#    test = yf_isin_to_ticker(isin_vector)
#
#
#    if(length(test) == 3 ){
#      print(test$tick)
#      return(test$tick)
#    } else{
#      return("")
#    }
#
#  }else{
#    user_input <- input$cmp_esg_sbl
#
#    if (is.null(user_input) || user_input == "") {
```

```
#     return("")
#   }
#
#   # Divisez la chaîne d'entrée par des virgules et supprimez les espaces en début et fin
#   ticker_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#
#   return(ticker_vector)
#  }
# })



# Déclaration d'une valeur réactive
the_cmp_esg_sbl <- reactiveVal()

# Utilisation d'observeEvent pour capturer l'événement du bouton
observeEvent(input$run_button, {
 if (input$stock_elm == "Select") {
  text <- input$cmp_esg_sbl
  print(text)
  the_cmp_esg_sbl(text) # Mise à jour de la valeur réactive

 } else if (input$stock_elm == "Isin") {
  user_input <- input$cmp_esg_sbl
  print(user_input)

  if (is.null(user_input) || user_input == "") {
   the_cmp_esg_sbl("")
  } else {
   isin_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
```

```
    test <- yf_isin_to_ticker(isin_vector)

    if (length(test) == 3) {
      print(test$tick)
      the_cmp_esg_sbl(test$tick)
    } else {
      the_cmp_esg_sbl("")
    }
   }

  } else {
   user_input <- input$cmp_esg_sbl

   if (is.null(user_input) || user_input == "") {
    the_cmp_esg_sbl("")
   } else {
    ticker_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
    the_cmp_esg_sbl(ticker_vector)
   }
  }
})

# Utilisation de the_cmp_esg_sbl() pour accéder à la valeur dans d'autres parties de ton code

cmp_tools <- reactive({
  # selected_tickers <- input$cmp_esg_sbl
  selected_tickers <- the_cmp_esg_sbl()

  # if (length(input$cmp_esg_sbl) == 1 && input$cmp_esg_sbl == "") {
```

```
 #   return(NULL)
 #
 # }

 if (is.null(selected_tickers) || length(selected_tickers) == 0 || all(selected_tickers == "")) {
   return(NULL)

 } else {
   print("je teste le selected_tickers")
   print(selected_tickers)
   # df = get_esg_data(input$cmp_esg_sbl)
   df = get_esg_data(selected_tickers)
   return(df)

 }

 # else if (input$cmp_esg_sbl == "") {
 #   return(NULL)
 # }
})

# cmp_tools <- reactive({
#   if (length(input$cmp_esg_sbl) == 1 && input$cmp_esg_sbl == "") {
#     return(NULL)
#
#   }else{
#     df = get_esg_data(input$cmp_esg_sbl)
#
#     return(df)
```

```
#   }
# })


# Sustainability
# Render the table with kableExtra
observe({
  if (!is.null(cmp_tools())) {
    output$funds_cmp_table <- renderText({
      cmp_tools() %>%
        kableExtra::kable() %>%
        kableExtra::kable_styling(full_width = TRUE,
                                  font_size = 15,
                                  bootstrap_options = c("striped",
                                                        "hover",
                                                        "condensed"),
                                  position = "center"#,
                                  # latex_options = "hold_position",
        ) %>%
        kableExtra::pack_rows("ESG Risk and Performance", 1, 5 ,
                              label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;"))%>%
        kableExtra::pack_rows("Controversy", 6, 9,
                              label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;"))%>%
        # label_row_css = "background-color: #666; color: #fff;") %>%
        kableExtra::pack_rows("Product Involvement Areas", 10, 23,
                              label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;"))
      # label_row_css = "background-color: #666; color: #fff;")
      # %>%
      #   column_spec(column = 1:7, border_left = "1px solid black", border_right = "1px solid black",
extra_css = "padding: 8px;")
```

```
  })

 } else {
  output$funds_cmp_table <- renderText({ 'For a comparative overview, please select stocks' })
 }
})


# Key stats
stats_tools <- reactive({
 # selected_tickers <- input$cmp_esg_sbl
 selected_tickers <- the_cmp_esg_sbl()

 # if (length(input$cmp_esg_sbl) == 1 && input$cmp_esg_sbl == "") {
 #   return(NULL)
 #
 # }

 # Ceci fonctionnait
 # if (is.null(selected_tickers) || length(selected_tickers) == 0 || all(selected_tickers == "")) {
 #   return(NULL)
 # }
 if (all(selected_tickers == "")) {
  return(NULL)
 } else {
  # df = yf_get_keys.stats(input$cmp_esg_sbl, full = FALSE)
  df = yf_get_keys.stats(selected_tickers, full = FALSE)
  return(df)
```

```
 }

})


# Render the table with kableExtra
observe({
 if (!is.null(stats_tools())) {
  # output$K_Stats_cmp_table <- renderText({
  #
  #  df = stats_tools()
  #
  #  # Ajouter les footnotes aux rownames du dataframe
  #  df_with_footnotes <- add_footnotes(df, footnotes)
  #
  #  df_with_footnotes %>%
  #   kable(booktabs = TRUE, escape = FALSE) %>%
  #   kable_styling(full_width = TRUE,
  #                 font_size = 14,
  #                 bootstrap_options = c("striped",
  #                                       "hover",
  #                                       "condensed"),
  #                 position = "center",
  #                 latex_options = "hold_position"
  #   )%>%
  #   footnote(general = paste(footnotes, collapse = "\n"))%>%
  #   # footnote(general = paste(footnotes, collapse = "<br>"))%>%
  #   kableExtra::pack_rows("Current Valuation Measures ", 1, 9 ,
```

```
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Financial Highlights", 10, 31,
#           label_row_css = "background-color: #666; color: #fff;")%>%
#           # label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Fiscal Year", 10, 11,
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Profitability", 12, 13,
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Management Effectiveness", 14, 15,
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Income Statement", 16, 23,
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Balance Sheet", 24, 29,
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Cash Flow Statement", 30, 31,
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Trading Information", 32, 60,
#           label_row_css = "background-color: #666; color: #fff;")%>%
#           # label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Stock Price History", 32, 38,
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Share Statistics", 39, 50,
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
#    kableExtra::pack_rows("Dividends & Splits", 51, 60,
#           label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;"))
#
# })

output$V_Stats_cmp_table <- renderText({
```

```
  df = stats_tools()

  df = df$val_measure

  df %>%
    kableExtra::kable(booktabs = TRUE, escape = FALSE) %>%
    kableExtra::kable_styling(full_width = TRUE,
                              font_size = 14,
                              bootstrap_options = c("striped",
                                                    "hover",
                                                    "condensed"),
                              position = "center",
                              latex_options = "hold_position"
    )

})


output$F_Stats_cmp_table <- renderUI({
  # output$F_Stats_cmp_table <- renderText({

  df = stats_tools()$f_Highlights

  # Ajouter les footnotes aux rownames du dataframe
  # df_with_footnotes <- add_footnotes(df, footnotes)
  # df_with_footnotes <- add_footnotes_F(df, footnotes_Financial)
  df_with_footnotes <- add_footnotes_F(df)
```

```
df_with_footnotes %>%
 # kableExtra::kable(booktabs = TRUE, escape = FALSE) %>%
 knitr::kable("html", booktabs = TRUE, escape = FALSE) %>%
 kableExtra::kable_styling(full_width = TRUE,
                           font_size = 14,
                           bootstrap_options = c("striped",
                                                 "hover",
                                                 "condensed"),
                           position = "center",
                           latex_options = "hold_position"
 )%>%
 # footnote
 kableExtra::footnote(general = paste(footnotes_Financial, collapse = "\n"))%>%
 # kableExtra::pack_rows("Financial Highlights", 10, 31,
 #         label_row_css = "background-color: #666; color: #fff;")%>%
 # label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
 kableExtra::pack_rows("Fiscal Year", 1, 2,
                       label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
 kableExtra::pack_rows("Profitability", 3, 4,
                       label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
 kableExtra::pack_rows("Management Effectiveness", 5, 6,
                       label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
 kableExtra::pack_rows("Income Statement", 7, 14,
                       label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
 kableExtra::pack_rows("Balance Sheet", 15, 20,
                       label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
 kableExtra::pack_rows("Cash Flow Statement", 21, 22,
                       label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;"))%>%
 htmltools::HTML()%>%  # Force le rendu HTML final
```

```
  htmltools::tagList()  # Encapsule dans une balise HTML pour Shiny

})

output$T_Stats_cmp_table <- renderUI({
 # output$T_Stats_cmp_table <- renderText({

 df = stats_tools()$t_Information

 # Ajouter les footnotes aux rownames du dataframe
 df_with_footnotes <- add_footnotes(df, footnotes)

 df_with_footnotes %>%
  knitr::kable("html", booktabs = TRUE, escape = FALSE) %>%
  # kableExtra::kable(booktabs = TRUE, escape = FALSE) %>%
  kableExtra::kable_styling(full_width = TRUE,
                font_size = 14,
                bootstrap_options = c("striped",
                          "hover",
                          "condensed"),
                position = "center",
                latex_options = "hold_position"
  )%>%
  # footnote
  kableExtra::footnote(general = HTML(paste(footnotes, collapse = "\n")))%>%
  # footnote(general = paste(footnotes, collapse = "<br>"))%>%
  # kableExtra::pack_rows("Trading Information", 32, 60,
  #         label_row_css = "background-color: #666; color: #fff;")%>%
  # label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
```

```
    kableExtra::pack_rows("Stock Price History", 1, 7,
                label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
    kableExtra::pack_rows("Share Statistics", 8, 19,
                label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;")) %>%
    kableExtra::pack_rows("Dividends & Splits", 20, 29,
                label_row_css = paste0("background-color: ", tbl_colour(), "; color: #fff;"))%>%
    htmltools::HTML()%>%  # Force le rendu HTML final
    htmltools::tagList()  # Encapsule dans une balise HTML pour Shiny


  })

  output$cur_v <- renderHighchart({
   current_values <- stats_tools()$val_measure %>%
    rownames_to_column("Info") %>%
    pivot_longer(cols = -Info, names_to = "Ticker", values_to = "Value")

   # current_values$Value = gsub("--", NA, current_values$Value)
   current_values$Value = gsub("--", '', current_values$Value)

   current_values <- current_values %>%
    mutate(Value = purrr::map_dbl(Value, convert_to_numeric))

   M_and_enterprise <- current_values[current_values$Info %in% c('Market Cap', 'Enterprise Value'), ]

   comp_graph_plot(M_and_enterprise , "Market cap and Enterprise Value by ticker")

   # other_current = current_values[current_values$Info != 'Market Cap' & current_values$Info != 'Enterprise Value', ]
```

```
  # comp_graph_plot(other_current , "Other Valuation Measures by ticker")

})

f_highlights <- reactive({
  df <- stats_tools()$f_Highlights

  # print(df)
  # Enlever les deux premières lignes
  df <- df %>%
    rownames_to_column("Info") %>%
    pivot_longer(cols = -Info, names_to = "Ticker", values_to = "Value")

  f_highlights <- df[-c(1,2), ]

  f_highlights$Value = gsub("--", '', f_highlights$Value)

  f_highlights <- f_highlights%>%
    mutate(Value = purrr::map_dbl(Value, convert_to_numeric))

  f_highlights

})

output$prof <- renderHighchart({
  f_highlights = f_highlights()
  Profitability <- f_highlights[f_highlights$Info %in%
                                  c("Profit Margin", "Operating Margin  (ttm)"), ]
```

```
  comp_graph_plot(Profitability , "Profitability comparison")

})

output$INC_st <- renderHighchart({
  f_highlights = f_highlights()

  # Inc_stat1 <- f_highlights[f_highlights$Info %in%
  #                     c("Revenue Per Share  (ttm)",
  #                       "Quarterly Revenue Growth  (yoy)",
  #                       "Diluted EPS  (ttm)",
  #                       "Quarterly Earnings Growth  (yoy)"), ]

  # comp_graph_plot( Inc_stat1, "Other Income Statement elements")

  Inc_stat <- f_highlights[f_highlights$Info %in%
                       c("Revenue  (ttm)",
                         "Gross Profit  (ttm)",
                         "EBITDA",
                         "Net Income Avi to Common  (ttm)"), ]

  comp_graph_plot( Inc_stat, "Income Statement : Revenue, EBITDA and Net Income")



})

output$BS <- renderHighchart({
  f_highlights = f_highlights()
```

```
 # BS1 <- f_highlights[f_highlights$Info %in%
 #                  c("Total Cash Per Share  (mrq)",
 #                    "Total Debt/Equity  (mrq)",
 #                    "Current Ratio  (mrq)",
 #                    "Book Value Per Share  (mrq)"), ]

 # comp_graph_plot(BS1 , "Other Balance Sheet element")



 BS <- f_highlights[f_highlights$Info %in%
                  c("Total Cash  (mrq)",
                    "Total Debt  (mrq)"), ]

 comp_graph_plot(BS , "Balance Sheet")



})

output$Cashfl <- renderHighchart({
 f_highlights = f_highlights()

 Cf_Stat <- f_highlights[f_highlights$Info %in%
                    c("Operating Cash Flow  (ttm)",
                      "Levered Free Cash Flow  (ttm)"), ]

 comp_graph_plot(Cf_Stat, "Cash flow statement")



})
```

```
  } else {
    # output$K_Stats_cmp_table <- renderText({ 'For a comparative Key Statistics, please select stocks' })

    output$V_Stats_cmp_table <- renderText({ 'For a comparative Key Statistics, please select stocks' })

    output$F_Stats_cmp_table <- renderUI({NULL})
    # output$F_Stats_cmp_table <- renderText({''})

    output$T_Stats_cmp_table <- renderText({''})

    output$cur_v <- renderHighchart({ return(NULL)})

    output$prof <- renderHighchart({ return(NULL)})
    output$INC_st <- renderHighchart({ return(NULL)})
    output$BS <- renderHighchart({ return(NULL)})
    output$Cashfl <- renderHighchart({ return(NULL)})

  }
 })

 # new_cmp_esg_sbl <- reactive({
 #   selected_tickers <- the_cmp_esg_sbl()
 #
 #   if (is.null(selected_tickers) || length(selected_tickers) == 0 || all(selected_tickers == "")) {
```

```
#    return(NULL)
#
#  }
#
#  # if(input$stock_elm == "Select"){
#  #   tic_vect = selected_tickers
#  #
#  #   data = yf.get_long_names(tic_vect)
#  #
#  #   # print(text)
#  #   return(data)
#  #
#  # } #else if
#  if (input$stock_elm == "Isin"){
#    user_input <- input$cmp_esg_sbl
#
#    # tic_vect = selected_tickers
#
#    # print(user_input)
#
#    if (is.null(selected_tickers) || length(selected_tickers) == 0 || all(selected_tickers == "")) {
#      return(NULL)
#
#    }
#
#
#    # if (is.null(user_input) || user_input == "") {
#    #   return(NULL)
#    # } else{
```

```
#    #
#    # }
#
#    # Divisez la chaîne d'entrée par des virgules et supprimez les espaces en début et fin
#    isin_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#
#    data = yf_isin_to_ticker(isin_vector)
#
#    if(length(data) == 3 ){
#      return(data$df)
#
#    } else{
#      return(NULL)
#    }
#
#  }else{
#    # user_input <- input$cmp_esg_sbl
#    #
#    # if (is.null(user_input) || user_input == "") {
#    #   return("")
#    # }
#    #
#    # # Divisez la chaîne d'entrée par des virgules et supprimez les espaces en début et fin
#    # ticker_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#    #
#    # return(ticker_vector)
#
#    tic_vect = selected_tickers
#
```

```
#    data = yf.get_long_names(tic_vect)
#
#    if(is.null(dim(data))){
#      data <- data.frame(
#        Ticker = names(tic_vect),
#        `Long name` = unlist(data),
#        stringsAsFactors = FALSE
#      )
#
#    }
#
#    return(data)
#
#  }
#
# })



new_cmp_esg_sbl <- reactive({
 selected_tickers <- the_cmp_esg_sbl()

 if (is.null(selected_tickers) || length(selected_tickers) == 0 || all(selected_tickers == "")) {
  return(NULL)

 }

 if (input$stock_elm == "Isin"){
  user_input <- input$cmp_esg_sbl
```

```
 # tic_vect = selected_tickers

 # print(user_input)

 if (is.null(selected_tickers) || length(selected_tickers) == 0 || all(selected_tickers == "")) {
  return(NULL)

 }

 isin_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]

 data = yf_isin_to_ticker(isin_vector)

 if(length(data) == 3 ){
  return(data$df)

 } else{
  return(NULL)
 }

}else{
 tic_vect = selected_tickers

 data = yf.get_long_names(tic_vect)

 if(is.null(dim(data))){
  data <- data.frame(
   Ticker = names(tic_vect),
   `Long name` = unlist(data),
```

```
      stringsAsFactors = FALSE
    )

  }

  return(data)

 }

})


output$ticker_and_long.names <- renderDT({
 if(is.null(new_cmp_esg_sbl())){

  return(NULL)

 } else{

  Data <- new_cmp_esg_sbl()

  Data <- DT::datatable(Data, rownames = FALSE,
                        extensions = 'Buttons', options = list(
                         # dom = 'Bfrtip',
                         buttons = list('copy',
                                        'print',
                                        list(extend = 'collection',
                                             buttons = c('csv', 'excel', 'pdf'),
                                             text = 'Download'), I('colvis')),
```

```
                    # buttons = c('copy', 'print', 'csv'),
                    columnDefs = list(list(targets = "_all", className = "dt-left",
                                           width = "35px"))
                  )
  )
  return(Data)

 }

})




################
# partie portofolio

output$db_esg_sbl_ui <- renderUI({
 if (input$pf_stock_elm == "Select") {
  selectInput(
   "db_esg_sbl", strong("Choose stock(s)"),
   c("", esg_etf_comparison$ticker),
   multiple = TRUE,
   selected = '',
   selectize = TRUE
  )
 } else if (input$pf_stock_elm == "Isin") {
  textInput(
   "db_esg_sbl", strong("Enter ISIN(s)"),
```

```
    # placeholder = "Enter ISIN(s) separated by commas"
    placeholder = "ISIN(s) separated by commas"
   )
  } else if (input$pf_stock_elm == "Entrance") {
   textInput(
    "db_esg_sbl", strong("Enter ticker(s)"),
    # placeholder = "Enter ticker(s) separated by commas"
    placeholder = "Ticker(s) separated by commas"
   )
  }
})

tickers_to_analyze <- eventReactive(input$pf_run_button, {

  if (input$pf_stock_elm == "Select") {

   text = input$db_esg_sbl
   return(text)
  } else if (input$pf_stock_elm == "Isin") {

   # if(length(input$db_esg_sbl) == 1 && input$db_esg_sbl == ""){
   #   return(NULL)
   # }
   user_input <- input$db_esg_sbl

   if (is.null(user_input) || user_input == "") {
    return(NULL)
    # return("")
```

```
    }

    isin_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
    test <- yf_isin_to_ticker(isin_vector)

    if (length(test) == 3) {
      return(test$tick)
    } else {
      return(NULL)
      # return("")
    }
  } else {

    # if(length(input$db_esg_sbl) == 1 && input$db_esg_sbl == ""){
    #   return(NULL)
    # }
    user_input <- input$db_esg_sbl

    if (is.null(user_input) || user_input == "") {
      return(NULL)
      # return("")
    }

    ticker_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
    return(ticker_vector)
  }
})
```

```
# new_tickers_to_analyze <- reactive({
#   if (input$pf_stock_elm == "Select") {
#     # removeUI(selector = ".dynamic-weight-input", multiple = TRUE)
#     text = input$db_esg_sbl
#     return(text)
#   } else if (input$pf_stock_elm == "Isin") {
#     # removeUI(selector = ".dynamic-weight-input", multiple = TRUE)
#
#     # if(length(input$db_esg_sbl) == 1 && input$db_esg_sbl == ""){
#     #   return(NULL)
#     # }
#     user_input <- input$db_esg_sbl
#
#     if (is.null(user_input) || user_input == "") {
#
#       return(NULL)
#       # return("")
#
#     }
#
#     isin_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#     test <- yf_isin_to_ticker(isin_vector)
#
#     if (length(test) == 3) {
#       return(test$tick)
#     } else {
#       return(NULL)
#       # return("")
#     }
```

