Case Study: gtcars

Let's make a display table using the `gtcars` dataset. We all know `mtcars`... what is `gtcars`? It's basically a modernized `mtcars` for the **gt** age. It's part of the **gt** package, and here is a preview of the tibble:

```
# This is `gtcars`
dplyr::glimpse(gtcars)
#> Observations: 47
#> Variables: 15
#> $ mfr         <chr> "Ford", "Ferrari", "Ferrari", "Ferrari", "Ferrari"...
#> $ model       <chr> "GT", "458 Speciale", "458 Spider", "458 Italia", ...
#> $ year        <dbl> 2017, 2015, 2015, 2014, 2016, 2015, 2017, 2015, 20...
#> $ trim        <chr> "Base Coupe", "Base Coupe", "Base", "Base Coupe", ...
#> $ bdy_style   <chr> "coupe", "coupe", "convertible", "coupe", "coupe",...
#> $ hp          <dbl> 647, 597, 562, 562, 661, 553, 680, 652, 731, 949, ...
#> $ hp_rpm      <dbl> 6250, 9000, 9000, 9000, 8000, 7500, 8250, 8000, 82...
#> $ trq         <dbl> 550, 398, 398, 398, 561, 557, 514, 504, 509, 664, ...
#> $ trq_rpm     <dbl> 5900, 6000, 6000, 6000, 3000, 4750, 5750, 6000, 60...
#> $ mpg_c       <dbl> 11, 13, 13, 13, 15, 16, 12, 11, 11, 12, 21, 16, 11...
#> $ mpg_h       <dbl> 18, 17, 17, 17, 22, 23, 17, 16, 16, 16, 22, 22, 18...
#> $ drivetrain  <chr> "rwd", "rwd", "rwd", "rwd", "rwd", "rwd", "awd", "...
#> $ trsmn       <chr> "7a", "7a", "7a", "7a", "7a", "7a", "7a", "7a", "7...
#> $ ctry_origin <chr> "United States", "Italy", "Italy", "Italy", "Italy...
#> $ msrp        <dbl> 447000, 291744, 263553, 233509, 245400, 198973, 29...
```

For the purpose of simply learning more about **gt**, let's reduce this 47-row tibble to one that has only 8 rows:

```
# Get a subset of 8 cars from the `gtcars` dataset: two
# from each manufacturer country of origin except the UK
gtcars_8 <-
  gtcars %>%
  dplyr::group_by(ctry_origin) %>%
  dplyr::top_n(2) %>%
  dplyr::ungroup() %>%
  dplyr::filter(ctry_origin != "United Kingdom")
#> Selecting by msrp

# Show the `gtcars_8` tibble
dplyr::glimpse(gtcars_8)
#> Observations: 8
#> Variables: 15
#> $ mfr         <chr> "Ford", "Ferrari", "Acura", "Nissan", "Lamborghini...
#> $ model       <chr> "GT", "LaFerrari", "NSX", "GT-R", "Aventador", "i8...
#> $ year        <dbl> 2017, 2015, 2017, 2016, 2015, 2016, 2017, 2016
#> $ trim        <chr> "Base Coupe", "Base Coupe", "Base Coupe", "Premium...
#> $ bdy_style   <chr> "coupe", "coupe", "coupe", "coupe", "coupe", "coup...
#> $ hp          <dbl> 647, 949, 573, 545, 700, 357, 645, 503
#> $ hp_rpm      <dbl> 6250, 9000, 6500, 6400, 8250, 5800, 5000, 6250
#> $ trq         <dbl> 550, 664, 476, 436, 507, 420, 600, 479
#> $ trq_rpm     <dbl> 5900, 6750, 2000, 3200, 5500, 3700, 5000, 1750
#> $ mpg_c       <dbl> 11, 12, 21, 16, 11, 28, 12, 16
#> $ mpg_h       <dbl> 18, 16, 22, 22, 18, 29, 19, 22
#> $ drivetrain  <chr> "rwd", "rwd", "awd", "awd", "awd", "awd", "rwd", "...
```

```
#> $ trsmn       <chr> "7a", "7a", "9a", "6a", "7a", "6am", "6m", "7a"
#> $ ctry_origin <chr> "United States", "Italy", "Japan", "Japan", "Italy...
#> $ msrp        <dbl> 447000, 1416362, 156000, 101770, 397500, 140700, 9...
```

Let's make a display table from this dataset. In doing so we'll fulfill the following 10 requirements:

1. putting the cars into characteristic groups (by the car manufacturer's country of origin)
2. removing some of the columns that we don't want to present
3. incorporating some columns into a column group
4. formatting the currency data and using a monospaced font for easier reading of that data
5. giving the table a title and a subtitle
6. adding footnotes to draw attention to some of the more interesting data points and to explain some of the more unusual aspects of the data
7. placing a citation for the dataset at the bottom of the table
8. transforming the transmission (`trsmn`) codes so that they are readable and understandable
9. styling some cells according to basic criteria
10. highlighting the cars that are considered to be *grand tourers*

Row Groups

Let's again use **dplyr** to help make groupings by the `ctry_origin` column, which provides the country of origin for the vehicle manufacturer of the car. We can simply use `dplyr::group_by()` on the `gtcars` dataset and pass that to `gt()`. What you get is a display table that arranges the cars into row groups, with the name of the group displayed prominently above.

```
# Use `group_by()` on `gtcars` and pass that to `gt()`
gtcars_8 %>%
  dplyr::group_by(ctry_origin) %>%
  gt() %>%
  tab_options(
    table.width = pct(50))
```

mfr	model	year	trim	bdy_style	hp	hp_rpm	trq	trq_rpm	mpg_c	mpg_
United States										
Ford	GT	2017	Base Coupe	coupe	647	6250	550	5900	11	
Dodge	Viper	2017	GT Coupe	coupe	645	5000	600	5000	12	
Italy										
Ferrari	LaFerrari	2015	Base Coupe	coupe	949	9000	664	6750	12	
Lamborghini	Aventador	2015	LP 700-4 Coupe	coupe	700	8250	507	5500	11	
Japan										
Acura	NSX	2017	Base Coupe	coupe	573	6500	476	2000	21	
Nissan	GT-R	2016	Premium Coupe	coupe	545	6400	436	3200	16	
Germany										
BMW	i8	2016	Mega World Coupe	coupe	357	5800	420	3700	28	
Mercedes-Benz	AMG GT	2016	S Coupe	coupe	503	6250	479	1750	16	

Getting the row groups in the preferred order can be done easily with **dplyr**'s `arrange()` function. For example, we can have groups that are arranged alphabetically by manufacturer (`mfr`) and then sorted by highest sticker price (`msrp`) to lowest.

```
gtcars_8 %>%
  dplyr::group_by(ctry_origin) %>%
  dplyr::arrange(mfr, desc(msrp)) %>%
  gt()
```

mfr	model	year	trim	bdy_style	hp	hp_rpm	trq	trq_rpm	mpg_c	mpg_
Japan										
Acura	NSX	2017	Base Coupe	coupe	573	6500	476	2000	21	
Nissan	GT-R	2016	Premium Coupe	coupe	545	6400	436	3200	16	
Germany										
BMW	i8	2016	Mega World Coupe	coupe	357	5800	420	3700	28	
Mercedes-Benz	AMG GT	2016	S Coupe	coupe	503	6250	479	1750	16	
United States										
Dodge	Viper	2017	GT Coupe	coupe	645	5000	600	5000	12	
Ford	GT	2017	Base Coupe	coupe	647	6250	550	5900	11	
Italy										
Ferrari	LaFerrari	2015	Base Coupe	coupe	949	9000	664	6750	12	
Lamborghini	Aventador	2015	LP 700-4 Coupe	coupe	700	8250	507	5500	11	