```
#   } else {
#     # removeUI(selector = ".dynamic-weight-input", multiple = TRUE)
#
#     # if(length(input$db_esg_sbl) == 1 && input$db_esg_sbl == ""){
#     #   return(NULL)
#     # }
#     user_input <- input$db_esg_sbl
#
#     if (is.null(user_input) || user_input == "") {
#       return(NULL)
#       # return("")
#     }
#
#     ticker_vector <- base::strsplit(gsub(" ", "", user_input), ",")[[1]]
#     return(ticker_vector)
#   }
# })




# Le Code fonctionne super bien
# Variable pour stocker les poids

weights_values <- reactiveValues()

observeEvent(tickers_to_analyze(), {
  removeUI(selector = ".dynamic-weight-input", multiple = TRUE)

  tickers <- tickers_to_analyze()
```

```
  if (is.null(tickers) || length(tickers) == 0 || (length(tickers) == 1 && tickers == "")) {
   return(NULL)
  }

  new_labels <- tickers
  weights <- paste0("w_", str_to_title(tickers))

  for (ticker in tickers) {
   if (is.null(weights_values[[ticker]])) {
    weights_values[[ticker]] <- 0
   }
  }

  for (i in seq_along(tickers)) {
   insertUI(
    selector = "#dynamic_weights",
    ui = tags$div(
     fluidRow(
      # column(12, numericInput(weights[i], new_labels[i], value = weights_values[[tickers[i]]], min = 0, max = 100, step = 0.001, width = '400px'))
       column(12, numericInput(weights[i], new_labels[i], value = weights_values[[tickers[i]]], min = 0, max = 100, step = 0.001, width = '400px'))
     ),
     class = "dynamic-weight-input"
    ),
    multiple = TRUE
   )
  }
```

```
})

observe({
 lapply(tickers_to_analyze(), function(ticker) {
  observeEvent(input[[paste0("w_", str_to_title(ticker))]], {
   weights_values[[ticker]] <- input[[paste0("w_", str_to_title(ticker))]]
  }, ignoreNULL = FALSE)
 })
})

# Reset UI when changing the input method
observeEvent(input$pf_stock_elm, {
 updateTextInput(session, "db_esg_sbl", value = "")
 removeUI(selector = ".dynamic-weight-input", multiple = TRUE)
})

# weights_values <- reactiveValues()
#
# observeEvent(input$db_esg_sbl, {
#  removeUI(selector = ".dynamic-weight-input", multiple = TRUE)
#
#  if (is.null(input$db_esg_sbl) || length(input$db_esg_sbl) == 0 || (length(input$db_esg_sbl) == 1 &&
input$db_esg_sbl == "")) {
#   return(NULL)
#  }
#
#  new_labels <- input$db_esg_sbl
#  weights <- paste0("w_", str_to_title(input$db_esg_sbl))
#
```

```
#   for (ticker in input$db_esg_sbl) {
#     if (is.null(weights_values[[ticker]])) {
#       weights_values[[ticker]] <- 0
#     }
#   }
#
#   for (i in seq_along(input$db_esg_sbl)) {
#     insertUI(
#       selector = "#dynamic_weights",
#       ui = tags$div(
#         fluidRow(
#                                       column(12, numericInput(weights[i], new_labels[i], value =
weights_values[[input$db_esg_sbl[i]]], min = 0, max = 100, step = 0.001, width = '400px'))
#         ),
#         class = "dynamic-weight-input"
#       ),
#       multiple = TRUE
#     )
#   }
#
#   if (is.null(input$db_esg_sbl) || length(input$db_esg_sbl) == 0 || (length(input$db_esg_sbl) == 1 &&
input$db_esg_sbl == "")) {
#     removeUI(selector = ".dynamic-weight-input", multiple = TRUE)
#   }
#
# })


# weights_values <- reactiveValues()
#
```

```
# observeEvent(input$db_esg_sbl, {
#   removeUI(selector = ".dynamic-weight-input", multiple = TRUE)
#
#   the_ticks = new_tickers_to_analyze()
#
#   if (is.null(the_ticks) || length(the_ticks) == 0 || (length(the_ticks) == 1 && the_ticks == "")) {
#     return(NULL)
#   }
#
#   new_labels <- new_tickers_to_analyze
#   weights <- paste0("w_", str_to_title(the_ticks))
#
#   for (ticker in the_ticks) {
#     if (is.null(weights_values[[ticker]])) {
#       weights_values[[ticker]] <- 0
#     }
#   }
#
#   for (i in seq_along(the_ticks)) {
#     insertUI(
#       selector = "#dynamic_weights",
#       ui = tags$div(
#         fluidRow(
#           column(12, numericInput(weights[i], new_labels[i], value = weights_values[[the_ticks[i]]],
min = 0, max = 100, step = 0.001, width = '400px'))
#         ),
#         class = "dynamic-weight-input"
#       ),
#       multiple = TRUE
```

```
#    )
#  }
# })

#  observe({
#  lapply(input$db_esg_sbl, function(ticker) {
#    observeEvent(input[[paste0("w_", str_to_title(ticker))]], {
#      weights_values[[ticker]] <- input[[paste0("w_", str_to_title(ticker))]]
#    }, ignoreNULL = FALSE)
#  })
# })

# weights_values <- reactiveValues()
#
# observeEvent(input$db_esg_sbl, {
# # observeEvent(input$db_esg_sbl, {
#  # Effacer les champs de poids précédents
#  removeUI(selector = ".dynamic-weight-input", multiple = TRUE)
#
#  if (length(tickers_to_analyze()) == 1 && tickers_to_analyze() == "") {
#  # if (length(input$db_esg_sbl) == 1 && input$db_esg_sbl == "") {
#    return(NULL)
#  }
#
#  # new_labels <- paste0(str_to_title(input$db_esg_sbl), " weight")
#  # new_labels <- paste0(str_to_title(input$db_esg_sbl))
#
#  # new_labels <- input$db_esg_sbl
#  new_labels <- tickers_to_analyze()
```

```
#
#   # weights <- paste0("w_", str_to_title(input$db_esg_sbl))
#
#   weights <- paste0("w_", str_to_title(tickers_to_analyze()))
#
#   # Mettre à jour les valeurs des poids
#   # for (ticker in input$db_esg_sbl) {
#
#   for (ticker in tickers_to_analyze()) {
#     if (is.null(weights_values[[ticker]])) {
#       weights_values[[ticker]] <- 0
#     }
#   }
#
#   # n <- length(input$db_esg_sbl)
#   n <- length(tickers_to_analyze())
#
#   row_num <- 1
#   col_num <- 1
#   max_cols <- 12
#
#   # for (i in seq_along(input$db_esg_sbl)) {
#   for (i in seq_along(tickers_to_analyze())) {
#     insertUI(
#       selector = "#dynamic_weights",
#       ui = tags$div(
#         fluidRow(
#                                     # column(12, numericInput(weights[i], new_labels[i], value =
weights_values[[input$db_esg_sbl[i]]], min = 0, max = 100, step = 0.001, width = '400px'))
```

```
#                                   column(12, numericInput(weights[i], new_labels[i], value =
weights_values[[tickers_to_analyze()[i]]], min = 0, max = 100, step = 0.001, width = '400px'))
#       ),
#       class = "dynamic-weight-input"
#     ),
#     multiple = TRUE
#   )
#
#   # Vérifier et enlever cette partie
#   col_num <- col_num + 1
#   if (col_num > max_cols) {
#     col_num <- 1
#     row_num <- row_num + 1
#   }
#
#  }
# })
#
# observe({
#  # lapply(input$db_esg_sbl, function(ticker) {
#  lapply(tickers_to_analyze(), function(ticker) {
#   observe({
#     weights_values[[ticker]] <<- input[[paste0("w_", str_to_title(ticker))]]
#   })
#  })
# })

# tickers_to_analyze <- reactive({
#  if(length(input$db_esg_sbl) == 1 && input$db_esg_sbl == ""){
```

```
#    return(NULL)
#  } else {
#    # Ajouter ici le code sur le type de la méthode pour obtenir les données
#
#    tickers = input$db_esg_sbl
#
#    return(tickers)
#  }
# })


# Créer une performance data pour tickers_to_analyze()
the_performance <- reactive({
  if(!is.null(tickers_to_analyze())){
    tickers_to_analyze = tickers_to_analyze()
    performance_data <- analyze_portfolio_performance(tickers_to_analyze, input$gp_date[1], input$gp_date[2])

    return(performance_data)
  } else{
    return(NULL)
  }
})

performance_data <- reactive({
  if(!is.null(the_performance())){
    performance = the_performance()
    performance_data <- performance$result

    # print("performance data passe")
```

```
    # print(performance_data)
    return(performance_data)
  } else{
    return(NULL)
  }
})

best_performance_data <- reactive({
  if(!is.null(the_performance())){
    performance = the_performance()
    best_performance_data <- performance$best_table

    print("The head_best_porf")
    # print(head(best_performance_data))

    # print("performance data passe")
    # print(performance_data)
    return(best_performance_data)
  } else{
    return(NULL)
  }
})

performance_data__tbl <- reactive({
  if(!is.null(performance_data())){
    performance_tbl = the_performance()
    # Verifier s'il faut toujours table ou non
    # Surtout garder le nom des colonnes
    performance_tbl <- performance_tbl$table
```

```
   # print("performance data passe")
   # print(performance_data)
   return(performance_tbl)
  } else{
   return(NULL)
  }
})

performance_plot_data <- reactive({
  if(!is.null(performance_data())){
   performance_data = performance_data()

   result <- analyze_portfolio_performance_plot(performance_data,
                                num_simulations = 500,
                                # type_simulation = input$type_simulation,
                                # num_simulations = input$nb_simulation,
                                type_simulation = "uniform"
                                # ,
                                # size_type = input$size_type
   )

   print("performance data passe")
   # print(performance_data)
   return(result)
  } else{
   return(NULL)
  }
})
```

```
# output$performance_3d_plot <- renderPlotly({
#
#   # reactive({
#    if(!is.null(performance_plot_data())){
#     df = performance_plot_data()
#
#     df = df$fig_df
#
#     # Créer un graphique 3D avec plotly
#     fig <- plot_ly(df, x = ~Return,
#                    y = ~Volatility,
#                    z = ~ESG,
#                    color = ~Profit,
#                    colors = c('#BF382A', "darkgreen", '#0C4B8E'))
#
#     fig <- fig %>% add_markers(marker = list(size = input$size_type))
#     fig <- fig %>% layout(scene = list(xaxis = list(title = 'Return'),
#                                        yaxis = list(title = 'Volatility'),
#                                        zaxis = list(title = 'ESG score')),
#                           title = "Simulated Portfolio Performance")
#
#    } else{
#     return(NULL)
#
#    }
#   # })
#
```

```
# })

new_performance_plot_data <- reactive({
 if(!is.null(best_performance_data())){
  performance_data = best_performance_data()

  tickers <- tickers_to_analyze()
  weights = weights_tb()$wt
  weights = as.numeric(weights)
  print("--Les weights dans new performance--")
  print(length(weights))
  print(dim(weights))

  result <- update_analyze_pf_perf_plot(performance_data = performance_data,
                                        size_type = input$size_type,
                                        # type_return = "daily",
                                        user_weights = weights)



  print("performance data passe")
  # print(performance_data)
  return(result)
 } else{
  return(NULL)
 }
})

## Afficher le 3D plot avec la frontière efficiente
## Nouvelle version
```

```
output$performance_3d_plot <- renderPlotly({
 result <- new_perf_plot_data()

 # if(!is.null(new_perf_plot_data())){
 if (!is.null(result)) {
  final_plot <- result$efficient_frontier_3d
  # final_plot <- result$efficient_frontier_3d.b

  return(final_plot)
 }else{
  return(NULL)
 }

})

## Afficher le 3D plot avec la frontière efficiente
## Ancienne version
# output$performance_3d_plot <- renderPlotly({
#   if (!is.null(new_performance_plot_data())) {
#     # Debutons le test
#     # df1 = performance_plot_data()
#     df = new_performance_plot_data()
#
#     fig_dt <- df$fig_df  # Récupérer le plot 3D actuel
#     # Créer un graphique 3D avec plotly
#     # # Ajout des points représentant la frontière efficiente dans le plot 3D
#     # frontier_data <- new_perf_plot_data()$table_simul
#     # # Trier par risque croissant
#     # frontier_data <- frontier_data[order(frontier_data$Risk), ]
```

```
#   #
#   # print("-- Affichons la frontière--")
#   # print(head(frontier_data))
#   # frontier_data$ESG = df1$simulation$ESG
#   # # Ajout des lignes de la frontière efficiente
#   #
#   # # Utilisation de plotly pour créer un graphe 3D avec la frontière efficiente
#   # fig <- plot_ly() %>%
#   #  add_trace(
#   #    data = fig_dt,
#   #    type = 'scatter3d',   # Spécifiez explicitement le type de trace
#   #    mode = "markers",     # Mode markers pour les points
#   #    x = ~Return,
#   #    y = ~Volatility,
#   #    z = ~ESG,
#   #    color = ~Profit,
#   #    colors = c('#BF382A', "darkgreen", "#0C4B8E"),
#   #    marker = list(size = input$size_type)
#   #  ) %>%
#   #  add_trace(
#   #    data = frontier_data,
#   #    type = 'scatter3d',   # Spécifiez explicitement le type de trace
#   #    # mode = "lines",      # Mode lines pour la ligne de la frontière efficiente
#   #    mode = "markers",
#   #    x = ~Return,
#   #    y = ~Risk,
#   #    z = ~ESG,
#   #    name = "Frontier",
#   #    # line = list(color = 'darkyellow'),
```

```
#    #    marker = list(color = 'yellow'),
#    #    # marker = list(color = 'darkyellow'),
#    #    showlegend = FALSE
#    #  ) %>%
#    #  layout(
#    #    title = "3D Portfolio Performance with Efficient Frontier",
#    #    legend = list(title = list(text = 'ESG performance'))
#    #  )
#
#    ## Les test
#    # print("head fig_dt")
#
#    # print(dput(head(fig_dt, 25)))
#
#    # print("tail fig_dt")
#
#    # print(dput(tail(fig_dt, 25)))
#
#    fig <- customize_eff_front(fig_dt)
#
#
#    return(fig)
#  } else {
#    return(NULL)
#  }
# })
```

```
 output$performance_data_tbl <- renderDT({
  if(!is.null(performance_data())){
   performance_data = performance_data()

   performance_data$Return = round(performance_data$Return*100, 5)
   performance_data$Volatility = round(performance_data$Volatility, 5)

   names(performance_data)[1] = "Return (in %)"

   # performance_tb = DT::datatable(performance_data, rownames = TRUE,
   #                     # extensions = list('ColReorder' = NULL, 'RowReorder' = NULL,
   #                     #                   'Buttons' = NULL, 'Responsive' = NULL),
   #                     # extensions = 'Buttons',
   #                     options = list(
   #                      buttons = list('copy', 'print',
   #                                     list(extend = 'collection',
   #                                          buttons = c('csv', 'excel', 'pdf'),
   #                                          text = 'Download'), I('colvis')),
   #                      # dom = 'Bfrtip',
   #                      # buttons = c('copy', 'print', 'csv'),
   #                      columnDefs = list(list(targets = "_all", className = "dt-center",
   #                                             width = "35px"))
   #                     )
   # )

   performance_tb = DT::datatable(performance_data, escape = FALSE, filter = 'top', rownames =
TRUE,
                       # style = "bootstrap4",
                       extensions = list('ColReorder' = NULL, 'RowReorder' = NULL,
```

```
                'Buttons' = NULL, 'Responsive' = NULL),
              options = list(
               # scrollY = 280,
               # scrollX = 450,
               # scroller = TRUE,

               # fixer les colonnes :
               fixedColumns = list(leftColumns = 1),

               # autoHideNavigation = getOption("DT.autoHideNavigation", TRUE),

               ColReorder = TRUE,
               rowReorder = TRUE,
               searching = TRUE,  # Activer l'option de recherche
               ordering = TRUE,
               showPageSizeOptions = FALSE,
               paging = FALSE,  # Désactiver la pagination
               lengthMenu = list(c(10, 50, 100, -1), c('5', 'All')),
               #pageSizeOptions = c(5, 10, 15),
               buttons = list('copy',
                        'print',
                        list(extend = 'collection',
                           buttons = c('csv', 'excel', 'pdf'),
                           text = 'Download'), I('colvis')),

               columnDefs = list(list(targets = "_all", className = "dt-center",
                             width = "50px"))
              )
)
```

```
   return(performance_tb)
  } else{
   return(NULL)
  }
})

weights_tb <- reactive({
  if(!is.null(performance_plot_data())){
   # if (length(input$db_esg_sbl) > 0) {
   # Ancienne versionne remplacée par tickers_to_analyze()
   # tickers <- input$db_esg_sbl
   tickers <- tickers_to_analyze()
   weights <- sapply(tickers, function(ticker) weights_values[[ticker]])

   df_opt = performance_plot_data()

   # print("test sur df_opt$optimal_weights")
   # print((df_opt[, "optimal_weights"]))

   # print('str df_opt')
   # print(str(df_opt))

   w_opt = df_opt$optimal_weights

   # Sys.sleep(1)
   # Conversion explicite en numérique
   weights <- as.numeric(weights)
```

```
w_opt <- as.numeric(w_opt)

print("test sur le weight")
print(weights)
print(str(weights))

# weights = round(weights, 6)
#
# Sys.sleep(2)
#
# w_opt = round(w_opt*100, 6)

# Vérification que les valeurs sont numériques avant d'appliquer round()
if (!is.null(weights) && is.numeric(weights)) {
  weights = round(weights, 6)
}

if (!is.null(w_opt) && is.numeric(w_opt)) {
  w_opt = round(w_opt*100, 6)
}


if(sum(weights)>=0 && sum(weights) <=100) {
  weights = weights

}else{
  weights = (weights/sum(weights))*100

}
```

```
    weights = round(weights, 3)

    df <- data.frame(
      Info = c(tickers, "Portfolio"),
      Weight = c(weights, sum(weights)),
      `Optimal.weight` = c(w_opt, sum(w_opt))
    )

    weights_vector <- as.numeric(df[df$Info != "Portfolio", "Weight"])
    df[df$Info == "Portfolio", "Optimal.weight"] = round(as.numeric(df[df$Info == "Portfolio", "Optimal.weight"]), 2)

    df[df$Info == "Portfolio", "Weight"] = round(as.numeric(df[df$Info == "Portfolio", "Weight"]), 2)

    # len_ = length(df$Weight)
    # weights_ = df$Weight[1:(len_-1)]

    return(list(df= df,
                wt = weights_vector))

  } else {
    return(NULL)
  }

 })


 output$weights_table <- renderDT({
```

```
if(!is.null(performance_plot_data())){
 # if (length(input$db_esg_sbl) > 0) {
 # tickers <- input$db_esg_sbl
 # weights <- sapply(tickers, function(ticker) weights_values[[ticker]])

 # df_opt = performance_plot_data()
 #
 # w_opt = df_opt$optimal_weights
 #
 # weights = round(weights, 6)
 # w_opt = round(w_opt*100, 6)
 #
 # df <- data.frame(
 #   Info = c(tickers, "Portfolio"),
 #   Weight = c(weights, sum(weights)),
 #   `Optimal.weight` = c(w_opt, sum(w_opt))
 # )

 df = weights_tb()$df

 # df$

 names(df) = c("Info", "Weight (in %)", "Optimal weight (in %)")

 df$Info = as.factor(df$Info)

 # datatable(df, rownames = FALSE)
 datatable(df, escape = FALSE, filter = 'top', rownames = FALSE,
         # style = "bootstrap4",
```

```
extensions = list('ColReorder' = NULL, 'RowReorder' = NULL,
                  'Buttons' = NULL, 'Responsive' = NULL),
options = list(
 # scrollY = 280,
 # scrollX = 450,
 # scroller = TRUE,

 # fixer les colonnes :
 fixedColumns = list(leftColumns = 1),

 # autoHideNavigation = getOption("DT.autoHideNavigation", TRUE),

 ColReorder = TRUE,
 rowReorder = TRUE,
 searching = TRUE,  # Activer l'option de recherche
 ordering = TRUE,
 showPageSizeOptions = FALSE,
 paging = FALSE,  # Désactiver la pagination
 lengthMenu = list(c(10, 50, 100, -1), c('5', 'All')),
 #pageSizeOptions = c(5, 10, 15),
 buttons = list('copy',
                'print',
                list(extend = 'collection',
                     buttons = c('csv', 'excel', 'pdf'),
                     text = 'Download'), I('colvis')),
 # For colomns definition
 # columnDefs = clm_format

 columnDefs = list(list(targets = "_all", className = "dt-left",
```

```
                            width = "50px"))
          )
   )

 } else {
   return(NULL)
 }
})