We could also use factor levels to get a more particular ordering within `arrange()`. For example, we can first arrange the groups themselves (the country of origin–`ctry_origin`) by our own preferred ordering and then arrange by `mfr` and descending `msrp` as before. Then, `group_by(ctry_origin)` can be used on the sorted tibble before passing this to `gt()`.

```
# Define our preferred order `ctry_origin`
order_countries <- c("Germany", "Italy", "United States", "Japan")

# Reorder the table rows by our specific ordering of groups
gtcars_8 %>%
  dplyr::arrange(
    factor(ctry_origin, levels = order_countries), mfr, desc(msrp)
  ) %>%
  dplyr::group_by(ctry_origin) %>%
  gt()
```

mfr	model	year	trim	bdy_style	hp	hp_rpm	trq	trq_rpm	mpg_c	mpg_
Germany										
BMW	i8	2016	Mega World Coupe	coupe	357	5800	420	3700	28	
Mercedes-Benz	AMG GT	2016	S Coupe	coupe	503	6250	479	1750	16	
Italy										
Ferrari	LaFerrari	2015	Base Coupe	coupe	949	9000	664	6750	12	
Lamborghini	Aventador	2015	LP 700-4 Coupe	coupe	700	8250	507	5500	11	
United States										
Dodge	Viper	2017	GT Coupe	coupe	645	5000	600	5000	12	
Ford	GT	2017	Base Coupe	coupe	647	6250	550	5900	11	
Japan										

Acura	NSX	2017	Base Coupe	coupe	573	6500	476	2000	21
Nissan	GT-R	2016	Premium Coupe	coupe	545	6400	436	3200	16

The last variation is to combine the manufacturer name with the model name, using those combined strings as row labels for the table. This is just a little more **dplyr** where we can use `dplyr::mutate()` to make a new `car` column followed by `dplyr::select()` where we remove the `mfr` and `model` columns. When introducing the tibble to the `gt()` function, we can now use the `rowname_col` argument to specify a column that will serve as row labels (which is the newly made `car` column).

```
# Reorder the table rows by our specific ordering of groups
tab <-
  gtcars_8 %>%
  dplyr::arrange(
    factor(ctry_origin, levels = order_countries),
    mfr, desc(msrp)
    ) %>%
  dplyr::mutate(car = paste(mfr, model)) %>%
  dplyr::select(-mfr, -model) %>%
  dplyr::group_by(ctry_origin) %>%
  gt(rowname_col = "car")

# Show the table
tab
```

	year	trim	bdy_style	hp	hp_rpm	trq	trq_rpm	mpg_c	mpg_h
Germany									
BMW i8	2016	Mega World Coupe	coupe	357	5800	420	3700	28	29
Mercedes-Benz AMG GT	2016	S Coupe	coupe	503	6250	479	1750	16	22
Italy									
Ferrari LaFerrari	2015	Base Coupe	coupe	949	9000	664	6750	12	16
Lamborghini Aventador	2015	LP 700-4 Coupe	coupe	700	8250	507	5500	11	18
United States									
Dodge Viper	2017	GT Coupe	coupe	645	5000	600	5000	12	19
Ford GT	2017	Base Coupe	coupe	647	6250	550	5900	11	18
Japan									
Acura NSX	2017	Base Coupe	coupe	573	6500	476	2000	21	22
Nissan GT-R	2016	Premium Coupe	coupe	545	6400	436	3200	16	22

Hiding and Moving Some Columns

Let's hide two columns that we don't need to the final table: `drivetrain` and `bdy_style`. We can use the `cols_hide()` function to hide columns. The same end result might also have been achieved by using `gtcars %>% dplyr::select(-c(drivetrain, bdy_style))`, before introducing the table to `gt()`. Why this function then? Sometimes you'll need variables for conditional statements within **gt** but won't want to display them in the end.