# ouptimal weight
output$optimal_weight <- renderDT({

 # reactive({
 if(!is.null(performance_plot_data())){
   df = performance_plot_data()

   df = df$optimal_weights

   DT::datatable(df)

 } else{
   return(NULL)

 }
 # })

})

# variance_cov table
```

```
output$variance_cov <- renderDT({

 # reactive({
 if(!is.null(performance_data__tbl())){

  # performance_tbl = the_performance()
  # performance_data_table <- performance_tbl$table

  performance_data_table <- performance_data__tbl()

  # Pivotez la dataframe
  pivoted_df <- performance_data_table%>%
   tidyr::pivot_wider(id_cols = "Date",
                      names_from = "Ticker",
                      values_from = "Return")

  # var_cov = cov(pivoted_df[, -c(1)])

  var_cov = cov(na.omit(pivoted_df[, -c(1)]))

  var_cov = round(var_cov, 6)

  DT::datatable(var_cov, rownames = TRUE,
                options = list(
                 buttons = list('copy', 'print',
                                list(extend = 'collection',
                                     buttons = c('csv', 'excel', 'pdf'),
                                     text = 'Download'), I('colvis')),
```

```
            columnDefs = list(list(targets = "_all", className = "dt-center",
                                   width = "35px"))
          )
  )

 } else{
  return(NULL)

 }
 # })

})



# variance_cov table
output$variance_cov_heatmap <- renderHighchart({

 # reactive({
 if(!is.null(performance_data__tbl())){

  # performance_tbl = the_performance()
  # performance_data_table <- performance_tbl$table

  performance_data_table <- performance_data__tbl()

  # Pivotez la dataframe
  pivoted_df <- performance_data_table%>%
   tidyr::pivot_wider(id_cols = "Date",
                      names_from = "Ticker",
```

```
                values_from = "Return")

# var_cov = cov(pivoted_df[, -c(1)])

var_cov = cov(na.omit(pivoted_df[, -c(1)]))

var_cov = round(var_cov, 6)

# print(var_cov)

hc <- hchart(var_cov,
          hcaes(x = ., y = ., value = .),
          type = "heatmap",
          name = "Variance-Covariance Heatmap")%>%
  hc_colorAxis(stops = color_stops(100))
# Cette partie fonctionne
#    %>% # Customize colors if needed
#      hc_tooltip(formatter = JS("function() {
#   return 'Value : <b>' + Highcharts.numberFormat(this.point.value, 6) + '</b>';
# }"))



# Convert matrix to data frame for highcharter
df <- as.data.frame(as.table(var_cov))
colnames(df) <- c("x", "y", "value")

# # Create the heatmap with highcharter
# hchart(df, type = "heatmap", hcaes(x = x, y = y, value = value)) %>%
#   hc_colorAxis(stops = color_stops(10)) %>%
```

```
    #   hc_tooltip(pointFormat = 'Value : <b>{point.value:.6f}</b>') %>%

    #   hc_title(text = "Variance-Covariance Heatmap") %>%

    #   hc_xAxis(title = list(text = '')) %>%

    #   hc_yAxis(title = list(text = ''))


    # Ceci fonctionne mais verifier comment l'ajuster

    # hc <- hchart(df, hcaes(x = x, y = y, value = value), type = "heatmap", name = "Variance-Covariance Heatmap") %>%

    #   hc_colorAxis(stops = color_stops(10)) %>%

    #   hc_tooltip(pointFormat = 'Value: <b>{point.value:.6f}</b>') %>%

    #   hc_title(text = "Variance-Covariance Heatmap") %>%

    #   hc_xAxis(title = list(text = ''), categories = rownames(df)) %>%

    #   hc_yAxis(title = list(text = ''), categories = colnames(df))


    # Display the heatmap

    hc


  } else{

    return(NULL)


  }

  # })


})



# SNOW à étudier


output$performance_data_tbl_ <- renderDT({
```

```
if(!is.null(performance_data())){
 performance_data = performance_data__tbl()

 performance_data$Return = round(performance_data$Return*100, 5)
 performance_data$Close = round(performance_data$Close, 5)

 names(performance_data)[4] = 'Log return (in %)'

 performance_tb = DT::datatable(performance_data, rownames = FALSE,
                    # extensions = list('ColReorder' = NULL, 'RowReorder' = NULL,
                    #                   'Buttons' = NULL, 'Responsive' = NULL),
                    # extensions = 'Buttons',
                    options = list(
                     buttons = list('copy', 'print',
                                    list(extend = 'collection',
                                         buttons = c('csv', 'excel', 'pdf'),
                                         text = 'Download'), I('colvis')),
                     # dom = 'Bfrtip',
                     # buttons = c('copy', 'print', 'csv'),
                     columnDefs = list(list(targets = "_all", className = "dt-center",
                                            width = "35px"))
                    )
 )


 return(performance_tb)
} else{
 return(NULL)
}
```

```
})



new_perf_plot_data <- reactive({
 # if(!is.null(performance_plot_data())){
 if(!is.null(best_performance_data())){

  # print(performance_plot_data)

  # Ancienne version remplacée par tickers_to_analyze()
  # tickers <- input$db_esg_sbl

  tickers <- tickers_to_analyze()

  # weights <- sapply(tickers, function(ticker) weights_values[[ticker]])
  #
  # df_opt = performance_plot_data()
  #
  # w_opt = df_opt$optimal_weights
  #
  # weights = round(weights, 6)
  # w_opt = round(w_opt*100, 6)
  #
  # df <- data.frame(
  #   Info = c(tickers),
  #   Weight = c(weights),
  #   `Optimal.weight` = c(w_opt)
  # )
```

```
# weights <- sapply(tickers_to_analyze(), function(ticker) weights_values[[ticker]])
# if(!is.null(tickers_to_analyze())){
#   weights <- sapply(tickers, function(ticker) weights_values[[ticker]])

# weights = weights[1,]
# print(dim(weights))
# print(weights)

# weights = df$Weight

weights = weights_tb()$wt

weights = as.numeric(weights)
print(length(weights))
print(dim(weights))

# weights =

# print(weights)



# weights = c(weights)
# print(dim(weights))
# weights = as.numeric(weights)
# print(weights)
# print("tickers : ")
# print(tickers)
# print("Weights : ")
# print(weights)
```

```
# print(paste0("Risk free rate : ",input$gp_rfrate_new))
# print(paste0("Start date : ", input$gp_date[1]))
# print(paste0("End date : ", input$gp_date[2]))
# print(paste0("Nb of simulation : ", input$nb_simulation))
# print(paste0("Return type : ", input$gp_retfreq))
# print(paste0("Type simulation : ", input$type_simulation))



# Le décommenter après
options(warn = -1)

# result <- best_analyze_pf_perf(tickers_to_analyze = tickers,
#                       rf = input$gp_rfrate_new,
#                       start_date = input$gp_date[1],
#                       end_date = input$gp_date[2],
#                       num_simulations = 500,
#                       # num_simulations = input$nb_simulation,
#                       type_return = input$gp_retfreq,
#                       type_simulation = 'uniform',
#                       # type_simulation = input$type_simulation,
#                       user_weights = weights)

performance_data = best_performance_data()

# EN utilisant la nouvelle version plus rapide
result <- best_analyze_pf_perf(data = performance_data,
                        rf = input$gp_rfrate_new,
                        type_return = input$gp_retfreq,
                        user_weights = weights,
```

```
                    num_simulations = 500,
                    type_simulation = 'uniform')

  # Ramener les messages
  options(warn = 0)

  return(result)

 } else{
  return(NULL)
 }
})

output$perf_plot_min_var <- renderPlotly({
 result <- new_perf_plot_data()

 # print(result)

 # if(!is.null(new_perf_plot_data())){
 if (!is.null(result)) {
  final_plot <- result$min_var_port_w
  # result = new_perf_plot_data()
  # print("head of real data")
  # print(head(result$table))

  # print("tail of simulation data")
  # print(tail(result$table_simul))

  return(final_plot)
```

```
 }else{
  return(NULL)
 }

})

output$perf_plot_max_var <- renderPlotly({
 result <- new_perf_plot_data()

 # if(!is.null(new_perf_plot_data())){
 if (!is.null(result)) {
  final_plot <- result$max_var_port_w

  return(final_plot)
 }else{
  return(NULL)
 }

})

output$perf_plot_front_eff <- renderPlotly({
 result <- new_perf_plot_data()

 # print(result)

 # if(!is.null(new_perf_plot_data())){
 if (!is.null(result)) {
```

```
   final_plot = result$efficient_frontier_2d

   return(final_plot)
  }else{
   return(NULL)
  }

})

##############
############## PARTIE STOCK

output$unik_esg_sbl_ui <- renderUI({
  if (input$unik_stock_elm == "Select") {
    selectInput(
      "esg_sbl", strong("Choose stock"),
      c("", esg_etf_comparison$ticker),
      multiple = FALSE,
      selected = '',
      selectize = TRUE
    )
  } else if (input$unik_stock_elm == "Isin") {
    textInput(
      "esg_sbl", strong("Enter ISIN"),
      placeholder = "e.g., US0378331005"
    )
  } else if (input$unik_stock_elm == "Entrance") {
    textInput(
      "esg_sbl", strong("Enter ticker"),
```

```
    placeholder = "e.g., AAPL"
   )
  }
})


stock_to_analyze <- eventReactive(input$unik_run_button, {

 if (input$unik_stock_elm == "Select") {

  text = input$esg_sbl

  if (text == "") {
   showNotification("Please select a stock", type = "warning")
   # return(NULL)
   return("")
  }

  return(text)

 } else if (input$unik_stock_elm == "Isin") {

  user_input <- input$esg_sbl

  if (is.null(user_input) || user_input == "") {
   showNotification("Please enter an ISIN", type = "warning")
   # return(NULL)
   return("")
```

```
  }

  # Nettoyer l'entrée
  the_isin <- gsub("[[:space:],]", "", user_input)

  # Vérifier si plusieurs ISIN ont été saisis
  if (grepl(",", user_input)) {
   showNotification("Only one ISIN can be submitted!", type = "error")
   # return(NULL)
   return("")

  }

  # Valider le format ISIN
  if (!is_valid_isin(the_isin)) {
   showNotification("Invalid ISIN format. ISIN should be 12 characters: 2 letters followed by 10 alphanumeric characters",
            type = "error")
   # return(NULL)
   return("")

  }

  test <- yf_isin_to_ticker(the_isin)

  if (length(test) == 3) {
   return(test$tick)
  } else {
   showNotification("ISIN not found. Please verify. May be it does not exist on Yahoo Finance",
```

```
                type = "error")
   # return(NULL)
   return("")
  }
 } else {

  user_input <- input$esg_sbl

  if (is.null(user_input) || user_input == "") {
   showNotification("Please enter a ticker", type = "warning")
   # return(NULL)
   return("")
  }

  # Nettoyer l'entrée
  the_ticker <- gsub("[[:space:],]", "", user_input)

  # Vérifier si plusieurs tickers ont été saisis
  if (grepl(",", user_input)) {
   showNotification("Only one ticker can be submitted!", type = "error")
   # return(NULL)
   return("")
  }

  #If length is greater than 1 show notification : Only one ticker can be sumited!
  # Give the code
  # else then return

  return(the_ticker)
```

```
 }
})



# Testim
Data <- reactive({

 if(nchar(stock_to_analyze()) == 0) {

  return(NULL)
  # Data <- as.data.frame(items_select())



 } else {
  Data <- tq_get(toupper(stock_to_analyze()), get = "stock.prices",
           from = input$date[1],
           to = input$date[2]+1)
 }

 return(Data)
 # Data
})



dt_esg_prices <- reactive({
 if(!is.null(Data())){
```

```
  fullData = Data()
  names(fullData) = str_to_title(names(fullData))

  fullData = fullData[, c("Date", "Open", "High", "Low",
                          "Close", "Volume", "Adjusted")]

  fullData[, c("Open", "High", "Low",
               "Close", "Volume", "Adjusted")] = round(fullData[, c("Open", "High", "Low",
                                                                    "Close", "Volume", "Adjusted")], 2)

  # rownames(fullData) = NULL
  return(fullData)

 }else{
  return(NULL)
 }
})



output$esg_prices <- renderDT({
 if(is.null(dt_esg_prices())){
  return(NULL)

 } else{

  fullData <- dt_esg_prices()

  fullData <- DT::datatable(fullData, rownames = FALSE,
```

```
                    extensions = 'Buttons', options = list(
                     dom = 'Bfrtip',
                     buttons = c('copy', 'print', 'csv'),
                     columnDefs = list(list(targets = "_all", className = "dt-center",
                                 width = "35px"))
                    )
   )
   return(fullData)

 }

})

Cours <- reactive({

 Data()$adjusted
 # Data()$close

})

Cours_closing <- reactive({
 if(!is.null(dim(Data()))){
   Data()$close

 } else{
   return(NULL)
 }
```

```
})

the_colour <- reactive({

 input$colour1

})

tbl_colour <- reactive({

 input$t_colour

})

output$esg_cl_price_chart <- renderPlotly({
 if (is.null(Cours_closing())) {
  # Si Cours est NULL, retourne un plotly vide
  # return(plot_ly())
  return(NULL)

 } else {

  lg_names = yf.get_long_names(stock_to_analyze())

  lg_names = gsub(", Inc.", "", lg_names)

  # the_title = paste0(lg_names, " closing price evolution")
  the_title = paste0(lg_names, " price evolution")
```

```
    plot = plot_ly(y = ~Cours_closing(), x = ~Data()$date, name = "Stock Price", mode = 'lines', type = 'scatter', line = list(color = the_colour())) %>%
      layout(title = the_title,
             xaxis = list(title = "Time"),
             yaxis = list(title = "Price"),
             paper_bgcolor = 'rgba(0,0,0,0)',
             plot_bgcolor = 'rgba(0,0,0,0)')
    # %>%
    #   layout(title = the_title)

    # plot = plot_ly(y = ~Cours_closing(), x = ~Data()$date, name = "Stock Price", mode = 'lines', line = list(color = the_colour())) %>%
    #    layout(xaxis = list(title = "Time"), yaxis = list(title = "Price"), paper_bgcolor = 'rgba(0,0,0,0)', plot_bgcolor = 'rgba(0,0,0,0)')

    # plot = plot_ly(y = Cours_closing(), x = Data()$date, name = "Stock Price",
    #        xlab = "Time", ylab = "Price", mode = 'lines',
    #        col = the_colour(),
    #        lty = 1, lwd = 2) %>%
    #   layout(paper_bgcolor = 'rgba(0,0,0,0)', plot_bgcolor = 'rgba(0,0,0,0)')

    # Ancienne version qui fonctionne aussi
    # plot = plot_ly(y = Cours_closing(), x = Data()$date, name = "Stock Price",
    #             xlab = "Time", ylab = "Price", mode = 'lines',
    #             # col = "dark blue",
    #             col = the_colour(),
    #             lty = 1, lwd = 2)
    return(plot)
  }
```

```
})

output$graphCours <- renderPlotly({
 if (is.null(Cours())) {
  # Si Cours est NULL, retourne un plotly vide
  # return(plot_ly())
  return(NULL)

 } else {
  return(plot_ly(y = Cours(), x = Data()$date, name = "Stock Price",
                 xlab = "Time", ylab = "Price", mode = 'lines',
                 # col = "dark blue",
                 col = the_colour,
                 lty = 1, lwd = 2))
 }
})

ESG_histo <- reactive({

 if (nchar(stock_to_analyze()) == 0) {
  return(NULL)
 } else {

  the_hist = get_historic_esg(stock_to_analyze())

  the_hist = na.omit(the_hist)
  # print(the_hist)
  # names(the_hist)
  return(the_hist)
```

```
   # names(the_hist)  = c("Date", "Total", "E", "S", "G")
 }

})


# output$esg_recent_score <- renderTable({
#   if (is.null(ESG_histo())) {
#     return(NULL)
#
#   } else {
#     the_hist = tail(ESG_histo(), 10)
#     names(the_hist) = c("Date", "T", "E", "S", "G")
#     # the_hist$Date = as_date(the_hist$Date)
#
#     return(the_hist)
#
#     }
#
#   })

output$esg_recent_score_DT <- renderDT({
 if (is.null(ESG_histo())) {
   return(NULL)

 } else {
   the_hist = tail(ESG_histo(), 7)
   names(the_hist) = c("Date", "T", "E", "S", "G")
```

```
the_hist = the_hist %>%
  arrange(desc(Date))

the_hist = the_hist[, c("Date", "E", "S", "G", "T")]

the_hist$Date = format(the_hist$Date, format = "%d/%m/%y")

the_hist = datatable(the_hist, escape = FALSE, filter = 'none', rownames = FALSE,
                     # extensions = list('ColReorder' = NULL, 'RowReorder' = NULL,
                     #                   'Buttons' = NULL,
                     #                   'Responsive' = NULL
                     #                   ),
                     extensions = 'Responsive',
                     options = list(scrollY = 280, scrollX = 450, scroller = TRUE,
                                    fixedColumns = list(leftColumns = 1),
                                    ColReorder = TRUE,
                                    rowReorder = TRUE,
                                    searching = FALSE,  # Désactiver l'option de recherche
                                    ordering = TRUE,  # Conserver l'option de tri
                                    paging = FALSE,  # Désactiver la pagination
                                    info = FALSE,  # Désactiver le message d'information
                                    showPageSizeOptions = FALSE,
                                    lengthMenu = list(c(10, 50, 100, -1), c('5', 'All')),
                                    pageSizeOptions = c(5, 10, 15),
                                    buttons = list('copy', 'print',
                                                   list(extend = 'collection',
                                                        buttons = c('csv', 'excel', 'pdf'),
                                                        text = 'Download'), I('colvis')),
```

```
                    columnDefs = list(list(targets = "_all", className = "dt-center",
                                           width = "35px"))
                )
)

# the_hist = datatable(the_hist, escape = FALSE, filter = 'top', rownames = FALSE,
#                 # style = "bootstrap4",
#                 extensions = list('ColReorder' = NULL, 'RowReorder' = NULL,
#                                   'Buttons' = NULL, 'Responsive' = NULL),
#                 options = list(scrollY = 280, scrollX = 450, scroller = TRUE,
#
#                                # fixer les colonnes :
#                                fixedColumns = list(leftColumns = 1),
#
#                                # autoHideNavigation = getOption("DT.autoHideNavigation", TRUE),
#
#                                ColReorder = TRUE,
#                                rowReorder = TRUE,
#                                searching = TRUE,  # Activer l'option de recherche
#                                ordering = TRUE,
#                                showPageSizeOptions = FALSE,
#                                lengthMenu = list(c(10, 50, 100, -1), c('5', 'All')),
#                                pageSizeOptions = c(5, 10, 15),
#                                buttons = list('copy',
#                                               'print',
#                                               list(extend = 'collection',
#                                                    buttons = c('csv', 'excel', 'pdf'),
#                                                    text = 'Download'), I('colvis')),
#                                # For colomns definition
```

```
    #                       # columnDefs = clm_format
    #
    #                       columnDefs = list(list(targets = "_all", className = "dt-center",
    #                                   width = "50px"))
    #               )
    # )
    return(the_hist)
  }


 })



 output$esg_recent_score_DT_full <- renderDT({
  if (is.null(ESG_histo())) {
    return(NULL)

  } else {
    the_hist = ESG_histo()
    names(the_hist) = c("Date", "Total ESG Risk Score", "Environement Risk Score", "Social Risk Score", "Governance Risk Score")

    the_hist = the_hist %>%
      arrange(desc(Date))

    # the_hist$Date = format(the_hist$Date, format = "%d/%m/%y")

    # the_hist = datatable(the_hist, escape = FALSE, filter = 'none', rownames = FALSE,
    #               extensions = list('ColReorder' = NULL, 'RowReorder' = NULL,
    #                           'Buttons' = NULL, 'Responsive' = NULL),
```

```
#               options = list(scrollY = 280, scrollX = 450, scroller = TRUE,
#                              fixedColumns = list(leftColumns = 1),
#                              ColReorder = TRUE,
#                              rowReorder = TRUE,
#                              searching = FALSE,  # Désactiver l'option de recherche
#                              ordering = TRUE,  # Conserver l'option de tri
#                              paging = FALSE,  # Désactiver la pagination
#                              info = FALSE,  # Désactiver le message d'information
#                              showPageSizeOptions = FALSE,
#                              lengthMenu = list(c(10, 50, 100, -1), c('5', 'All')),
#                              pageSizeOptions = c(5, 10, 15),
#                              buttons = list('copy', 'print',
#                                             list(extend = 'collection',
#                                                  buttons = c('csv', 'excel', 'pdf'),
#                                                  text = 'Download'), I('colvis')),
#                              columnDefs = list(list(targets = "_all", className = "dt-center",
#                                                     width = "35px"))
#               )
# )

the_hist = datatable(the_hist, escape = FALSE, filter = 'top', rownames = FALSE,
              # style = "bootstrap4",
              extensions = list('ColReorder' = NULL, 'RowReorder' = NULL,
                                'Buttons' = NULL, 'Responsive' = NULL),
              options = list(scrollY = 280, scrollX = 450, scroller = TRUE,

                             # fixer les colonnes :
                             fixedColumns = list(leftColumns = 1),
```

```
                    # autoHideNavigation = getOption("DT.autoHideNavigation", TRUE),

                    ColReorder = TRUE,
                    rowReorder = TRUE,
                    searching = TRUE,  # Activer l'option de recherche
                    ordering = TRUE,
                    showPageSizeOptions = FALSE,
                    lengthMenu = list(c(10, 50, 100, -1), c('5', 'All')),
                    pageSizeOptions = c(5, 10, 15),
                    buttons = list('copy',
                                   'print',
                                   list(extend = 'collection',
                                        buttons = c('csv', 'excel', 'pdf'),
                                        text = 'Download'), I('colvis')),
                    # For colomns definition
                    # columnDefs = clm_format

                    columnDefs = list(list(targets = "_all", className = "dt-center",
                                           width = "50px"))
                )
  )
  return(the_hist)
 }

})


# saving esg data
esg_score_DT_full <- reactive({
```

```
 if (is.null(ESG_histo())) {
  return(NULL)

 } else {
  the_hist = ESG_histo()
  names(the_hist) = c("Date", "ESG", "E_Score", "S_Score", "G_Score")

  #       the_hist = the_hist %>%
  #         arrange(desc(Date))

  return(the_hist)
 }

})



# Partie Evolution des score
# Fonctionne très bien
output$esg_score_chart <- renderHighchart({
 if (is.null(ESG_histo())) {
  return(NULL)

 } else {
  chart_hist = ESG_histo()
  names(chart_hist) = c("Date", "Total", "E", "S", "G")

  chart_hist = chart_hist %>% hchart(
   name = "Total ESG Score",
```

```
    'area',
    # 'line',
    hcaes(x = Date, y = Total),
    # color = "steelblue"
    color =  the_colour()

  )

  return(chart_hist)



 }
})



# Partie rentabilité
# Choix de la rentabilité
Data_return <- reactive({
 if(!is.null(dim(dt_esg_prices()))){
  # dt_esg_prices()$close

  Data <- dt_esg_prices()

  Data <- Data %>%
   tq_transmute(select = Adjusted,
                mutate_fun = periodReturn,
                period=input$retfreq,
                type= input$log,
```

```
                col_rename = "Return")

    return(Data)


  } else{
    return(NULL)
  }


})

Data_return_after <- reactive({
  if(!is.null(Data_return())){
    Data <- Data_return()

    # print("--Return Stock--")
    # print(head(Data))
    # print(dput(head(Data, 15)))

    # proc_dt = process_data(Data)
    # return(proc_dt)


    return(process_data(Data))
  }
  return(NULL)
})
```

```
# Data_return_after <- reactive({
#   if(!is.null(dim(Data_return()))){
#     # dt_esg_prices()$close
#
#     Data <- Data_return()
#
#     process_data(Data)
#
#   } else{
#     return(NULL)
#   }
# })

###################################

# LA MOYENNE DES RENDEMENTS D'ACTION

Moyenne <- reactive({

  tq_performance(Data_return(),
                 Ra = Return,
                 performance_fun = mean)

})

# LA VOLATILITE DES RENDEMENTS

Volatilite <- reactive({
```

```
 tq_performance(Data_return(),
                Ra = Return,
                performance_fun = sd)

})

#* LA MEDIANE

Mediane <- reactive({


 MEDIAN(Data_return()$Return)

})

#* LE MIN DES RENDEMENTS

Min <- reactive({

 tq_performance(Data_return(),
                Ra= Return,
                performance_fun = min)

})

#* LE MAX DES RENDEMENTS

Max <- reactive({
```

```
 tq_performance(Data_return(),
                Ra = Return,
                performance_fun = max)

})

#* LE SKEWNESS DES RENDEMENTS

Skewn <- reactive({

 tq_performance(Data_return(),
                Ra= Return,
                performance_fun = skewness)

})

#* LE KURTOSIS DES RENDEMENTS

Kurt <- reactive({

 tq_performance(Data_return(),
                Ra= Return,
                performance_fun = kurtosis)

})
```

```
#* TEST DE LA NORMALITE

test1 <- reactive({

 dt = Data_return()

 if(nrow(dt)>3 & nrow (dt) <5000){
  rsl = shapiro.test(dt$Return)

  return(rsl)
 }else{
  return(NaN)
 }




})

test2 <- reactive({

 jarque.bera.test(Data_return()$Return)

})

#* VALUE AT RISK

vaR <- reactive({

 tq_performance(Data_return(),
```

```
        Ra= Return,
        performance_fun = VaR,
        p = 1 - input$alpha)

})


#* VALUE AT RISK CORNISH FISHER

varRcf <- reactive({

  zc <- qnorm(input$alpha,0,1,lower.tail=FALSE)

  # z <- zc+1/6*(zc^2-1)*Skewn()$skewness.1+1/24*(zc^3-3*zc)*Kurt()$kurtosis.1-1/36*(2*zc^3-5*zc)*Skewn()$skewness.1

  z <- zc+1/6*(zc^2-1)*Skewn()$skewness[1]+1/24*(zc^3-3*zc)*Kurt()$kurtosis[1]-1/36*(2*zc^3-5*zc)*Skewn()$skewness[1]

  Moyenne()$mean.1-z*Volatilite()$sd.1

})


#* MAX DRAWDOWN

Drawdown_max <- reactive({

  maxDrawdown(Data_return()$Return)
```

```
})

# Partie statistique
Stats <- reactive({

  # Test sur les données
  # print("--La moyenne--")
  # print(Moyenne()$mean.1)
  # print("-- Volatilité --")
  # print(Volatilite()$sd.1)
  # print("-- Skewness --")
  # print(Skewn())
  # print(Skewn()$skewness[1])
  # print("-- Kurtosis --")
  # print(Kurt())
  # print(Kurt()$kurtosis[1])
  # print("-- Min --")
  # print(Min()$min.1)
  # print("-- Max --")
  # print(Max()$max.1)
  # print("-- Mediane --")
  # print(Mediane())

  statistiques <- data.frame(Statistic = c("Mean","Volatility","Skewness","Kurtosis",
"Minimum","Maximum","Median"),
                             Value = round(
                               c(Moyenne()$mean.1,
                                 Volatilite()$sd.1,
```

```
                    # Skewn()$skewness.1,
                    Skewn()$skewness[1],
                    # Kurt()$kurtosis.1,
                    Kurt()$kurtosis[1],
                    Min()$min.1,
                    Max()$max.1,
                    Mediane()), 4)
 )

})

output$statistics <- renderDT({
 # output$statistics <- DT::renderDataTable({

 if(!is.null(Data_return())){
  # Stats()

  clm_format = list(list(className = "dt-center", targets = c(1)
                  # ,
                  # width = "50px"
  ),
  list(className = "dt-left", targets = 0)
  # ,
  # width = "50px"
  )

  stats <- DT::datatable(Stats(), rownames = FALSE,
                  extensions = 'Responsive', options = list(
                   # dom = 'Bfrtip',
```

```
                    # buttons = c(),
                    # buttons = c('copy',
                    #           # 'print',
                    #           'csv'),
                    searching = FALSE,
                    ordering = TRUE,  # Conserver l'option de tri
                    paging = FALSE,  # Désactiver la pagination
                    info = FALSE,  # Désactiver le message d'information
                    showPageSizeOptions = FALSE,
                    columnDefs = clm_format
                    # columnDefs = list(list(targets = "_all", className = "dt-center",
                    #                   width = "50px"))
                  )
  )

  return(stats)


 } else{
  return(NULL)
 }



})

# input$alpha

# DateVector <- reactive({
#
```

```
#   # as.Date(Rend1()$date)
#   Date_Vector <- as.Date(intersect(Data_return()$Date,
#                          RF_Data_return()$Date))
#
# })

 DateVector <- reactive({
  # Récupération des dates communes entre Data_return() et le benchmark (BNC_return ou RF_Data_return)
  benchmark_dates <- if (!is.null(BNC_return()) && nrow(BNC_return()) > 1) {
   BNC_return()$Date
  } else {
   RF_Data_return()$Date
  }

  as.Date(intersect(Data_return()$Date, benchmark_dates))
 })

 #* LES RENDEMENTS POUR LES DATES COMMUNES ACTIONS

 Rendem_bis <- reactive({

  Rendement_bis <- vector(mode = "double",length = length(DateVector()))
  names(Rendement_bis) <- "return_bis"


  n1 <- 0

  for (i in 1:length(Data_return()$Date))
```

```
 {

   if (Data_return()$Date[[i]]%in%DateVector()==TRUE){

     n1 <- n1+1

     Rendement_bis[[n1]] <-  Data_return()$Return[[i]] }

 }


  as.ts(Rendement_bis)

})

Rendem1_bis <- reactive({

  Rendem_bis_1 <-  DT::datatable(Rendem_bis(),
                         extensions = 'Buttons', options = list(
                           dom = 'Bfrtip',
                           buttons = c('copy', 'print', 'csv')))

})

# LES RENDEMENTS POUR LES DATES COMMUNES BENCHMARK
# Rendrf_bis <- reactive({
#
#
#   Rendementrf_bis <- vector(mode = "double",length = length(DateVector()))
```

```
#
#   n2 <- 0
#
#   for (j in 1:length(RF_Data_return()$Date))
#   {
#
#     if (RF_Data_return()$Date[[j]]%in%DateVector()==TRUE){
#
#       n2 <- n2+1
#
#       Rendementrf_bis[[n2]] <-  RF_Data_return()$Return[[j]] }
#
#   }
#
#   # Rendementrf_bis
#   as.ts(Rendementrf_bis)
#
# })



# LES RENDEMENTS POUR LES DATES COMMUNES BENCHMARK
Rendrf_bis <- reactive({
  benchmark <- if (!is.null(BNC_return()) && nrow(BNC_return()) > 1) {
    BNC_return()
  } else {
    RF_Data_return()
  }

  Rendementrf_bis <- numeric(length(DateVector()))
```

```
 n2 <- 0

 for (j in seq_along(benchmark$Date)) {
  if (benchmark$Date[[j]] %in% DateVector()) {
   n2 <- n2 + 1
   Rendementrf_bis[[n2]] <- benchmark$Return[[j]]
  }
 }

 as.ts(Rendementrf_bis)
})


## CALCUL DES RENDEMENTS GROUPE POUR LES ACTIONS
Rendem_gr <- reactive({

 Rendementgr_bis <- vector(mode = "double",length = length(Data_return()$Date))

 n3 <- 0

 Rendementgr_bis[[1]] <- 100


 for (i in 2:length(Data_return()$Date))
 {

  if (input$log == "arithmetic"){

   n3 <- n3+1
```

```
    Rendementgr_bis[[n3+1]] <-  Rendementgr_bis[[n3]] * (1 + Data_return()$Return[[i-1]]) }



  else {

    n3 <- n3+1

    Rendementgr_bis[[n3+1]] <-  Rendementgr_bis[[n3]] * exp(Data_return()$Return[[i-1]])

  }

 }


 as.ts(Rendementgr_bis)

})

# Version update
# Rendem_gr <- reactive({
#   Rendementgr_bis <- numeric(length(Data_return()$Date))
#   Rendementgr_bis[[1]] <- 100
#   n3 <- 0
#
#   for (i in 2:length(Data_return()$Date)) {
#    n3 <- n3 + 1
#    Rendementgr_bis[[n3 + 1]] <- if (input$log == "arithmetic") {
```

```
#     Rendementgr_bis[[n3]] * (1 + Data_return()$Return[[i - 1]])
#    } else {
#     Rendementgr_bis[[n3]] * exp(Data_return()$Return[[i - 1]])
#    }
#  }
#
#  as.ts(Rendementgr_bis)
# })



## CALCUL DES RENDEMENTS GROUPE POUR LE BENCHMARK
# Old version
# Rendrf_gr_bis <- reactive({
#
#
#  Rendementrfgr_bis <- vector(mode = "double",length = length(RF_Data_return()$Date))
#
#  n4 <- 0
#
#  Rendementrfgr_bis[[1]] <- 100
#
#  for (j in 2:length(RF_Data_return()$Date))
#  {
#
#    if (input$log =="arithmetic"){
#
#     n4 <- n4+1
#
#     Rendementrfgr_bis[[n4+1]] <-  Rendementrfgr_bis[[n4]] * (1 + RF_Data_return()$Return[[j-1]]) }
```

```
#
#
#
#    else {
#
#      n4 <- n4+1
#
#      Rendementrfgr_bis[[n4+1]] <-  Rendementrfgr_bis[[n4]] * exp(RF_Data_return()$Return[[j-1]])
#
#    }
#  }
#
#   # Rendementrf_bis
#   as.ts(Rendementrfgr_bis)
#
# })

Rendrf_gr_bis <- reactive({
 benchmark <- if (!is.null(BNC_return()) && nrow(BNC_return()) > 1) {
  BNC_return()
 } else {
  RF_Data_return()
 }

 Rendementrfgr_bis <- numeric(length(benchmark$Date))
 # Rendementrfgr_bis <- vector(mode = "double",length = length(benchmark$Date))
 Rendementrfgr_bis[[1]] <- 100
 n4 <- 0
```

```
 for (j in 2:length(benchmark$Date)) {
  n4 <- n4 + 1
  Rendementrfgr_bis[[n4 + 1]] <- if (input$log == "arithmetic") {
   Rendementrfgr_bis[[n4]] * (1 + benchmark$Return[[j - 1]])
  } else {
   Rendementrfgr_bis[[n4]] * exp(benchmark$Return[[j - 1]])
  }
 }

 as.ts(Rendementrfgr_bis)
})

############################
## TAUX SANS RISQUE

# addTooltip(session, id = "rfrate", title = "Enter the risk-free interest rate",
#         placement = "bottom", trigger = "hover")

# FRA <- reactive({
#
#   FRA <- input$rfrate
#
# })

## TAUX SANS RISQUE AJUSTEMENTS
# Nouvelle methode
FR <- reactive({

 if (input$benchmark) {
```

```
 if(!is.null(stock_and_rf_new2())){
  data = stock_and_rf_new2()

  names(data) = c("Date", "Ticker_return", "Index_return")

  dt = na.omit(data$Index_return)

  the_mean = mean(dt)
  # print(the_mean)

  the_mean

 }else{
  return(NULL)
 }

} else {
 the_rf = input$rfrate_new

 # if (input$retfreq == "daily"){
 #
 #   FR <- FRA()/365
 # }
 #
 # else if (input$retfreq == "weekly") {
 #
 #   FR <- FRA()/52
 #
 # }
```

```
  #
  # else if (input$retfreq == "monthly") {
  #
  #   FR <- FRA()/12
  #
  # }
  #
  # else if (input$retfreq == "yearly") {
  #
  #   FR <- FRA()
  #
  # }

  the_rf

 }



})

## Ancienne méthode
# FR <- reactive({
#
#   if (input$retfreq == "daily"){
#
#     FR <- FRA()/365
#   }
#
#   else if (input$retfreq == "weekly") {
```

```
#
#    FR <- FRA()/52
#
#  }
#
#  else if (input$retfreq == "monthly") {
#
#    FR <- FRA()/12
#
#  }
#
#  else if (input$retfreq == "yearly") {
#
#    FR <- FRA()
#
#  }
#
# })


# VECTEUR DES RENDEMENTS INFERIEURS AU RENDEMENT MOYEN


Rend_inf <- reactive({


  n <- length((which(Data_return()$Return < Moyenne()$mean.1))) # nombre de rendement < moyenne


  nn <- 0


  Rend_inf <- vector(mode = "double",length = n)
```

```
 for (j in 1:length(Data_return()$Return))
 {
   if (Data_return()$Return[[j]] < Moyenne()$mean.1) {

     nn <- nn+1

     Rend_inf[[nn]] <- Data_return()$Return[[j]]
   }
 }

 Rend_inf

})

## SEMIVARIANCE

semivar <- reactive({

 var(Rend_inf())

})



## MPI

MPI2 <- reactive({

 sqrt(semivar())
```

```
})

#
*****************************************************************************************
*************************
# * INDICATEURS                                                    *
#
*****************************************************************************************
*************************

Sharpe_ratio <- reactive({

  sp = tq_performance(Data_return(),
                      Ra = Return,
                      performance_fun = SharpeRatio,
                      Rf = FR())

  # print(sp)

  sp




})


Adjusted_Sharpe_Ratio <- reactive({

  tq_performance(Data_return(),
```

```
            Ra = Return,
            performance_fun = AdjustedSharpeRatio,
            Rf = FR())

})


MSharpe <- reactive({

 # Old version is working
 # but will be deprecate soon
 # ts(Data_return()$Return) %>%
 #   SharpeRatio.modified(FR())

 ts(Data_return()$Return) %>%
   SharpeRatio(FR())

})

## Espected Shortfall

ES <- reactive({

 tq_performance(Data_return(),
            Ra = Return,
            performance_fun = ES)
})

# M SQARED
```

```
 M_carre <- reactive({
  # print("Rendrf_bis()")
  # print(Rendrf_bis())
  MSquared(Ra = Rendem_bis(),
         Rb = Rendrf_bis(),
         FR())

 })


 # Tracking error
 TE <- reactive({

   TrackingError(Ra = Rendem_bis(),
             Rb = Rendrf_bis(),
             FR())

   # tq_performance(Rend_both(),Ra = Return_bis,Rb = Returnsrf_bis,performance_fun =
TrackingError)

 })


 IR <- reactive({

   the_inf = InformationRatio(Ra = Rendem_bis(),
                      Rb = Rendrf_bis(),
                      FR())
```

```
  # print(the_inf)

  return(the_inf)

  #         tq_performance(Rend_both(),Ra=Rendem_bis(),Rb=Rendrf_bis(),performance_fun         =
InformationRatio)

 })

 ## OMEGA

 omega_ratio <- reactive({

  tq_performance(Data_return(),
                 Ra = Return,
                 performance_fun = Omega,
                 Rf = FR(),
                 L = input$minaccret)

 })

 ## SHARPE OMEGA

 somega_ratio <- reactive({

  tq_performance(Data_return(),
```

```
            Ra = Return,
            performance_fun = OmegaSharpeRatio,
            MAR=input$minaccret)

})

## SORTINO RATIO

sortino <- reactive({

 tq_performance(Data_return(),
            Ra = Return,
            performance_fun = SortinoRatio,
            MAR = input$minaccret)

})

## CALMAR RATIO

calmar <- reactive({

 tq_performance(Data_return(),
            Ra = Return,
            performance_fun = CalmarRatio)

})

## TREYNOR RATIO
```

```
treynor <- reactive({
 the_treynor = TreynorRatio(Ra = Rendem_bis(),
                    Rb = Rendrf_bis(),
                    FR())

 print('le ratio treynor')
 print(the_treynor)

 if(length(Rendem_bis())>3 & length(Rendem_bis()) <5000){

  return(the_treynor)
 }else{
  return(NaN)
 }

})




## Jensen's alpha
Jenalpha <- reactive({
 CAPM.alpha(Ra = Rendem_bis(),
        Rb = Rendrf_bis(),
        FR())

})
```

```
## CAPM Beta
Beta <- reactive({

  CAPM.beta(Ra = Rendem_bis(),
        Rb = Rendrf_bis(),
        FR())
  #   tq_performance(Rend_both(),Ra   =   Return_bis,Rb   =   Returnsrf_bis,performance_fun   =
CAPM.beta,Rf=FR())

})

## bear beta
beta_bear <- reactive({
  CAPM.beta.bear(Ra = Rendem_bis(),
            Rb = Rendrf_bis(),
            FR())

})

## bull beta
beta_bull <- reactive({

  CAPM.beta.bull(Ra = Rendem_bis(),
            Rb = Rendrf_bis(),
            FR())
```

```
})


## INDICATEURS DU RISQUE
output$risks <- DT::renderDT({
 # output$risks <-  DT::renderDataTable({

 if (nchar(stock_to_analyze())==0) {
  dt_ = data.frame(Message="Error : Please select a stock, these indicators could not be calculated")