Aside from hiding columns, let's *move* some of them. Again, this could be done with `dplyr::select()` but there are options here in **gt** via the `cols_move_to_start()`, `cols_move()`, and `cols_move_to_end()` functions.

```
# Use a few `cols_*()` functions to hide and move columns
tab <-
  tab %>%
  cols_hide(columns = vars(drivetrain, bdy_style)) %>%
  cols_move(
    columns = vars(trsmn, mpg_c, mpg_h),
    after = vars(trim)
  )

# Show the table
tab
```

	year	trim	trsmn	mpg_c	mpg_h	hp	hp_rpm	trq	trq_rpm	
Germany										
BMW i8	2016	Mega World Coupe	6am	28	29	357	5800	420	3700	
Mercedes-Benz AMG GT	2016	S Coupe	7a	16	22	503	6250	479	1750	
Italy										
Ferrari LaFerrari	2015	Base Coupe	7a	12	16	949	9000	664	6750	1
Lamborghini Aventador	2015	LP 700-4 Coupe	7a	11	18	700	8250	507	5500	
United States										
Dodge Viper	2017	GT Coupe	6m	12	19	645	5000	600	5000	
Ford GT	2017	Base Coupe	7a	11	18	647	6250	550	5900	
Japan										
Acura NSX	2017	Base Coupe	9a	21	22	573	6500	476	2000	
Nissan GT-R	2016	Premium Coupe	6a	16	22	545	6400	436	3200	

Putting Columns Into Groups

It's sometimes useful to arrange variables/columns into groups by using spanner column labels. This can be done in **gt** by using the `tab_spanner()` function. It takes the `label` and `columns` arguments; `label` is the spanner column label and the `columns` are those columns that belong in this group.

Here, we'll put the `mpg_c`, `mpg_h`, `hp`, `hp_rpm`, `trq`, `trq_rpm` columns under the `Performance` spanner column, and the remaining columns won't be grouped together. This single spanner column label is styled with Markdown by using the `md()` helper.

```
# Put the first three columns under a spanner
# column with the label 'Performance'
tab <-
  tab %>%
  tab_spanner(
    label = md("*Performance*"),
    columns = vars(mpg_c, mpg_h, hp, hp_rpm, trq, trq_rpm)
  )

# Show the table
tab
#> Warning in spanners[is.na(spanners)] <- headings[is.na(spanners)]: number
#> of items to replace is not a multiple of replacement length
```

				Performance						
	year	trim	trsmn	mpg_c	mpg_h	hp	hp_rpm	trq	trq_rpm	
Germany										
BMW i8	2016	Mega World Coupe	6am	28	29	357	5800	420	3700	
Mercedes-Benz AMG GT	2016	S Coupe	7a	16	22	503	6250	479	1750	
Italy										
Ferrari LaFerrari	2015	Base Coupe	7a	12	16	949	9000	664	6750	1
Lamborghini Aventador	2015	LP 700-4 Coupe	7a	11	18	700	8250	507	5500	
United States										
Dodge Viper	2017	GT Coupe	6m	12	19	645	5000	600	5000	
Ford GT	2017	Base Coupe	7a	11	18	647	6250	550	5900	
Japan										
Acura NSX	2017	Base Coupe	9a	21	22	573	6500	476	2000	
Nissan GT-R	2016	Premium Coupe	6a	16	22	545	6400	436	3200	

Merging Columns Together and Labeling Them

Sometimes we'd like to combine the data from two columns into a single column. The `cols_merge()` function allows us to do this, we just need to describe how the data should be combined. For our table, let's merge together the following pairs of columns:

- `mpg_c` and `mpg_h` (miles per gallon in city and highway driving modes)
- `hp` and `hp_rpm` (horsepower and associated RPM)
- `trq` and `trq_rpm` (torque and associated RPM)

The `cols_merge()` function uses a `col_1` column and a `col_2` column. Once combined, the `col_1` column will be retained and the `col_2` column will be dropped. The pattern argument uses `{1}` and `{2}` to represent the content of `col_1` and `col_2`. Here, we can use string literals to add text like `rpm` or the `@` sign. Furthermore, because we are targeting an HTML table, we can use the `<br>` tag to insert a linebreak.

Labeling columns essentially means that we are choosing display-friendly labels that are no longer simply the column names (the default label). The `cols_label()` function makes this relabeling possible. It accepts a series of named arguments in the form of `<column_name> = <column_label>, ...`.

```
# Perform three column merges to better present
# MPG, HP, and torque; relabel all the remaining
# columns for a nicer-looking presentation
tab <-
  tab %>%
  cols_merge(
    col_1 = vars(mpg_c),
    col_2 = vars(mpg_h),
    pattern = "{1}c<br>{2}h"
    ) %>%
  cols_merge(
    col_1 = vars(hp),
    col_2 = vars(hp_rpm),
    pattern = "{1}<br>@{2}rpm"
  ) %>%
  cols_merge(
    col_1 = vars(trq),
```

```
    col_2 = vars(trq_rpm),
    pattern = "{1}<br>@{2}rpm"
  ) %>%
  cols_label(
    mpg_c = "MPG",
    hp = "HP",
    trq = "Torque",
    year = "Year",
    trim = "Trim",
    trsmn = "Transmission",
    msrp = "MSRP"
  )

# Show the table
tab
#> Warning in spanners[is.na(spanners)] <- headings[is.na(spanners)]: number
#> of items to replace is not a multiple of replacement length
```

				Performance		
	Year	Trim	Transmission	MPG	HP	
Germany						
BMW i8	2016	Mega World Coupe	6am	28c 29h	357 @5800rpm	420 @3
Mercedes-Benz AMG GT	2016	S Coupe	7a	16c 22h	503 @6250rpm	479 @1
Italy						
Ferrari LaFerrari	2015	Base Coupe	7a	12c 16h	949 @9000rpm	664 @6
Lamborghini Aventador	2015	LP 700-4 Coupe	7a	11c 18h	700 @8250rpm	507 @5
United States						
Dodge Viper	2017	GT Coupe	6m	12c 19h	645 @5000rpm	600 @5
Ford GT	2017	Base Coupe	7a	11c 18h	647 @6250rpm	550 @5
Japan						
Acura NSX	2017	Base Coupe	9a	21c 22h	573 @6500rpm	476 @2
Nissan GT-R	2016	Premium Coupe	6a	16c 22h	545 @6400rpm	436 @3

Using Formatter Functions

There are a number of formatter functions, all with the general naming convention `fmt*()`. The various formatters are convenient for applying formats to numeric or character values in the table's field. Here, we will simply use `fmt_currency()` on the `msrp` column (we still refer to columns by their original names) to get USD currency will no decimal places. We're not supplying anything for the `rows` argument and this means we want to apply the formatting to the entire column of data.

```
# Format the `msrp` column to USD currency
# with no display of the currency subunits
tab <-
  tab %>%
  fmt_currency(
    columns = vars(msrp),
    currency = "USD",
    decimals = 0
```

```
  )

# Show the table
tab
#> Warning in spanners[is.na(spanners)] <- headings[is.na(spanners)]: number
#> of items to replace is not a multiple of replacement length
```

				Performance		
	Year	Trim	Transmission	MPG	HP	
Germany						
BMW i8	2016	Mega World Coupe	6am	28c 29h	357 @5800rpm	420 @5
Mercedes-Benz AMG GT	2016	S Coupe	7a	16c 22h	503 @6250rpm	479 @1
Italy						
Ferrari LaFerrari	2015	Base Coupe	7a	12c 16h	949 @9000rpm	664 @6
Lamborghini Aventador	2015	LP 700-4 Coupe	7a	11c 18h	700 @8250rpm	507 @5
United States						
Dodge Viper	2017	GT Coupe	6m	12c 19h	645 @5000rpm	600 @5
Ford GT	2017	Base Coupe	7a	11c 18h	647 @6250rpm	550 @5
Japan						
Acura NSX	2017	Base Coupe	9a	21c 22h	573 @6500rpm	476 @2
Nissan GT-R	2016	Premium Coupe	6a	16c 22h	545 @6400rpm	436 @3

Column Alignment and Style Changes

We can change the alignment of data in columns with `cols_align()`. For our table, let's center-align the `mpg_c`, `hp`, and `trq` columns. All other columns will maintain their default alignments.