  DT::datatable(dt_, options = list(paging = FALSE,
                    searching = FALSE,
                    showPageSizeOptions = FALSE,
                    info = FALSE,
                    lengthChange = FALSE),
          rownames = FALSE)

 } else {
  print("-- vaR --")
  # print(vaR())
  # print(vaR()$VaR)
  print("-- varRcf --")
  # print(varRcf())
  # revoir aussi le calcul de Drawdown_max
  print("-- Drawdown_max --")
  # print(Drawdown_max())
  print("-- semivar --")
```

```
    # print(semivar())
    print("-- MPI2 --")
    # print(MPI2())
    # print("-- --")
    # print()

    dt_ = data.frame(Indicator = c("Value at Risk", "Value at Risk Cornish - Fisher","Expected Shortfall",
                                   "Max Drawdown", "Semi-variance", "Semi-deviation" ),
                     Value = c(round(c(vaR()$VaR,
                                       varRcf(),
                                       ES()$ES,
                                       Drawdown_max())
                                     ,4),
                               round(semivar(),4),
                               round(MPI2(),4)))

    render_dt_table(dt_)

  }

 })

 ## INDICATEURS DE PERFORMANCE

 output$performances_stock <- DT::renderDT({
  # output$performances_stock <- DT::renderDataTable({
```

```
  if (nchar(stock_to_analyze())==0) {
   dt_ = data.frame(Message="Error : Please select a stock, these indicators could not be calculated")
   DT::datatable(dt_, options = list(paging = FALSE,
                                     searching = FALSE,
                                     lengthChange = FALSE,
                                     showPageSizeOptions = FALSE,
                                     info = FALSE
   ),
   rownames = FALSE)

  } else {

   # if (input$benchmark){
   #   if (input$benchmark
   # }
   if (input$benchmark && nchar(input$Rf_rate)== 0) {

    dt_ = data.frame(Message="Error : Indicators could not be calculated! If you decide to use Benchmark returns as the risk-free rate, you have to select a benchmark first...")
    DT::datatable(dt_, options = list(paging = FALSE,
                                      searching = FALSE,
                                      lengthChange = FALSE,
                                      showPageSizeOptions = FALSE,
                                      info = FALSE
    ),
    rownames = FALSE)

   } else{
```

```
    dt_ = data.frame(Indicator = c("Sharpe Ratio", "Adjusted Sharpe Ratio", "Modified Sharpe Ratio",

                                   "Omega Ratio", "Omega Sharpe Ratio", "Sortino Ratio",

                                   "Calmar Ratio"),

                     Value = round(c(Sharpe_ratio()[[1]],
                                     Adjusted_Sharpe_Ratio()[[1]],
                                     MSharpe()[[1]],
                                     omega_ratio()[[1]],
                                     somega_ratio()$OmegaSharpeRatio.1,
                                     sortino()[[1]],calmar()$CalmarRatio),
                                   4))

    render_dt_table(dt_)

   }

  }

})



 output$performances_rf <- DT::renderDT({
  # output$performances_rf <- DT::renderDataTable({
  if (nchar(input$Rf_rate)==0) {
   dt_ = data.frame(Message="Error : Benchmark indicators could not be calculated! Please, select a benchmark.")
```

```
 DT::datatable(dt_, options = list(paging = FALSE,
                                   searching = FALSE,
                                   showPageSizeOptions = FALSE,
                                   info = FALSE,
                                   lengthChange = FALSE),
                   rownames = FALSE)

} else {

 Dt = data.frame(Indicator = c( "Mesure-Modigliani Modigliani",
                                "Tracking Error",
                                "Information Ratio",
                                "Treynor Ratio",
                                "Alpha", "Beta",
                                "Bear Beta",
                                "Bull Beta"),

                 Result=round(c(M_carre(),
                                TE(),
                                IR(),
                                treynor(),
                                Jenalpha(),
                                Beta(),
                                beta_bear(),
                                beta_bull()),
                              4))

 render_dt_table(Dt)
```

```
 }
})



# Nouvelle version
observe({
 if (!is.null(Data_return())) {
  testResultSW <- test1()
  testResultJB <- test2()

  output$testJBComment <- renderUI({
   # output$testSWComment <- renderText({
   displayNormalityComment(testResultSW, input$alpha, stock_to_analyze())
  })

  output$testSWComment <- renderUI({
   # output$testJBComment <- renderText({
   displayNormalityComment(testResultJB, input$alpha, stock_to_analyze())
  })

  output$pvalue1 <- renderText({
   # paste0("P-value : ",
   #       # "P-value du test de Shapiro-Wilk : ",
   #       "<b>", format(testResultSW$p.value, scientific = TRUE), "</b>")

   paste0("P-value : ",
         format(testResultSW$p.value, scientific = TRUE))
  })
```

```
    output$pvalue2 <- renderText({
      # paste0("P-value : ",
      #       # "P-value du test de Jarque-Bera : ",
      #       "<b>",format(testResultJB$p.value, scientific = TRUE), "</b>")

      paste0("P-value : ",
             format(testResultJB$p.value, scientific = TRUE))
    })
  } else {
    output$testSWComment <- renderText({ "" })
    output$testJBComment <- renderText({ "" })
    output$pvalue1 <- renderText({ "Sélectionnez une action pour effectuer le test" })
    output$pvalue2 <- renderText({ "Sélectionnez une action pour effectuer le test" })
  }
})


  # Ancienne version
  # output$testSW <- renderPrint({
  #
  #   if(!is.null(Data_return())){
  #
  #     if (test1()$p.value < input$alpha) {
  #
  #       cat("Les résultats du test de Shapiro-Wilk montrent que la p-value est inferieure au seuil de risque alpha. On rejette donc l'hypothèse nulle de normalité de la distribution. On peut en conclure que les rendements de l'action sélectionnée ne suivent pas une loi Normale.")
  #
```

```
#    } else {
#
#      cat("Les résultats du test de Shapiro-Wilk montrent que la p-value est plus importante que le seuil de risque alpha. On accepte donc l'hypothèse nulle de normalité de la distribution. On peut en conclure que les rendements de l'action sélectionnée suivent une loi Normale.")
#
#    }
#
#  }else{
#    cat("")
#    # cat("Select a stock to enable the test")
#  }
#
#
#
# })
#
# output$testJB <- renderPrint({
#
#  if(!is.null(Data_return())){
#
#    if (test2()$p.value < input$alpha) {
#
#       cat("Les résultats du test de Jarque-Bera montrent que la p-value est inferieure au seuil de risque alpha. On rejette donc l'hypothèse nulle de normalité de la distribution. On peut en conclure que les rendements de l'action sélectionnée ne suivent pas une loi Normale.")
#
#    } else {
#
```

```
#       cat("Les résultats du test de Jarque-Bera montrent que la p-value est plus importante que le seuil de risque alpha. On accepte donc l'hypothèse nulle de normalité de la distribution. On peut en conclure que les rendements de l'action sélectionnée suivent une loi Normale.")
#    }
#
#
# }else{
#   # return(NULL)
#   cat("")
#   # cat("Select a stock to enable the test")
# }
# })
#
# output$pvalue1 <- renderPrint( {
#
#   if(!is.null(Data_return())){
#     cat(paste0("P-value of this test is : ", format(test1()$p.value,
#                                         scientific = TRUE)))
#     # print(format(test1()$p.value,scientific = TRUE))
#
#   }else{
#     # return(NULL)
#     # cat("")
#     cat("Select a stock to perform the test")
#   }
#
# })
#
# output$pvalue2 <- renderPrint ({
```

```
#   if(!is.null(Data_return())){
#
#     cat(paste0("P-value of this test is : ", format(test2()$p.value,scientific = TRUE)))
#     # print(format(test2()$p.value,scientific = TRUE))
#
#   }else{
#     # cat("")
#     cat("Select a stock to perform the test")
# }
# })



##########################################"

#* RENDEMENTS ACTIONS
output$returns <- renderDT({
  if(!is.null(Data_return())){
    dt_return = Data_return()

    rownames(dt_return) = NULL

    # dt_return$Return = round(dt_return$Return, 4)*100
    dt_return[, 'Return'] = round(dt_return[, 'Return'] * 100, 4)

    names(dt_return) = c("Date", "Return (in %)")

    # dt_return = dt_return %>%
    #   formatPercentage(c("Return"),2)
```

```
  dt_return <- DT::datatable(dt_return, rownames = FALSE,
                    extensions = 'Responsive', options = list(
                     # dom = 'Bfrtip',
                     # buttons = c(),
                     # buttons = c('copy',
                     #          # 'print',
                     #          'csv'),
                     info = FALSE,
                     columnDefs = list(list(targets = "_all", className = "dt-center",
                                  width = "35px")),
                     # Formater l'affichage des nombres pour garder 4 décimales
                     formatRound = list(columns = 'Return', digits = 4)
                    )
  )

  dt_return

 } else{
  return(NULL)

 }



})

rf_id <- reactive({
 if(input$Rf_rate != ""){
  new_id = switch(input$Rf_rate,
            # 1 year benchmark
```

```
            "OAT 1y" = "FR1YT=RR",
            'T-Note 1y' = 'US1YT=X',
            'Bund 1y' = 'DE1YT=RR',
            # 5 year benchmark
            "OAT 5y" = "FR5YT=RR",
            'T-Note 5y' = 'US5YT=X',
            'Bund 5y' = 'DE5YT=RR',
            # 10 year benchmark
            "OAT 10y" = "FR10YT=RR",
            'Bund 10y' = 'DE10YT=RR',
            'T-Note 10y' = 'US10YT=X'
   )

   rsl = INV_get_id(new_id)

   return(rsl)

  }else{
   return(NULL)
  }
})



rf_info <- reactive({
  if(input$Rf_rate != ""){

   the_info = switch(input$Rf_rate,
                # 1 year benchmark
                "OAT 1y" = "FR1YT=RR",
```

```
            'T-Note 1y' = 'US1YT=X',
            'Bund 1y' = 'DE1YT=RR',
            # 5 year benchmark
            "OAT 5y" = "FR5YT=RR",
            'T-Note 5y' = 'US5YT=X',
            'Bund 5y' = 'DE5YT=RR',
            # 10 year benchmark
            "OAT 10y" = "FR10YT=RR",
            'Bund 10y' = 'DE10YT=RR',
            'T-Note 10y' = 'US10YT=X'
  )

  the_info



 }else{
  return(NULL)
 }
})



################################################################################
################################################################################
############################### PARTIE DEBUGGAGE ###############################
################################################################################
################################################################################

# show selected dates
```

```
output$selected_RF_id <- renderPrint({
 # print(rf_id())
 print(input$date[1])
 print(input$date[2])
 print(input$Rf_rate)
})

# show selected dates
output$print_1 <- renderPrint({
 dt_return = Data_return()
 print("head dt_return")
 print(head(dt_return))
 dt_esg = fill_ESG_dt_new()
 print("head dt ESG")
 print(head(dt_esg))
 print("tail dt ESG")
 print(tail(dt_esg))

 the_hist = ESG_histo()
 names(the_hist) = c("Date", "ESG", "E_Score", "S_Score", "G_Score")
 the_hist = the_hist[, c("Date", "ESG")]
 print("--Head histo ESG--")
 print(head(the_hist))
 print("--Tail histo ESG--")
 print(tail(the_hist))

})

output$print_2 <- renderPrint({
```

```
  # print(rf_id())
  print(yf.get_long_names(stock_to_analyze()))
  # print()
  # print()

})

output$print_3 <- renderPrint({
  dt_return = Data_return()
  dt_ESG = fill_ESG_dt_new()
  names(dt_ESG) = c("Date", "ESG")
  df =  dplyr::left_join(dt_return, dt_ESG, by = c("Date" = "Date"))
  # print(rf_id())
  print("--Head left join--")
  print(head(df, 7))
  print("-- Tail left join --")
  print(tail(df, 7))

  # df1 =  dplyr::left_join(dt_return, dt_ESG, by = c("Date"))
  # print(head(df1, 7))
})

output$print_4 <- renderPrint({
  ## print(head(plotly_3D_dt()))
  # selected_tickers <- the_cmp_esg_sbl()
  # print(selected_tickers)
  #
  # df = yf_get_keys.stats(selected_tickers, full = FALSE)
  # print("--Keys Stats--")
```

```
# df


selected_tickers <- the_cmp_esg_sbl()

print(selected_tickers)

ticker = selected_tickers[1]

print("--Cas du premier ticker--")

ticker = base::toupper(ticker)

print(ticker)

url = paste0("https://finance.yahoo.com/quote/", ticker, "/key-statistics/")

cookies = c(

  A1 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-
Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",

  GUC = "AQABCAFm-EdnKEIeOgSl&s=AQAAAIJTL2AD&g=Zvb4qg",

  A3 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-
Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",

  A1S = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-
Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",

  cmp = "t=1728944112&j=1&u=1---&v=48",

  EuConsent = "CQFm5cAQFm5cAAOACKFRBLFgAAAAAAAAACiQAAAAAAA",

  PRF =
"t=ADP%2BKO%2BGIS%2BAAPL%2B%5EFVX%2BGNFT.PA%2BMSFT%2BA%2BGOOG%2BBA%2B
AL%2BRELIANCE.NS%2BVIIIX%2BFSKAX%2BFCNTX&newChartbetateaser=0%2C1719358224417
"

)


headers = c(

  accept =
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.7",

  `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",

  `cache-control` = "no-cache",

  pragma = "no-cache",
```

```
  priority = "u=0, i",
  `sec-ch-ua` = '"Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"',
  `sec-ch-ua-mobile` = "?0",
  `sec-ch-ua-platform` = '"Windows"',
  `sec-fetch-dest` = "document",
  `sec-fetch-mode` = "navigate",
  `sec-fetch-site` = "none",
  `sec-fetch-user` = "?1",
  `upgrade-insecure-requests` = "1",
  `user-agent` = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36"
 )
 response <- httr::GET(url, httr::add_headers(.headers=headers), httr::set_cookies(.cookies = cookies))

 content <- httr::content(response, "text")

 html_parsed <- rvest::read_html(response)

 tables = html_parsed %>%
  rvest::html_nodes('table') %>%
  rvest::html_table()

 print("--Info sur length table")
 print(length(tables))

 print("--Table 10--")
 print(tables[[10]])
```

```
  # print("--Table 11--")
  print("--Toute la Table--")
  # print(tables[[11]])
  print(tables)
  # print()
  # print()

 })


 output$print_5 <- renderPrint({
  # selected_tickers <- the_cmp_esg_sbl()
  # print(selected_tickers)
  # ticker = selected_tickers[1]
  # print("--Cas du premier ticker--")
  # ticker = base::toupper(ticker)
  # print(ticker)
  # url = paste0("https://finance.yahoo.com/quote/", ticker, "/key-statistics/")
  # cookies = c(
  #   A1 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
  #   GUC = "AQABCAFm-EdnKEIeOgSl&s=AQAAAIJTL2AD&g=Zvb4qg",
  #   A3 = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
  #   A1S = "d=AQABBHrmWGQCEJFwTQMyJ_UOfbY1ppboDwkFEgABCAFH-GYoZ-Uzb2UBAiAAAAcIeuZYZJboDwk&S=AQAAAqpFC6L7-87rqpuaAqspZ3M",
  #   cmp = "t=1728944112&j=1&u=1---&v=48",
  #   EuConsent = "CQFm5cAQFm5cAAOACKFRBLFgAAAAAAAAACiQAAAAAAAA",
  #   PRF = "t=ADP%2BKO%2BGIS%2BAAPL%2B%5EFVX%2BGNFT.PA%2BMSFT%2BA%2BGOOG%2BBA%2B
```

```
AL%2BRELIANCE.NS%2BVIIIX%2BFSKAX%2BFCNTX&newChartbetateaser=0%2C1719358224417
"

  # )

  #

  # headers = c(

  #     accept =
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.7",

  #   `accept-language` = "fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7",

  #   `cache-control` = "no-cache",

  #   pragma = "no-cache",

  #   priority = "u=0, i",

  #   `sec-ch-ua` = '"Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"',

  #   `sec-ch-ua-mobile` = "?0",

  #   `sec-ch-ua-platform` = '"Windows"',

  #   `sec-fetch-dest` = "document",

  #   `sec-fetch-mode` = "navigate",

  #   `sec-fetch-site` = "none",

  #   `sec-fetch-user` = "?1",

  #   `upgrade-insecure-requests` = "1",

  #   `user-agent` = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/129.0.0.0 Safari/537.36"

  # )

  # response <- httr::GET(url, httr::add_headers(.headers=headers), httr::set_cookies(.cookies =
cookies))

  #

  # content <- httr::content(response, "text")

  #

  # html_parsed <- rvest::read_html(response)

  #
```

```
# tables = html_parsed %>%
#   rvest::html_nodes('table') %>%
#   rvest::html_table()
#
# print("--Info sur length table--")
# print(length(tables))


# print("--En utilisant base--")
# # print(base::length(tables))
#
# val_measure = tables[[1]]
# print("--Head val_measure --")
# print(head(val_measure))
# val_measure = val_measure[, c(1,2)]
# print("--Head val_measure premier filtre --")
# print(head(val_measure))
# val_measure <- as.data.frame(val_measure)
# # names(val_measure) = c("Info", "Current")
# names(val_measure) = c("X1", "X2")
# print("--Head val_measure premier renommer entête --")
# print(head(val_measure))
#
# # Financial Highlights tab
# Fiscal_Year = tables[[2]]
# print("--fin year--")
# print(head(Fiscal_Year))
# Profitability = tables[[3]]
# print("--Prof--")
# print(head(Profitability))
```

```
# Management_Effectiveness = tables[[4]]
# print("--Man Eff--")
# print(head(Management_Effectiveness))
# INC.stat = tables[[5]]
# print("--Inc Stat--")
# print(head(INC.stat))
# BS = tables[[6]]
# print("--BS--")
# print(head(BS))
# CF = tables[[7]]
# print("--CF--")
# print(head(CF))
# St_price_h = tables[[8]]
# print("--Sp price--")
# print(head(St_price_h))
# Share_stats = tables[[9]]
# print("--Share Stats--")
# print(head(Share_stats))
# Div_split = tables[[10]]
# print("--Dividende--")
# print(head(Div_split))


# val_measure <- as.data.frame(val_measure)
# Fiscal_Year <- as.data.frame(Fiscal_Year)
# Profitability <- as.data.frame(Profitability)
# Management_Effectiveness <- as.data.frame(Management_Effectiveness)
# INC.stat <- as.data.frame(INC.stat)
# BS <- as.data.frame(BS)
# CF <- as.data.frame(CF)
```

```
# St_price_h <- as.data.frame(St_price_h)
# Share_stats <- as.data.frame(Share_stats)
# Div_split <- as.data.frame(Div_split)
#
# df_init = as.data.frame(matrix(NA, ncol = 2, nrow = 1))
#
# names(df_init) = c('X1', 'X2')
# df_init$X1 = c("Ticker")
# df_init$X2 = ticker

# df = rbind(df_init,
#         val_measure, Fiscal_Year, Profitability, Management_Effectiveness,
#         INC.stat, BS, CF, St_price_h, Share_stats, Div_split)
#
# names(df) = c('Info', 'Detail')
#
# print("--Pour le premier cas--")
# print(df)



# print(str(df_init))
# print(str(val_measure))
# print(str(Fiscal_Year))
# print(str(Profitability))
# print(str(Management_Effectiveness))
# print(str(INC.stat))
# print(str(BS))
# print(str(CF))
```

```
 # print(str(St_price_h))
 # print(str(Share_stats))
 # print(str(Div_split))

 # df = stats_tools()
 # print("--Current Valuation Measures--")
 # print(head(df))

 # print()
 # print()

})

################################################################################
################################################################################
################################ FIN DEBUGGAGE ################################
################################################################################
################################################################################

################################################################################
#################### EN UTILISANT LES DONNEES BENCHMARK ##################
################################################################################

# Pour le benchmark global
# output$benchmark_selector <- renderUI({
#   if(input$BNC_source == "Yahoo") {
#    selectInput(inputId = "Benchmarks",
#                label = strong("Select a benchmark"),
#                choices = c("", benchmark_list$Name),
```

```
#             selectize = TRUE)
#   } else {
#    selectInput(inputId = "Benchmarks",
#             label = strong("Select a benchmark"),
#             choices = c("", benchmark_list_Euro$Name),
#             options = list(maxOptions = 1617),
#             selectize = TRUE)
#   }
# })