It's sometimes useful to modify the default styles of table cells. We can do this in a targeted way with the `tab_style()` function. That function require two key pieces of information: a `style` definition, and one or more `locations` (which cells should the styles be applied to?). The `style` argument commonly uses the `cells_styles()` helper function, which contains arguments for all the styles that are supported (use `?cells_styles` for more information on this). Here we will use a text size of `12px` in our targeted cells—both `px(12)` and `"12px"` work equally well here. We also use helper functions with the `locations` argument and these are the `cells_*()` functions. We would like to target the data cells in all columns except `year` and `msrp` so we need to use `cells_data` and then supply our target columns to the `columns` argument.

```
# Center-align three columns in the gt table
# and modify the text size of a few columns
# of data
tab <-
  tab %>%
  cols_align(
    align = "center",
    columns = vars(mpg_c, hp, trq)
  ) %>%
  tab_style(
    style = cells_styles(
      text_size = px(12)),
    locations = cells_data(
```

```
      columns = vars(trim, trsmn, mpg_c, hp, trq))
  )

# Show the table
tab
#> Warning in spanners[is.na(spanners)] <- headings[is.na(spanners)]: number
#> of items to replace is not a multiple of replacement length
```

				Performance		
	Year	Trim	Transmission	MPG	HP	Torqu
Germany						
BMW i8	2016	Mega World Coupe	6am	28c 29h	357 @5800rpm	420 @3
Mercedes-Benz AMG GT	2016	S Coupe	7a	16c 22h	503 @6250rpm	479 @1
Italy						
Ferrari LaFerrari	2015	Base Coupe	7a	12c 16h	949 @9000rpm	664 @6
Lamborghini Aventador	2015	LP 700-4 Coupe	7a	11c 18h	700 @8250rpm	507 @5
United States						
Dodge Viper	2017	GT Coupe	6m	12c 19h	645 @5000rpm	600 @5
Ford GT	2017	Base Coupe	7a	11c 18h	647 @6250rpm	550 @5
Japan						
Acura NSX	2017	Base Coupe	9a	21c 22h	573 @6500rpm	476 @2
Nissan GT-R	2016	Premium Coupe	6a	16c 22h	545 @6400rpm	436 @3

Text Transforms

A text transform via the `text_transform()` function is a great way to further manipulate text in data cells (even after they've been formatted with the `fmt*()` function). After targeting data cells with the `cells_data()` location helper function, we supply a function to the `fn` argument that processes a vector of text. If we intend to render as an HTML table, we can directly apply HTML tags in the transformation function. The function we provide here will build strings that read better in a display table.

```
# Transform the column of text in `trsmn` using
# a custom function within `text_transform()`;
# here `x` represents a character vector defined
# in the `cells_data()` function
tab <-
  tab %>%
  text_transform(
    locations = cells_data(columns = vars(trsmn)),
    fn = function(x) {

      # The first character of `x` always
      # indicates the number of transmission speeds
      speed <- substr(x, 1, 1)

      # We can carefully determine which transmission
      # type we have in `x` with a `dplyr::case_when()`
      # statement
      type <-
```

```
        dplyr::case_when(
          substr(x, 2, 3) == "am" ~ "Automatic/Manual",
          substr(x, 2, 2) == "m" ~ "Manual",
          substr(x, 2, 2) == "a" ~ "Automatic",
          substr(x, 2, 3) == "dd" ~ "Direct Drive"
        )

      # Let's paste together the `speed` and `type`
      # vectors to create HTML text replacing `x`
      paste(speed, " Speed<br><em>", type, "</em>")
    }
  )

# Show the table
tab
#> Warning in spanners[is.na(spanners)] <- headings[is.na(spanners)]: number
#> of items to replace is not a multiple of replacement length
```

	Year	Trim	Transmission	MPG
Germany				
BMW i8	2016	Mega World Coupe	6 Speed Automatic/Manual	28c 29h
Mercedes-Benz AMG GT	2016	S Coupe	7 Speed Automatic	16c 22h
Italy				
Ferrari LaFerrari	2015	Base Coupe	7 Speed Automatic	12c 16h
Lamborghini Aventador	2015	LP 700-4 Coupe	7 Speed Automatic	11c 18h
United States				
Dodge Viper	2017	GT Coupe	6 Speed Manual	12c 19h
Ford GT	2017	Base Coupe	7 Speed Automatic	11c 18h
Japan				
Acura NSX	2017	Base Coupe	9 Speed Automatic	21c 22h
Nissan GT-R	2016	Premium Coupe	6 Speed Automatic	16c 22h