# Mettre à jour les choix côté serveur
# observe({
#   if(input$BNC_source == "Yahoo") {
#    updateSelectizeInput(
#     session,
#     'Benchmarks',
#     choices = c("", benchmark_list$Name),
#     server = TRUE
#    )
#   } else {
#    updateSelectizeInput(
#     session,
#     'Benchmarks',
#     choices = c("", benchmark_list_Euro$Name),
#     server = TRUE
#    )
#   }
# })
```

```
BNC_selected <- reactive({
 if(input$Benchmarks != ""){

  # Rechercher le Symbol correspondant au Name

  benchmark_list_f <- benchmark_list

  print("name benchmark_list_f")
  print(names(benchmark_list_f))
  # En utilisant Yahoo ou Euronext comme source
  names(benchmark_list_f) <- c("Name", "Symbol")

  symbol <- benchmark_list_f$Symbol[benchmark_list_f$Name == input$Benchmarks]

  # En utilisant Euronext comme source
  # symbol <- benchmark_list_Euro$Ticker_adn[benchmark_list_Euro$Name == Name]

  return(symbol)

 }else{
  return(NULL)
 }

})

BNC_selected_short_n <- reactive({
 if(input$Benchmarks != ""){
```

```
   return(abbreviate_text(input$Benchmarks))

  }else{
   return(NULL)
  }

})



# Les indices ou benchmarks
BNC_prices <- reactive({
  # print(input$Benchmarks)
  # if(!is.null(BNC_selected)){
  if(input$Benchmarks != ""){
    result <- Get_bench_data(Name = input$Benchmarks,
                             from = as.Date(input$date[1]),
                             na_method = "keep",
                             # na_method = "previous",
                             # na_method = input$na_method, #Je l'utilisais avant
                             to = as.Date(input$date[2]),
                             # source = input$BNC_source
                             )

    # print("essaie sur result")
    # print(head(result))

    return(result)

  }else{
```

```
  return(NULL)
 }
})

BNC_return <- reactive({
 if(!is.null(dim(BNC_prices()))){

  Data <- BNC_prices()
  print('the_head of data')

  # print(head(Data))

  colnames(Data)[2] <- 'Close' #Renommer par close la seconde colonne
  Data <- Data %>%
   # tq_transmute(select = 2,
   tq_transmute(select = Close,
            mutate_fun = periodReturn,
            period=input$retfreq,
            type= input$log,
            col_rename = "Return")

 }else{
  return(NULL)
 }
})

BNC_return_after <- reactive({
 if(!is.null(dim(BNC_return()))){
  # dt_esg_prices()$close
```

```
  Data <- BNC_return()
  # print("--Return Benchmarch--")
  # print(dput(head(Data, 15)))

  proc_dt = process_data(Data)
  return(proc_dt)



 } else{
  return(NULL)
 }

})



output$BNC_dt <- renderDT({
 if(!is.null(BNC_prices())){
  # print(head(BNC_prices()))

  datatable(BNC_prices(),
        options = list(dom = 't',
                ordering = FALSE))

 }else{
  return(NULL)
 }

})
```

```
output$BNC_dt_return <- renderDT({
 if(!is.null(BNC_return())){
  # print(head(BNC_prices()))

  datatable(BNC_return(),
        options = list(dom = 't',
                ordering = FALSE))

 }else{
  return(NULL)
 }

})



# Stock, Risk free rate et Benchmark
Stock_RF_and_BNC <- reactive({
 if(!is.null(Data_return()) && !is.null(BNC_return()) && !is.null(RF_Data_return())){

  # Le stock
  stock_dt = Data_return()
  name_st = stock_to_analyze()
  stock_dt$Ticker = stock_to_analyze()
  names(stock_dt) = c('Date', 'Return', 'Ticker')
  # print(names(stock_dt))

  # Le risk_free rate
  rf_dt = RF_Data_return()
```

```
  rf_dt$Ticker = input$Rf_rate
  names(rf_dt) = c('Date', 'Return', 'Ticker')
  # print(names(rf_dt))

  # Le benchmark
  BNC_dt = BNC_return()
  # BNC_dt$Ticker = BNC_selected()
  BNC_dt$Ticker = BNC_selected_short_n()
  names(BNC_dt) = c('Date', 'Return', 'Ticker')

  R_stock_rf_BNC <- rbind(stock_dt, rf_dt, BNC_dt)

  names(R_stock_rf_BNC) = c('Date', 'Return', 'Ticker')

  R_stock_rf_BNC[, 'Return'] = round(R_stock_rf_BNC[, 'Return'], 4)

  # print(R_stock_rf_BNC)

  R_stock_rf_BNC

 }else{

  return(NULL)

 }

})

# Tableau des deux rentabilités
```

```
# output$St_BNC_and_rf_returns <- renderDT({
#   if(!is.null(Stock_RF_and_BNC())){
#     # Pour obtenir le pivot wider
#     # dt_return = Stock_and_rf_new()
#
#     dt_return = Stock_RF_and_BNC()
#
#     rownames(dt_return) = NULL
#
#     dt_return[, 2] = dt_return[, 2]*100
#
#     # dt_return$Return = round(dt_return$Return, 4)
#
#
#     dt_return <- DT::datatable(dt_return,
#                                extensions = 'Responsive',
#                                rownames = FALSE, options = list(
#                                  columnDefs = list(list(targets = "_all", className = "dt-center",
#                                                         width = "35px"))
#                                )
#     )
#   } else{
#     return(NULL)
#
#   }
#
#
# })
```

```
# Tableau des deux rentabilités
output$St_BNC_and_rf_returns <- renderDT({
 if(!is.null(Stock_RF_and_BNC())){
  dt_return = Stock_RF_and_BNC()
  rownames(dt_return) = NULL

  # Arrondir la colonne Return après la multiplication par 100
  dt_return[, 'Return'] = round(dt_return[, 'Return'] * 100, 4)

  dt_return <- DT::datatable(dt_return,
                    extensions = 'Responsive',
                    rownames = FALSE,
                    options = list(
                     columnDefs = list(list(
                      targets = "_all",
                      className = "dt-center",
                      width = "35px"
                     )),
                     # Formater l'affichage des nombres pour garder 4 décimales
                     formatRound = list(columns = 'Return', digits = 4)
                    )
  )

  dt_return
 } else{
  return(NULL)
 }
})
```

```
# Stock, Risk free rate et Benchmark après le choix de la méthode sur les NA
Stock_RF_and_BNC_after <- reactive({
  if(!is.null(Data_return()) && !is.null(BNC_return()) && !is.null(RF_Data_return())){
    # if(!is.null(Data_return()) && !is.null(RF_Data_return())){

    the_dt = Stock_RF_and_BNC()

    names(the_dt) = c('Date', 'Return', 'Ticker')

    # print("--Head Stock, RF et Bench --")
    # print(head(the_dt, 15))
    # print("--Tail Stock, RF et Bench --")
    # print(tail(the_dt, 15))

    proc_dt = process_data_gp(the_dt)

    return(proc_dt)

  }else{

    return(NULL)

  }

})


Stock_rf_and_BNC_new <- reactive({
  if (!is.null(Data_return()) && !is.null(BNC_return()) && !is.null(RF_Data_return())) {
```

```
  Dt <- Stock_RF_and_BNC()

  df <- Dt %>%
   tidyr::pivot_wider(
    names_from = "Ticker",
    values_from = "Return"
   )

  # Remplir les valeurs manquantes dans les colonnes 3 et 4
  df <- df %>%
   tidyr::fill(3:4, .direction = "down")

  # Si les premiers éléments sont encore des NA, les remplacer par la première valeur non-NA
  if (any(is.na(df[, 3]))) {
   first_non_na_3 <- df[which(!is.na(df[, 3]))[1], 3]
   df[is.na(df[, 3]), 3] <- first_non_na_3
  }

  if (any(is.na(df[, 4]))) {
   first_non_na_4 <- df[which(!is.na(df[, 4]))[1], 4]
   df[is.na(df[, 4]), 4] <- first_non_na_4
  }

  return(df)
 } else {
  return(NULL)
 }
})
```

```
# Créer un pivot wider
# # Et ajuster la valeur des risk free rate  et Benchamark avec l'option 'fill' ie colonnes 3 et 4
# Stock_rf_and_BNC_new <- reactive({
#   if(!is.null(Data_return()) && !is.null(BNC_return()) && !is.null(RF_Data_return())){
#   # if(!is.null(Data_return()) && !is.null(RF_Data_return())){
#    Dt <- Stock_RF_and_BNC()
#
#    # print(names(Dt))
#
#    # print(head(Dt, 10))
#
#    df <- Dt%>%
#     tidyr::pivot_wider("Date",
#                        names_from = "Ticker",
#                        values_from = "Return")
#
#    # Après le pivot on obtiendra 4 colonnes, Date, nom du stock, nom du Rf et nom du Benchmark
#
#    # print(dput(head(df)))
#
#    # print(head(df), 10)
#
#    # print(tail(df), 10)
#    # Troisième colonne
#    df = df %>%
#     tidyr::fill(3)
#
#    # print(head(df), 10)
```

```
#
#   na_elm = which(is.na(df[,3]))
#   non_nul_elm = which(!is.na(df[,3]))
#
#   # Quatrième colonne
#   df = df %>%
#     tidyr::fill(4)
#
#   # print(head(df), 10)
#
#   na_elm_1 = which(is.na(df[,4]))
#   non_nul_elm_1 = which(!is.na(df[,4]))
#
#   # Si les premiers elements sont encore des NA
#   if(length(na_elm)!=0){
#     if(length(non_nul_elm)!=0){
#       df[na_elm,3] = df[non_nul_elm[1],3]
#
#       # return(df)
#     }
#     df = df
#     # return(df)
#   }
#
#   if(length(na_elm_1)!=0){
#     if(length(non_nul_elm_1)!=0){
#       df[na_elm_1,4] = df[non_nul_elm_[1],4]
#
#       return(df)
```

```
#     }
#
#     return(df)
#   }
#
#   # }
#
#   # print(head(df, 10))
#   return(df)
#
#
#  }else{
#
#   return(NULL)
#
#  }
#
# })



stock_rf_and_BNC_new2 <- reactive({
 if (!is.null(Data_return()) && !is.null(BNC_return()) && !is.null(RF_Data_return())) {
  # if(!is.null(Data_return()) && !is.null(RF_Data_return())){
  stock_dt = Data_return_after()
  name_st = stock_to_analyze()
  names(stock_dt) = c('Date', name_st)

  # Risk free rate after
  rf_dt = RF_Data_return_after()
```

```
name_rf = input$Rf_rate
names(rf_dt) = c('Date', name_rf)

# Benchmark after
bnc_dt = BNC_return_after()
# name_bnc = BNC_selected()
name_bnc = BNC_selected_short_n()
names(bnc_dt) = c('Date', name_bnc)



# R_stock_rf_BNC <- dplyr::left_join(stock_dt, rf_dt, by = c("Date"))

R_stock_rf <- dplyr::full_join(stock_dt, rf_dt, by = "Date")

R_stock_rf_BNC <- dplyr::full_join(R_stock_rf, bnc_dt, by = "Date")

R_stock_rf_BNC[,2][is.na(R_stock_rf_BNC[,2])] <- 0
R_stock_rf_BNC[,3][is.na(R_stock_rf_BNC[,3])] <- 0
R_stock_rf_BNC[,4][is.na(R_stock_rf_BNC[,4])] <- 0
# R_stock_rf_BNC <- R_stock_rf_BNC %>%
#   tidyr::fill(c(2, 3, 4)) # J'ai décidé de remplir les lignes

# join the two dataframes by 'Date' and keep only rows where dates match
# R_stock_rf_BNC <- merge(stock_dt, rf_dt, by = "Date", all = TRUE)

R_stock_rf_BNC[, 2] = round(R_stock_rf_BNC[, 2], 4)

R_stock_rf_BNC[, 3] = round(R_stock_rf_BNC[, 3], 4)
```

```
   R_stock_rf_BNC[, 4] = round(R_stock_rf_BNC[, 4], 4)

   return(R_stock_rf_BNC)

  }else{

   return(NULL)

  }

})


# Avec le pivot wider
# dataframe Date, Ticker and benchmark
output$St_rf_BNC_returns_new2 <- renderDT({
  if(!is.null(Stock_RF_and_BNC())){
   # Pour obtenir le pivot wider

   dt_return = stock_rf_and_BNC_new2()

   rownames(dt_return) = NULL

   dt_return[, 2] = dt_return[, 2]
   dt_return[, 3] = dt_return[, 3]
   dt_return[, 4] = dt_return[, 4]

   # names(dt_return)[3] = input$Rf_rate
```

```
  dt_return <- DT::datatable(dt_return,
                    extensions = 'Responsive',
                    rownames = FALSE, options = list(
                      columnDefs = list(list(targets = "_all", className = "dt-center",
                                  width = "35px"))
                    )
  )

  return(dt_return)

 } else{
  return(NULL)

 }



})

# Optimisation des returns du stock, RF et Benchmark
Returns_dt_new_ <- reactive({
 # Vérification si aucun des éléments nécessaires n'est disponible
 if (is.null(Data_return()) && is.null(RF_Data_return()) && is.null(BNC_return())) {
  return(
   highchart() %>%
    hc_title(text = "Select at least one security to display the chart")
  )
 }

 # Préparation des données combinées non nulles
```

```
combined_data <- NULL



if (!is.null(RF_Data_return())) {
  dt_rf <- RF_Data_return_after()
  dt_rf$Ticker <- input$Rf_rate  # Identifier comme risk-free rate
  dt_rf$Return <- dt_rf$Return * 100
  combined_data <- dt_rf
}

if (!is.null(BNC_return())) {
  dt_bnc <- BNC_return_after()
  dt_bnc$Ticker <- input$Benchmarks  # Identifier comme benchmark
  dt_bnc$Return <- dt_bnc$Return * 100
  combined_data <- rbind(combined_data, dt_bnc)
}

if (!is.null(Data_return())) {
  dt <- Data_return_after()
  dt$Ticker <- stock_to_analyze()  # Ajouter une colonne pour identifier la source
  dt$Return <- round(dt$Return, 4) * 100
  combined_data <- rbind(combined_data, dt)
}

# Vérification finale si des données combinées existent
if (is.null(combined_data) || nrow(combined_data) == 0) {
  return(
    highchart() %>%
      hc_title(text = "Aucune donnée disponible pour afficher le graphique")
```

```
  )
 }

 # Création du graphique combiné
 return(
  combined_data %>%
   hchart(
    type = "line",
    hcaes(x = Date, y = Return, group = Ticker)
   ) %>%
   hc_yAxis(opposite = FALSE, labels = list(format = "{value}%")) %>%
   hc_tooltip(pointFormat = "{point.x: %Y-%m-%d} : {point.y:.4f}% ") %>%
   hc_yAxis(title = list(text = "Return (en %)")) %>%
   hc_title(text = paste0("Graphique comparatif : de ", input$date[1], " à ", input$date[2])) %>%
   hc_subtitle(text = paste0("Comparaison entre ", stock_to_analyze(),
                             if (!is.null(RF_Data_return())) paste0(", ", input$Rf_rate) else "",
                             if (!is.null(BNC_return())) paste0(", ", input$Benchmarks) else "")) %>%
   hc_exporting(
    enabled = TRUE,
    filename = paste0("Comparaison_returns_", input$date[1], "_", input$date[2])
   )
 )
})

# Old version
# Returns_dt_new_ <- reactive({
#   # Si stock null
#   # if(nchar(stock_to_analyze()) == 0){
#   # print("Data_return()")
```

```
#   # print(Data_return())
#   # print(dim(Data_return()))
#
#   # Si stock null
#   if (is.null(Data_return()) && is.null(dim(Data_return()))) {  # Correction de la condition
#     return(
#       highchart() %>%
#         hc_title(text = "Sélectionner un titre pour afficher le graphique")
#     )
#
#   # if (is.null(Data_return()) && is.null(dim(Data_return()))) {
#   #   showNotification("Sélectionner un titre pour afficher le graphique")
#   #   return(NULL)
#   } else {
#     # Si riskfree_rate et Original_BNC sont null
#     if (is.null(RF_Data_return()) && is.null(BNC_return())) {
#       dt <- Data_return_after()
#       dt$Return <- round(dt$Return, 4) * 100
#
#       return(
#         dt %>% hchart(
#           name = stock_to_analyze(),
#           'line',
#           hcaes(x = Date, y = Return),
#           color = the_colour()
#         ) %>%
#           hc_yAxis(title = list(text = "Return (in %)")) %>%
#             hc_title(text = paste0(stock_to_analyze(), " return chart : from ", input$date[1], " to ",
input$date[2])) %>%
```

```
#         hc_exporting(
#           enabled = TRUE,
#           filename = paste0(stock_to_analyze(), " chart : from", input$date[1], "to", input$date[2])
#         )
#     )
#     # Si riskfree_rate et RF_Data_return ne sont pas null et Original_BNC est null
#   } else if (!is.null(RF_Data_return()) && is.null(BNC_return())) {
#     dt <- Stock_and_rf_after()
#     dt[, 'Return'] <- dt[, 'Return'] * 100
#
#     return(
#       dt %>% hchart(
#         type = "line",
#         hcaes(x = Date, y = Return, group = Ticker)
#       ) %>%
#         hc_yAxis(opposite = FALSE, labels = list(format = "{value}%")) %>%
#         hc_tooltip(pointFormat = '{point.x: %Y-%m-%d} {point.y:.4f}% ') %>%
#         hc_yAxis(title = list(text = "Return")) %>%
#        hc_title(text = paste0(stock_to_analyze(), " and ", input$Rf_rate, " return comparative chart")) %>%
#         hc_subtitle(text = paste0("From ", input$date[1], " to ", input$date[2])) %>%
#         hc_exporting(
#           enabled = TRUE,
#           filename = paste0("Return comparative chart : from", input$date[1], "to", input$date[2])
#         )
#     )
#     # Si riskfree_rate, RF_Data_return et Original_BNC ne sont pas null
#   } else if (!is.null(RF_Data_return()) && !is.null(BNC_return())) {
#     dt <- Stock_RF_and_BNC()
```

```
#
#     dt[, 'Return'] <- dt[, 'Return'] * 100
#
#     return(
#       dt %>% hchart(
#         type = "line",
#         hcaes(x = Date, y = Return, group = Ticker)
#       ) %>%
#         hc_yAxis(opposite = FALSE, labels = list(format = "{value}%")) %>%
#         hc_tooltip(pointFormat = '{point.x: %Y-%m-%d} {point.y:.4f}% ') %>%
#         hc_yAxis(title = list(text = "Return")) %>%
#         hc_title(text = paste0(stock_to_analyze(), ", ", input$Benchmarks, " and ",
#                                input$Rf_rate, " return comparative chart")) %>%
#         hc_subtitle(text = paste0("From ", input$date[1], " to ", input$date[2])) %>%
#         hc_exporting(
#           enabled = TRUE,
#           filename = paste0("Return comparative chart : from", input$date[1], "to", input$date[2])
#         )
#     )
#   }
# })


output$graphchart_hc_ <- renderHighchart({

  Returns_dt_new_()

})
```

```
output$summary_stats_new <- renderDT({
 # Fonction utilitaire pour calculer les métriques
 calculate_metrics <- function(data_before, data_after) {
  data_before$Return <- as.numeric(data_before$Return)
  data_after$Return <- as.numeric(data_after$Return)

  # list(
  #   sum(is.na(data_before$Return)),
  #   round(mean(data_before$Return, na.rm = TRUE), 4),
  #   round(mean(data_after$Return, na.rm = TRUE), 4),
  #   nrow(data_before),
  #   nrow(data_after)
  # )

  return(c(
   sum(is.na(data_before$Return)),
   round(mean(data_before$Return, na.rm = TRUE), 4),
   round(mean(data_after$Return, na.rm = TRUE), 4),
   nrow(data_before),
   nrow(data_after)
  ))

 }

 # Initialiser la liste pour les métriques comparatives
 metrics_list <- list(
  Metrics = c("Nb. of NA",
```

```
        "Mean before",
        "Mean after",
        "Nb. of rows before",
        "Nb. of rows after")
)

# Vérifier et calculer les métriques pour le stock
if (!is.null(Data_return())) {
  dt_before <- Data_return()
  dt_after <- Data_return_after()
  metrics_list[[toupper(stock_to_analyze())]] <- calculate_metrics(dt_before, dt_after)
}

# Vérifier et calculer les métriques pour le Risk Free rate
if (!is.null(RF_Data_return())) {
  RF_before <- RF_Data_return()
  RF_after <- RF_Data_return_after()
  metrics_list[[toupper(input$Rf_rate)]] <- calculate_metrics(RF_before, RF_after)
}

# Vérifier et calculer les métriques pour le Benchmark
if (!is.null(BNC_return())) {
  bench_before <- BNC_return()
  bench_after <- BNC_return_after()
  # name_bnc <- BNC_selected()
  name_bnc <- BNC_selected_short_n()
  metrics_list[[toupper(name_bnc)]] <- calculate_metrics(bench_before, bench_after)
}
```

```
  # Si aucune donnée disponible, retourner NULL
  if (length(metrics_list) <= 1) {
   return(NULL)
  }

  # print("class")
  # print(class((metrics_list)))
  # print("dput metrics_list")
  # print(dput(metrics_list))

  # Créer la matrice finale et le DataTable
  # comparison_df <- as.data.frame(metrics_list)
  comparison_df <- as.data.frame(metrics_list, check.names = FALSE)  # Ajout de check.names = FALSE
  comparison_matrix <- as.matrix(comparison_df[,-1, drop = FALSE])
  rownames(comparison_matrix) <- comparison_df$Metrics

  datatable(
   comparison_matrix,
   options = list(
    dom = 't',
    ordering = FALSE
   )
  )
 })

 ##########################################################################
 ################### EN UTILISANT LES DONNEES RISK FREE RATE ##############
 ##########################################################################
```

```
# Obtenir les données du benchmark choisi
dt_rf_prices <- reactive({
 if(!is.null(rf_info())){

  # Cette fonction est la plus récente
  # rf_data <-  ready_hcDt_new(ticker_info = rf_info(),
  #                   "D",
  #                   input$date[1],
  #                   input$date[2])

  # rf_data <-  fetch_inv_prices(ticker_info = rf_info(),
  #                   from = input$date[1],
  #                   to = input$date[2],
  #                   time_frame = 'Daily')

  RF_id_new = switch(input$Rf_rate,
                "OAT 1y" = 23769,
                'Bund 1y' = 23684,
                'T-Note 1y' = 23700,
                "OAT 5y" = 23773,
                'Bund 5y' = 23688,
                'T-Note 5y' = 23703,
                "OAT 10y" = 23778,
                'Bund 10y' = 23693,
                'T-Note 10y' = 23705
  )

  # print('sur mon test')
```

```
  # rf_data <- ready_hcDt_new(ticker_info = rf_info(), "D", input$date[1], input$date[2])
  # rf_data <- ready_hcDt(ticker_id = rf_id(), period = 'MAX', max_retries = 30)



  # Cette version aussi fontionne
  # rf_data <- ready_hcDt(ticker_id = rf_id(), max_retries = 30)
  rf_data <- ready_hcDt(ticker_id = RF_id_new, max_retries = 30)

  # print(rf_id())
  # print(RF_id_new)

  # rf_data <- Financial.data::INV_hcDt(ticker_id = rf_id(), max_retries = 10)
  # rf_data <- Financial.data::INV_hcDt(ticker_id = RF_id_new, max_retries = 10)

  rf_data$Date <- as.Date(rf_data$Date)

  rf_data <- dplyr::as_tibble(rf_data) %>%
    dplyr::filter(Date >= as.Date(input$date[1])) %>%
    dplyr::filter(Date <= as.Date(input$date[2]))

  return(rf_data)

 }else{
  return(NULL)
 }
}
)
```

```
# Partie rentabilité BEnchmarcks
# Choix de la rentabilité
RF_Data_return <- reactive({
 if(!is.null(dim(dt_rf_prices()))){
  # dt_esg_prices()$close

  Data <- dt_rf_prices()
  print("Data --")
  print(head(Data, 2))

  Data <- Data %>%
   tq_transmute(select = Close,
           mutate_fun = periodReturn,
           period=input$retfreq,
           type= input$log,
           col_rename = "Return")

  # Data <- dplyr::as_tibble(Data) %>%
  #   dplyr::filter(Date >= as.Date(input$date[1])) %>%
  #   dplyr::filter(Date <= as.Date(input$date[2]))


  # names(Data)[1] = "Date"
  # print(names(Data))

  return(Data)

 } else{
```

```
   return(NULL)
  }



})


RF_Data_return_after <- reactive({
  if(!is.null(dim(RF_Data_return()))){
    # dt_esg_prices()$close

    Data <- RF_Data_return()
    # print("--Return Benchmarch--")
    # print(dput(head(Data, 15)))

    proc_dt = process_data(Data)
    return(proc_dt)



  } else{
    return(NULL)
  }



})

# Afficher les statistiques sommaires dans un tableau comparatif
output$summary_stats <- renderDT({
  # Vérifier si les données principales existent
```

```
if (!is.null(Data_return())) {
 # Créer la base du dataframe comparatif
 # metrics_list <- list(
 #   Metrics = c("Number of NA",
 #             "Mean before processing",
 #             "Mean after processing",
 #             "Number of rows before",
 #             "Number of rows after")
 # )

 metrics_list <- list(
  Metrics = c("Nb. of NA",
              "Mean before",
              "Mean after",
              "Nb. of rows before",
              "Nb. of rows after")
 )

 # Ajouter les métriques pour dt
 dt_before <- Data_return()
 dt_after <- Data_return_after()

 # S'assurer que Return est numérique
 dt_before$Return <- as.numeric(dt_before$Return)
 dt_after$Return <- as.numeric(dt_after$Return)

 metrics_list[[toupper(stock_to_analyze())]] <- c(
  sum(is.na(dt_before$Return)),
  round(mean(dt_before$Return, na.rm = TRUE), 4),
```

```
  round(mean(dt_after$Return, na.rm = TRUE), 4),
  nrow(dt_before),
  nrow(dt_after)
)

# Ajouter les métriques pour bench si le benchmark existe
if(!is.null(RF_Data_return())) {
  RF_before <- RF_Data_return()
  RF_after <- RF_Data_return_after()

  # S'assurer que Return est numérique
  RF_before$Return <- as.numeric(RF_before$Return)
  RF_after$Return <- as.numeric(RF_after$Return)

  metrics_list[[toupper(input$Rf_rate)]] <- c(
    sum(is.na(RF_before$Return)),
    round(mean(RF_before$Return, na.rm = TRUE), 4),
    round(mean(RF_after$Return, na.rm = TRUE), 4),
    nrow(RF_before),
    nrow(RF_after)
  )
}

# Créer la matrice finale
comparison_df <- as.data.frame(metrics_list)
comparison_matrix <- as.matrix(comparison_df[,-1, drop = FALSE])
rownames(comparison_matrix) <- comparison_df$Metrics

datatable(comparison_matrix,
```

```
          options = list(dom = 't',
                         ordering = FALSE))
  } else {
    return(NULL)
  }
})



# Afficher les statistiques sommaires dans un tableau comparatif
# output$summary_stats <- renderDT({
#   if (is.null(Data_return())) {
#     # Créer la base du dataframe comparatif
#     metrics_list <- list(
#       Metrics = c("Number of NA",
#                   "Mean before processing",
#                   "Mean after processing",
#                   "Number of rows before",
#                   "Number of rows after")
#     )
#
#     # Ajouter les métriques pour dt
#     dt_before <- Data_return()
#     dt_after <- Data_return_after()
#     # S'assurer que Return est numérique
#     dt_before$Return <- as.numeric(dt_before$Return)
#     dt_after$Return <- as.numeric(dt_after$Return)
#
#     metrics_list$dt <- c(
#       sum(is.na(dt_before$Return)),
```

```
#     round(mean(dt_before$Return, na.rm = TRUE), 4),
#     round(mean(dt_after$Return, na.rm = TRUE), 4),
#     nrow(dt_before),
#     nrow(dt_after)
#   )
#
#   names(metrics_list)[2] = toupper(stock_to_analyze())
#
#   # Ajouter les métriques pour bench si le benchmark est activé
#   if(!is.null(RF_Data_return())) {
#     RF_before <- RF_Data_return()
#     RF_after <- RF_Data_return_after()
#
#     # S'assurer que Return est numérique
#     RF_before$Return <- as.numeric(RF_before$Return)
#     RF_after$Return <- as.numeric(RF_after$Return)
#
#     metrics_list$Bench <- c(
#       sum(is.na(RF_before$Return)),
#       round(mean(RF_before$Return, na.rm = TRUE), 4),
#       round(mean(RF_after$Return,  na.rm = TRUE), 4),
#       nrow(RF_before),
#       nrow(RF_after)
#     )
#
#     names(metrics_list)[3] = toupper(input$Rf_rate)
#   }
#
#   # Créer la matrice finale
```

```
#    comparison_df <- as.data.frame(metrics_list)
#    comparison_matrix <- as.matrix(comparison_df[,-1, drop = FALSE])
#    rownames(comparison_matrix) <- comparison_df$Metrics
#
#    datatable(comparison_matrix,
#              options = list(dom = 't',
#                             ordering = FALSE))
#
#   }else{
#    return(NULL)
#   }
#
#
# })

# RENDEMENTS Benchmark
output$rf_returns <- renderDT({
 if(!is.null(RF_Data_return())){
   # dt_return = RF_Data_return()
   dt_return = RF_Data_return_after()

   rownames(dt_return) = NULL

   # dt_return$Return = round(dt_return$Return, 4)*100
   dt_return[, 'Return'] = round(dt_return[, 'Return'] * 100, 4)

   names(dt_return) = c("Date", "Return (in %)")

   # dt_return = dt_return %>%
```

```
  #  formatPercentage(c("Return"),2)

  dt_return <- DT::datatable(dt_return,
                    extensions = 'Responsive',
                    rownames = FALSE, options = list(
                     info = FALSE,
                     columnDefs = list(list(targets = "_all", className = "dt-center",
                                  width = "35px")),
                     # Formater l'affichage des nombres pour garder 4 décimales
                     formatRound = list(columns = 'Return (in %)', digits = 4)
                    )
  )

  dt_return



 } else{
  return(NULL)

 }



})

# Mettre les données en timeserie
# Au lieu de Faire un left join pour les deux base de données

Stock_and_rf <- reactive({
 if(!is.null(Data_return()) && !is.null(RF_Data_return())){
```

```
stock_dt = Data_return()
# name_st = paste0(stock_to_analyze()," return (in %)")
name_st = stock_to_analyze()
stock_dt$Ticker = stock_to_analyze()

# names(stock_dt) = c('Date', name_st)
names(stock_dt) = c('Date', 'Return', 'Ticker')

# print(names(stock_dt))

rf_dt = RF_Data_return()
# name_rf = paste0(input$Rf_rate," return (in %)")
# name_rf = input$Rf_rate
rf_dt$Ticker = input$Rf_rate
# names(rf_dt) = c('Date', name_rf)
names(rf_dt) = c('Date', 'Return', 'Ticker')

# print(names(rf_dt))

Rstock_Rrf <- rbind(stock_dt, rf_dt)

names(Rstock_Rrf) = c('Date', 'Return', 'Ticker')

Rstock_Rrf[, 'Return'] = round(Rstock_Rrf[, 'Return'], 4)

# Merge stock returns with baseline
# Rstock_Rrf <- dplyr::full_join(stock_dt, rf_dt, by = "Date")
# Rstock_Rrf <- dplyr::left_join(stock_dt, rf_dt, by = c("Date" = "Date"))
```

```
    # Rstock_Rrf[, name_st] = round(Rstock_Rrf[, name_st], 4)

    # Rstock_Rrf[, name_rf] = round(Rstock_Rrf[, name_rf], 4)

    # print(Rstock_Rrf)

    Rstock_Rrf

  }else{

    return(NULL)

  }

})



Stock_and_rf_after <- reactive({
  if(!is.null(Data_return()) && !is.null(RF_Data_return())){

    stock_dt = Stock_and_rf()

    names(stock_dt) = c('Date', 'Return', 'Ticker')

    # print("--Head Stock et Bench --")
    # print(head(stock_dt, 15))
    # print("--Tail Stock et Bench --")
    # print(tail(stock_dt, 15))
```

```
   proc_dt = process_data_gp(stock_dt)

   return(proc_dt)

  }else{

   return(NULL)

  }

})



# Créer un pivot wider
# Et ajuster la valeur des risk free rate avec l'option 'fill'
Stock_and_rf_new <- reactive({
  if(!is.null(Data_return()) && !is.null(RF_Data_return())){
   Dt <- Stock_and_rf()

   # print(names(Dt))

   # print(head(Dt, 10))

   df <- Dt%>%
    tidyr::pivot_wider("Date",
                names_from = "Ticker",
                values_from = "Return")
```

```
# print(dput(head(df)))


# print(head(df), 10)


# print("ok")


# print(tail(df), 10)
# En supposant que nous aurons toujours trois colonnes
# df %>% fill(names(df)[3])
# Ou
# if(!all((is.na(df[,3])))){
df = df %>%
  tidyr::fill(3)


# print(head(df), 10)


na_elm = which(is.na(df[,3]))
non_nul_elm = which(!is.na(df[,3]))


# Si les premiers elements sont encore des NA
if(length(na_elm)!=0){
  if(length(non_nul_elm)!=0){
    df[na_elm,3] = df[non_nul_elm[1],3]


    return(df)
  }
```

```
    return(df)
  }

  # }

  # print(head(df, 10))
  return(df)


 }else{

  return(NULL)

 }

})


stock_and_rf_new2 <- reactive({

 if(!is.null(Data_return()) && !is.null(RF_Data_return())){
  # stock_dt = Data_return() #remplacer ici par Data_return_after()
  stock_dt = Data_return_after()
  name_st = stock_to_analyze()
  names(stock_dt) = c('Date', name_st)

  # rf_dt = RF_Data_return() #ancien #remplacer ici RF_Data_return_after()
  rf_dt = RF_Data_return_after()
  name_rf = input$Rf_rate
```

```
names(rf_dt) = c('Date', name_rf)

# Rstock_Rrf <- dplyr::left_join(stock_dt, rf_dt, by = c("Date"))

Rstock_Rrf <- dplyr::full_join(stock_dt, rf_dt, by = "Date")

# join the two dataframes by ‘Date’ and keep only rows where dates match
# Rstock_Rrf <- merge(stock_dt, rf_dt, by = "Date", all = TRUE)

Rstock_Rrf[, 2] = round(Rstock_Rrf[, 2], 4)

Rstock_Rrf[, 3] = round(Rstock_Rrf[, 3], 4)

# df = Rstock_Rrf

# print(head(df), 5)

# # print(names(df))
# sav_name = names(df)[3]
#
# # renommer la colonne du rf
# names(df)[3] = "rf"
#
# # print(names(df))
#
#
# df = df %>% fill(rf)
#
# na_elm = which(is.na(df,"rf"]))
```

```
  #
  # # print(na_elm)
  # non_nul_elm = which(!is.na(df[,"rf"]))
  #
  # # Si les premiers elements sont encore des NA
  # if(length(na_elm)!=0){
  #   if(length(non_nul_elm)!=0){
  #
  #     df[na_elm,"rf"] = df[non_nul_elm[1],"rf"]
  #
  #   }
  #
  # }

  # names(df)[3] = sav_name

  # Rstock_Rrf = df

  return(Rstock_Rrf)

 }else{

  return(NULL)

 }

})
```

```
# Avec le pivot wider
# dataframe Date, Ticker and benchmark
output$St_and_rf_returns_new2 <- renderDT({
 if(!is.null(Stock_and_rf())){
  # Pour obtenir le pivot wider

  dt_return = stock_and_rf_new2()

  rownames(dt_return) = NULL

  dt_return[, 2] = dt_return[, 2]
  dt_return[, 3] = dt_return[, 3]



  # names(dt_return)[3] = input$Rf_rate

  dt_return <- DT::datatable(dt_return,
                    extensions = 'Responsive',
                    rownames = FALSE, options = list(
                     columnDefs = list(list(targets = "_all", className = "dt-center",
                                  width = "35px"))
                    )
  )

  return(dt_return)

 } else{
  return(NULL)
```

```
  }


})


fill_ESG_dt <- reactive({
  if(!is.null(Stock_and_rf())){
    dt_ESG = esg_score_DT_full()
    dt_ESG = dt_ESG[, c("Date", "ESG")]

    start_date <- as.Date(input$date[1])
    end_date <- as.Date(input$date[2])

    # Générez une séquence de dates entre start_date et end_date (inclusivement) pour les jours ouvrables (business days)
    date_range <- base::seq(from = start_date, to = end_date, by = "days")
    jours_semaine <- c("lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche")

    if (any(weekdays(date_range, abbreviate = FALSE) %in% jours_semaine)){
     workday_dates <- date_range[weekdays(date_range) %in% c("lundi", "mardi", "mercredi", "jeudi", "vendredi")]

    } else{
     jours_semaine_en <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
     workday_dates <- date_range[weekdays(date_range, abbreviate = FALSE) %in% jours_semaine_en]
    }

    # Créez une dataframe avec les dates
```

```
  df <- data.frame(Date = workday_dates)

  # Fusionnez les dataframes en utilisant la colonne "Date"
  merged_df <- merge(df, dt_ESG, by = "Date", all.x = TRUE)

  df = merged_df%>%
   tidyr::fill(ESG)

  na_elm = which(is.na(df[,"ESG"]))
  non_nul_elm = which(!is.na(df[,"ESG"]))

  # Si les premiers elements sont encore des NA
  if(length(na_elm)!=0){
   if(length(non_nul_elm)!=0){
    df[na_elm,"ESG"] = 0

    # return(df)
   }

   # return(df)
  }

  return(df)

 }else{

 }
})
```

```
 fill_ESG_dt_new <- reactive({

  if(!is.null(Data_return())){

    dt_ESG = esg_score_DT_full()

    dt_ESG = dt_ESG[, c("Date", "ESG")]


    start_date <- as.Date(input$date[1])

    end_date <- as.Date(input$date[2])


    # Générez une séquence de dates entre start_date et end_date (inclusivement) pour les jours ouvrables (business days)

    date_range <- base::seq(from = start_date, to = end_date, by = "days")

    jours_semaine <- c("lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche")


    if (any(weekdays(date_range, abbreviate = FALSE) %in% jours_semaine)){

     workday_dates <- date_range[weekdays(date_range) %in% c("lundi", "mardi", "mercredi", "jeudi", "vendredi")]


    } else{

     jours_semaine_en <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")

     workday_dates <- date_range[weekdays(date_range, abbreviate = FALSE) %in% jours_semaine_en]

    }


    # Créez une dataframe avec les dates

    df <- data.frame(Date = workday_dates)


    # Fusionnez les dataframes en utilisant la colonne "Date"

    merged_df <- merge(df, dt_ESG, by = "Date", all.x = TRUE)


    df = merged_df%>%
```

```
    tidyr::fill(ESG)

  na_elm = which(is.na(df[,"ESG"]))
  non_nul_elm = which(!is.na(df[,"ESG"]))

  # Si les premiers elements sont encore des NA
  if(length(na_elm)!=0){
    if(length(non_nul_elm)!=0){
      df[na_elm,"ESG"] = 0

    }

  }

  return(df)

 }else{

  return(NULL)

 }
})



# Donnée pour le calul des sharpe ratio
St_rf_ESG <- reactive({

 ######################################################
 # créer d'autres conditions
```

```
# Du moment où on a choisi le titre
# On a ses données ESG
# Et on saisi le Rf on peut afficher le tableau

if(!is.null(Data_return())){
 # Si j'ai les données ESG

 if(!is.null(esg_score_DT_full())){

  # J'ai décidé de remplir le rf
  if (!input$benchmark){
   print("Ici je n'ai pas séléctionné de benchmark")
   if(nchar(input$Rf_rate)== 0){
    dt = plotly_3D_dt()

    dt[, 'Return'] = round(dt[, 'Return'], 4)
    print('voici mes données date, return et esg')
    print(tail(dt, 3))
    #dt$rf = input$rfrate_new

    return(dt)

   }else{
    # Sélection du Benchmark

    if(!is.null(Stock_and_rf())){
     # Pour obtenir le pivot wider
     dt_return = stock_and_rf_new2()
```

```
  dt_ESG = fill_ESG_dt()

  # Merge and
  # join the two dataframes by ‘Date’ and keep only rows where dates match
  df =  merge(dt_return, dt_ESG, by = "Date", all = FALSE)

  # df = dplyr::left_join(dt_return, dt_ESG, by = ('Date' = 'Date'))

  return(df)

 } else{
  # envoyer ici les données dates, return esg

  dt = plotly_3D_dt()

  dt[, 'Return'] = round(dt[, 'Return'], 4)
  print('voici mes données date, return et esg')
  print(tail(dt, 3))
  #dt$rf = input$rfrate_new

  return(dt)