Table Header: Title and Subtitle

The `tab_header()` function allows us to place a table title and, optionally, a subtitle at the top of the display table. It's generally a good idea to have both in a table, where the subtitle provides additional information (though that isn't quite the case in our example below).

```
# Add a table title and subtitle; we can use
# markdown with the `md()` helper function
tab <-
  tab %>%
  tab_header(
    title = md("The Cars of **gtcars**"),
    subtitle = "These are some fine automobiles"
  )

# Show the table
```

```
tab
#> Warning in spanners[is.na(spanners)] <- headings[is.na(spanners)]: number
#> of items to replace is not a multiple of replacement length
```

The Cars of **gtcars**
These are some fine automobiles

	Year	Trim	Transmission	MPG
Germany				
BMW i8	2016	Mega World Coupe	6 Speed Automatic/Manual	28c 29h
Mercedes-Benz AMG GT	2016	S Coupe	7 Speed Automatic	16c 22h
Italy				
Ferrari LaFerrari	2015	Base Coupe	7 Speed Automatic	12c 16h
Lamborghini Aventador	2015	LP 700-4 Coupe	7 Speed Automatic	11c 18h
United States				
Dodge Viper	2017	GT Coupe	6 Speed Manual	12c 19h
Ford GT	2017	Base Coupe	7 Speed Automatic	11c 18h
Japan				
Acura NSX	2017	Base Coupe	9 Speed Automatic	21c 22h
Nissan GT-R	2016	Premium Coupe	6 Speed Automatic	16c 22h

Adding a Source Citation

A *source note* can be added below the display table using the `tab_source_note()` function. We can even add multiple source notes with multiple calls of that function. Here, we supply a web URL and by using Markdown (with `md()`) it's easy to create a link to the source of the data.

```
# Add a source note to the bottom of the table; this
# appears below the footnotes
tab <-
  tab %>%
  tab_source_note(
    source_note = md(
      "Source: Various pages within [edmunds.com](https://www.edmunds.com).")
  )

# Show the table
tab
#> Warning in spanners[is.na(spanners)] <- headings[is.na(spanners)]: number
#> of items to replace is not a multiple of replacement length
```

The Cars of **gtcars**
These are some fine automobiles

	Year	Trim	Transmission	MPG
Germany				
BMW i8	2016	Mega World Coupe	6 Speed Automatic/Manual	28c 29h
Mercedes-Benz AMG GT	2016	S Coupe	7 Speed Automatic	16c 22h

Italy				
Ferrari LaFerrari	2015	Base Coupe	7 Speed Automatic	12c 16h
Lamborghini Aventador	2015	LP 700-4 Coupe	7 Speed Automatic	11c 18h
United States				
Dodge Viper	2017	GT Coupe	6 Speed Manual	12c 19h
Ford GT	2017	Base Coupe	7 Speed Automatic	11c 18h
Japan				
Acura NSX	2017	Base Coupe	9 Speed Automatic	21c 22h
Nissan GT-R	2016	Premium Coupe	6 Speed Automatic	16c 22h

Source: Various pages within edmunds.com.

Using the Complete `gtcars` table and Adding Footnotes

Let's bring it all together by putting together all the statements we developed for `gtcars_8`, and applying that to the complete `gtcars` dataset. At the same time, we'll add a few interesting footnotes and our specific requirements for footnoting are:

```
a. identifying the car with the best gas mileage (city)
b. identifying the car with the highest horsepower
c. stating the currency of the MSRP
```

The `tab_footnote()` function expects note text for the `footnote` argument, and locations for where the glyph should be attached. It will handle the placement of the note glyph and also place the footnote in the footnotes area. Here, we'll use the `cells_data()` *location helper* function. There are several location helper functions for targeting all parts of the table (e.g,. `cells_data()`, `cells_stub()`, etc.). Each *location helper* has their own interface for targeting cells, help is available at `?gt::location_cells`.