  # return(NULL)
 }

}
# Le tableau aura les colonnes suivantes :
# Date, Ticker_Return, ESG, Rf, Sharpe_Ratio,Modified_SR
```

```
}else{ # Fin condition choix du riskfree directement

  # dernière partie à corriger
  # Ici j'ai décider d'utiliser Index_return comme rf
  if(nchar(input$Rf_rate) != 0 && !is.null(RF_Data_return())){
    print('les données rf sont dispo')
    dt_return = stock_and_rf_new2()

    dt_ESG = fill_ESG_dt()
    # Merge and
    # join the two dataframes by ‘Date’ and keep only rows where dates match
    df =  merge(dt_return, dt_ESG, by = "Date", all = FALSE)

    return(df)



  }else{

    return(NULL)

  }
}


} else{ # Fin condition ESG
  return(NULL)
```

```
 }

}else{
 return(NULL)
}

#########################################################

# if(!is.null(Stock_and_rf())){
#   # Pour obtenir le pivot wider
#   dt_return = stock_and_rf_new2()
#
#   # rownames(dt_return) = NULL
#
#   # dt_return[, 2] = dt_return[, 2]
#   # dt_return[, 3] = dt_return[, 3]
#
#
#   if(!is.null(esg_score_DT_full())){
#
#    dt_ESG = fill_ESG_dt()
#
#    # Merge and
#    # join the two dataframes by 'Date' and keep only rows where dates match
#    df =  merge(dt_return, dt_ESG, by = "Date", all = FALSE)
#
#    # df = left_join(dt_return, dt_ESG, by = ('Date' = 'Date'))
#
#    return(df)
```

```
  #
  #   }else{
  #
  #    return(NULL)
  #
  #   }
  #
  # } else{
  #   return(NULL)
  #
  # }



})



output$St_rf_ESG_data <- renderDT({
  if(!is.null(St_rf_ESG())){
    dt = St_rf_ESG()

    rownames(dt) = NULL

    # dt[, 2] = dt[, 2]*100
    # dt[, 3] = dt[, 3]*100

    dt <- DT::datatable(dt,
                        extensions = 'Responsive',
                        rownames = FALSE, options = list(
                          columnDefs = list(list(targets = "_all", className = "dt-center",
```

```
                                width = "35px"))
                )
  )

  dt

 }else{
  return(NULL)

 }
})



# Tableau des deux rentabilités
output$St_and_rf_returns <- renderDT({
 if(!is.null(Stock_and_rf())){
  # Pour obtenir le pivot wider
  # dt_return = Stock_and_rf_new()

  dt_return = Stock_and_rf()

  rownames(dt_return) = NULL

  dt_return[, 2] = dt_return[, 2]*100

  # dt_return$Return = round(dt_return$Return, 4)



  dt_return <- DT::datatable(dt_return,
```

```
                    extensions = 'Responsive',
                    rownames = FALSE, options = list(
                      columnDefs = list(list(targets = "_all", className = "dt-center",
                                             width = "35px"))
                    )
    )
  } else{
    return(NULL)

  }



})

# Test
# Avec le pivot wider
# output$St_and_rf_returns_new <- renderDT({
#   if(!is.null(Stock_and_rf())){
#     # Pour obtenir le pivot wider
#
#     dt_return = Stock_and_rf_new()
#
#     rownames(dt_return) = NULL
#
#     dt_return[, 2] = dt_return[, 2]*100
#     dt_return[, 3] = dt_return[, 3]*100
#
#     dt_return <- DT::datatable(dt_return,
#                                extensions = 'Responsive',
```

```
#                       rownames = FALSE, options = list(
#                         columnDefs = list(list(targets = "_all", className = "dt-center",
#                                                width = "35px"))
#                       )
#    )
#
#    return(dt_return)
#
#   } else{
#    return(NULL)
#
#   }
#
#
# })



# Création des variables
# Rendement anunualisé
Rend_Annuel <- reactive({

 if(!is.null(dt_esg_prices())){
   # Remplacer Data() par mes données
   Data <- dt_esg_prices()

   Data %>%
    tq_transmute(select = Adjusted,
                 mutate_fun = periodReturn,
                 period="yearly",
```

```
            type= input$log,
            col_rename = "Return")

 }else{
  return(NULL)
 }

})


returns_dt <- reactive({

 if (is.null(RF_Data_return()))  {

  fig <- plot_ly(Data_return(),
            x = ~Date,
            y = ~Return,
            xlab = "Time",
            ylab = "Price",
            type = 'scatter',
            name = 'Stock', mode = 'lines')

  # fig <- fig %>% add_trace(y = ~Data_return()$Return, name = 'Stock', mode = 'lines', col = 'red')

 } else {
  #CORRECT
  fig <- plot_ly(Stock_and_rf(), x=~Stock_and_rf()$Date)
  fig <- fig %>% add_trace(y = ~Stock_and_rf()[,2], name = 'Stock', mode = 'lines', col = 'red')
  fig <- fig %>% add_trace(y = ~Stock_and_rf()[,3], name = 'Benchmark', mode = 'lines', col = 'green')
```

```
  }

})

# Reactive expression pour traiter les NA selon la méthode choisie
process_data <- function(df) {
  switch(input$na_method,
         "omit" = {
           df %>% na.omit()
         },
         "mean" = {
           df %>%
             dplyr::mutate(Return = replace(Return,
                                            is.na(Return),
                                            mean(Return, na.rm = TRUE)))
         },
         "zero" = {
           df %>%
             dplyr::mutate(Return = replace(Return,
                                            is.na(Return),
                                            0))
         },
         "previous" = {
           df %>%
             tidyr::fill(Return, .direction = "down")
         })

}
```

```
process_data_gp <- function(df) {
  switch(input$na_method,
         "omit" = {
           df %>% na.omit()
         },
         "mean" = {
           df %>%
             dplyr::group_by(Ticker) %>%
             dplyr::mutate(Return = replace(Return,
                                            is.na(Return),
                                            mean(Return, na.rm = TRUE))) %>%
             dplyr::ungroup()
         },
         "zero" = {
           df %>%
             dplyr::mutate(Return = replace(Return,
                                            is.na(Return),
                                            0))
         },
         "previous" = {
           df %>%
             dplyr::group_by(Ticker) %>%
             tidyr::fill(Return, .direction = "down") %>%
             dplyr::ungroup()
         })
}

returns_dt_new <- reactive({
```

```
  if (is.null(RF_Data_return())) {


   if(!is.null(Data_return())){
    # dt <- Data_return()


    dt <- Data_return_after()


    dt$Return = round(dt$Return, 4)*100
    # print(head(dt))


    dt %>% hchart(
     name = stock_to_analyze(),
     'line',
     # 'line',
     hcaes(x = Date, y = Return),
     # color = "steelblue"
     color =  the_colour()
    )%>%
     hc_yAxis(title = list(text = "Return (in %)"))%>%
     hc_title(text = paste0(stock_to_analyze()," return chart : from ",input$date[1], " to ", input$date[2])) %>%
     hc_exporting(
      enabled = TRUE, # always enabled,
      filename = paste0(stock_to_analyze()," chart : from",input$date[1], "to", input$date[2])
     )


    # highchart() %>%
    #  hc_add_series(
```

```
    #    data = dt,
    #    type = "line",
    #    hcaes(x = Date, y = Return),
    #    name = stock_to_analyze(),
    #    color = the_colour()
    #  ) %>%
    #  hc_yAxis(title = list(text = "Return (in %)"))%>%
    #  hc_title(text = paste0(stock_to_analyze()," chart : from ",input$date[1], " to ", input$date[2]))
%>%
    #  hc_exporting(
    #    enabled = TRUE, # always enabled,
    #    filename = paste0(stock_to_analyze()," chart : from",input$date[1], "to", input$date[2])
    #  )




   }else{
     return(NULL)
   }



  } else {
    #CORRECT
    if(is.null(Data_return())){
     return(NULL)
    }
    # dt = Stock_and_rf()
    dt = Stock_and_rf_after()
```

```
  dt[, 'Return'] = dt[, 'Return']*100


  dt %>%
   # dplyr::filter(date >= "2017-01-01" & date < "2018-01-01") %>%
   hchart(., type = "line",
        hcaes(x = Date,
            y = Return,
            group = Ticker)) %>%
   hc_yAxis(opposite = FALSE,
         labels = list(format = "{value}%")) %>%
   hc_tooltip(pointFormat = '{point.x: %Y-%m-%d}
                {point.y:.4f}% ')%>%
   hc_yAxis(title = list(text = "Return"))%>%
   hc_title(text = paste0(stock_to_analyze(),
                  " and ",
                  input$Rf_rate, " return comparative chart")) %>%
   hc_subtitle(text = paste0("From ",input$date[1], " to ", input$date[2])) %>%
   hc_exporting(
    enabled = TRUE, # always enabled,
    filename = paste0("Return comparative chart : from",input$date[1], "to", input$date[2])
   )

 }

})



output$graphchart <- renderPlotly({
```

```
 returns_dt()

})


output$graphchart_hc <- renderHighchart({

 returns_dt_new()

})

plotly_3D_dt <- reactive({

 if(!is.null(Data_return())){
  # Pour obtenir le pivot wider

  dt_return = Data_return()
  names(dt_return) = c("Date", "Return")
  # print(tail(dt_return))

  if(!is.null(esg_score_DT_full())){
   dt_ESG = fill_ESG_dt_new()
   names(dt_ESG) = c("Date", "ESG")
   # print(tail(dt_ESG))



   # join the two dataframes by 'Date' and keep only rows where dates match
```

```
   # df =  merge(dt_return, dt_ESG, by = "Date", all = FALSE)

   # df =  dplyr::left_join(dt_return, dt_ESG, by = c("Date" = "Date"))

   df =  dplyr::left_join(dt_return, dt_ESG, by = c("Date"))



   # df = df[, c('Date', 'Return', 'ESG')]

   names(df) = c('Date', 'Return', 'ESG')

   # print("mesdonnées 3 D")
   # print(head(df))

   # print(tail(df))
   # print("tout est ok ici")
   return(df)

  }else{

   return(NULL)

  }

 } else{
  return(NULL)

 }
```

```
})


plotly_3D_dt_new <- reactive({

 if(!is.null(dim(plotly_3D_dt()))){

  df = plotly_3D_dt()

  # print("nouveau tout est ok ici")

  df = na.omit(df)

  # print(tail(df))

  # names(df)[2] = 'Return'

  # df = df[, c('Date', 'Return', 'ESG')]

  df$Volatility = NA



  # print("début_mesdonnées 3 D")

  for (i in 1:nrow(df)) {
   df[i, 'Volatility'] = sd(df$Return[c(1:i)])

   # df[[i, 'Volatility']] = sd(df[[c(1:i), 'Return']])
```

```
  # print(df[i, 'Volatility'])
  # df[[i, 'Volatility']] = sd(df$Return[c(1:i)])

}

# print(df[i, 'Volatility'])

df <- df %>%
  dplyr::select(Date, Return, ESG, Volatility) %>%
  mutate(
    # Volatility = sapply(1:n(), function(i) sd(Return[1:i])),
    Profit = case_when(
      Return > 0 ~ "Gain",
      Return < 0 ~ "Loss",
      TRUE ~ "Neutral"
    ))

df$Profit <- factor(df$Profit, levels = c("Loss", "Gain", "Neutral"))

# df$Profit = df$Return
# df$Profit[which(df$Profit < 0)] <- 'Loss'
# # df$Profit[which(df$Profit <= 0)] <- 'Loss'
# df$Profit[which(df$Profit > 0)] <- 'Gain'
# df$Profit[which(df$Profit == 0)] <- 'Nothing' #Propose moi un bon nom
# df$Profit <- as.factor(df$Profit)
# print(head(df$Profit))

# print(tail(df))
```

```
    # print("ok")

    # print("Final_mesdonnées 3 D")
    # print(head(df))

    return(df)


  } else{
    return(NULL)

  }

})


output$plotly_3D <- renderPlotly({

  if(!is.null(plotly_3D_dt())){

    df = plotly_3D_dt_new()

    lg_names = yf.get_long_names(stock_to_analyze())

    the_title = paste0(lg_names, " 3D plot")

    # dans le plotly
    # print("je suis dans le plotly")
```

```
  # print(names(df))
  fig <- plot_ly(df, x = ~Return,
                 y = ~Volatility,
                 z = ~ESG,
                 color = ~Profit,
                 colors = c('#BF382A',
                            "darkgreen",
                            '#0C4B8E'))
  # "darkred",
  #         'darkgreen'
  # , '#0C4B8E' #,in case i want three colors
  # ))
  fig <- fig %>% add_markers(marker = list(size = 5))
  fig <- fig %>% layout(title = the_title,
                        scene = list(xaxis = list(title = 'Return'),
                                     yaxis = list(title = 'Volatility'),
                                     zaxis = list(title = 'ESG score')))




  return(fig)

 }else{
  return(NULL)
 }

})
```

```
# Rendement annualisé benchmarck
rf_Rend_Annuel <- reactive({
 if(!is.null(dt_rf_prices())){

  # Calculer les rendements annualisés du Benchmarck
  tq_transmute(dt_rf_prices(),
           Close,
           periodReturn,
           period = "yearly",
           col_rename = "Benchmark return")


 }else{

  return(NULL)

 }

})

R_merge_stockandrf <- reactive({
 if(!is.null(dt_rf_prices()) && !is.null(rf_Rend_Annuel())){

  # Merge stock returns with baseline
  Rstock_Rrf <- dplyr::left_join(dt_rf_prices(), rf_Rend_Annuel(), by = c("Date" = "Date"))

  # print(Rstock_Rrf)
```

```
  Rstock_Rrf


 }else{

  return(NULL)

 }

})


# Sharpe ratio annuel
Sharpe_annuel <- reactive({

 tq_performance(Rend_Annuel(),
           Ra = Return,
           performance_fun = SharpeRatio,
           # Rf = input$rfrate,
           Rf=FR()
 )

})



# output$vbox_esg_note <- renderValueBox(createVBx_esg("AAPL", ticker = ""))
```

```
 # output$vbox_esg_note <- renderValueBox({
 #   if (is.null(ESG_histo()) || stock_to_analyze() == "") {
 #     return(NULL)
 #
 #   } else {
 #     chart_hist = ESG_histo()
 #
 #     the_box = createVBx_esg(chart_hist, ticker = toupper(stock_to_analyze()))
 #
 #     return(the_box)
 #
 #     }
 #   })


  output$vbox_esg_note <- renderValueBox({
    if (is.null(ESG_histo()) || stock_to_analyze() == "") {
      if(is.null(ESG_histo())){
        # return(valueBox("Please select a stock and click 'submit'", icon = icon("exclamation-triangle"),
color = "warning"))
        # return(valueBox("Please select a stock and click 'submit'"))
        # return("Please select a stock and click 'submit'")
        # Remplacer le simple texte par un valueBox
        return(valueBox(
          value = "No Data",
          subtitle = "Please select a stock and click 'submit'",
          icon = icon("exclamation-triangle"),
          color = "warning"
        ))
```

```
    }

    if(nchar(stock_to_analyze()) == 0){
     return(valueBox("No Data", "Please select a stock", icon = icon("exclamation-triangle"), color = "warning"))

    }else{
     return(valueBox("Score history not available", "Please select another ESG", icon = icon("exclamation-triangle"), color = "warning"))
    }

  } else {
    chart_hist = ESG_histo()
    the_box = createVBx_esg(chart_hist, ticker = toupper(stock_to_analyze()))
    return(the_box)
  }
 })

 output$vbox_esg_note_spark <- renderSparkline({
  if (is.null(ESG_histo()) || stock_to_analyze() == "") {
    return(NULL)

  } else {

    esg_dt = ESG_histo()

    names(esg_dt) = c("Date", "Total", "E", "S", "G")

    # esg_dt_note = as.vector(esg_dt$Total)
```

```
    esg_dt_note <- tibble(esg_dt)

    # the_spark = sparkline::sparkline(esg_dt_note$Total, type = "line", width = "100%", height = "100%")

    the_spark = sparkline::sparkline(
      esg_dt$Total,
      type = "line",
      width = "100%",
      height = "100%",
      # tooltipFormat = '{{offset:offset}}<br>Date: {{x}}<br>Total : {{y}}',
      # tooltipChartTitle = 'ESG Scores',
      lineColor = 'blue',
      fillColor = 'lightblue',
      spotColor = NULL,
      minSpotColor = NULL,
      maxSpotColor = NULL,
      highlightSpotColor = NULL,
      highlightLineColor = NULL
    )

    return(the_spark)

  }
 })

 ### Partie calcul du sharpe ratio
 # dt_St_rf_ESG <- reactive({
```

```
#   if(!is.null(St_rf_ESG())){
#
#   }else{
#     return(NULL)
#   }
# })

sc_optimal <- reactive({
  # req(input$sc_question)
  if (input$sc_question){
    # j'ai décidé d'utiliser sc optimisé

    # lorsque je propose d'utiliser directement rf comme rf
    if(input$benchmark){

      if(!is.null(St_rf_ESG())){

        # vérifier si mes données contiennent "Index_return"
        data = St_rf_ESG()
        print("testons St_rf_ESG")
        print(tail(data))

        print("j'utilise rf com rf")
        if(nchar(input$Rf_rate) != 0 && !is.null(RF_Data_return())){
          print('les données rf sont dispo')

          names(data) = c("Date", "Ticker_return", "Index_return",  "ESG")
          opt_value = optimize(optimize_sc,
                               interval = c(0, 1),
```

```
                    returns = data$Ticker_return,
                    rf = data$Index_return,
                    esg = data$ESG,
                    type_return = input$retfreq)$minimum

    return(opt_value)
   }else{
    print('les données rf ne sont pas dispo')
    return(NULL)
   }

  } else {

   return(NULL)

  }

 } else {
  print("je propose directement rf")

  # Ajoutez cette ligne pour définir data si input$benchmark == FALSE
  if (!is.null(St_rf_ESG())) {
   data = St_rf_ESG()
   data$rf = input$rfrate_new # ligne 12394

   opt_value = optimize(optimize_sc,
                        interval = c(0, 1),
                        returns = data$Ticker_return,
                        rf = data$rf,
```

```
                    esg = data$ESG,
                    type_return = input$retfreq)$minimum

      return(opt_value)
    } else {
      print("Les données ne sont pas disponibles")
      return(NULL)
    }

  }

 } else {
  # cette souspartie est bonne
  desire_value = input$sc

  print(paste0("The selected sc is : ", desire_value))

  desire_value
 }
})




rf <- reactive({
 if (input$benchmark) {
  if(!is.null(St_rf_ESG())){
```

```
    data = St_rf_ESG()
    print('je verifie bien mes données')

    print(tail(data, 3))
    names(data) = c("Date", "Ticker_return", "Index_return", "ESG")

    data$Index_return

   }else{
    return(NULL)
   }

  } else {
   rep(input$rfrate_new,
      # input$rfrate_new / 100,
      nrow(St_rf_ESG()))
  }
})

output$Sharpe__result <- renderDT({
 # output$Sharpe__result <- renderTable({

 if(!is.null(St_rf_ESG())){

   data = St_rf_ESG()

   if(nchar(input$Rf_rate) != 0 && !is.null(RF_Data_return())){
    print("jai des données rf looo")
    sav_name = paste0(names(data)[2:3], "_return")
```

```
  names(data) = c("Date", "Ticker_return", "Index_return", "ESG")

}else{
  print("je n'ai des données rf")
  sav_name = paste0(names(data)[2], "_return")
  names(data) = c("Date", "Ticker_return", "ESG")
}

# print("-- sc_optimal --")
# print(sc_optimal()) #line 12525
# print("-- rf --")
# print(rf())
# print("-- Return frequency --")
# print(input$retfreq)



data_modified <- calculate_modified_sharpe(data, sc_optimal(), rf(), input$retfreq)

data_modified[, "Sharpe_Ratio"] <- round(data_modified[, "Sharpe_Ratio"], 4)
data_modified[, "Modified_SR"] <- round(data_modified[, "Modified_SR"], 4)

if(nchar(input$Rf_rate) != 0 && !is.null(RF_Data_return())){
  names(data_modified)[2:3] =  sav_name

}else{
  names(data_modified)[2] =  sav_name
}
```

```
    data_modified <- DT::datatable(data_modified,
                          extensions = 'Responsive',
                          rownames = FALSE, options = list(
                            columnDefs = list(list(targets = "_all", className = "dt-center",
                                        width = "25px"))
                          )
    )

    data_modified

   }else{

    return(NULL)

   }


 })









}

bs_global_theme()
```

```
shinyApp(ui, server)
```