What `cells_data()` expects is `columns` (column names, which can be conveniently provided in `vars()`) and `rows` (which can be a vector of row names or row indices). The `cells_stub()` location helper only expects a vector of `rows`. For `cells_column_labels()`, we can either provided targeted column labels in the `columns` argument or spanner column labels in the `groups` argument. Here, we are targeting a footnote to the `msrp` column label so we will use `columns = vars(msrp)`.

In terms of structuring the code, we're taking all the previous statements and putting those in first. It should be noted that the order of the statements does not matter to the end result, we could also put in all of the `tab_footnote()` statements first (again, any in order) and expect the same output table.

```
# Use dplyr functions to get the car with the best city gas mileage;
# this will be used to target the correct cell for a footnote
best_gas_mileage_city <-
  gtcars %>%
  dplyr::arrange(desc(mpg_c)) %>%
  dplyr::slice(1) %>%
  dplyr::mutate(car = paste(mfr, model)) %>%
  dplyr::pull(car)

# Use dplyr functions to get the car with the highest horsepower
# this will be used to target the correct cell for a footnote
highest_horsepower <-
  gtcars %>%
  dplyr::arrange(desc(hp)) %>%
```

```
  dplyr::slice(1) %>%
  dplyr::mutate(car = paste(mfr, model)) %>%
  dplyr::pull(car)

# Create a display table with `gtcars`, using all of the previous
# statements piped together + additional `tab_footnote()` stmts
tab <-
  gtcars %>%
  dplyr::arrange(
    factor(ctry_origin, levels = order_countries),
    mfr, desc(msrp)
  ) %>%
  dplyr::mutate(car = paste(mfr, model)) %>%
  dplyr::select(-mfr, -model) %>%
  dplyr::group_by(ctry_origin) %>%
  gt(rowname_col = "car") %>%
  cols_hide(columns = vars(drivetrain, bdy_style)) %>%
  cols_move(
    columns = vars(trsmn, mpg_c, mpg_h),
    after = vars(trim)
  ) %>%
  tab_spanner(
    label = md("*Performance*"),
    columns = vars(mpg_c, mpg_h, hp, hp_rpm, trq, trq_rpm)
  ) %>%
  cols_merge(
    col_1 = vars(mpg_c),
    col_2 = vars(mpg_h),
    pattern = "{1}c<br>{2}h"
  ) %>%
  cols_merge(
    col_1 = vars(hp),
    col_2 = vars(hp_rpm),
    pattern = "{1}<br>@{2}rpm"
  ) %>%
  cols_merge(
    col_1 = vars(trq),
    col_2 = vars(trq_rpm),
    pattern = "{1}<br>@{2}rpm"
  ) %>%
  cols_label(
    mpg_c = "MPG",
    hp = "HP",
    trq = "Torque",
    year = "Year",
    trim = "Trim",
    trsmn = "Transmission",
    msrp = "MSRP"
  ) %>%
  fmt_currency(
    columns = vars(msrp),
    currency = "USD",
    decimals = 0
```

```
  ) %>%
  cols_align(
    align = "center",
    columns = vars(mpg_c, hp, trq)
  ) %>%
  tab_style(
    style = cells_styles(text_size = px(12)),
    locations = cells_data(columns = vars(trim, trsmn, mpg_c, hp, trq))
  ) %>%
  text_transform(
    locations = cells_data(columns = vars(trsmn)),
    fn = function(x) {

      speed <- substr(x, 1, 1)

      type <-
        dplyr::case_when(
          substr(x, 2, 3) == "am" ~ "Automatic/Manual",
          substr(x, 2, 2) == "m" ~ "Manual",
          substr(x, 2, 2) == "a" ~ "Automatic",
          substr(x, 2, 3) == "dd" ~ "Direct Drive"
        )

      paste(speed, " Speed<br><em>", type, "</em>")
    }
  ) %>%
  tab_header(
    title = md("The Cars of **gtcars**"),
    subtitle = "These are some fine automobiles"
  ) %>%
  tab_source_note(
    source_note = md(
      "Source: Various pages within [edmunds.com](https://www.edmunds.com).")
  ) %>%
  tab_footnote(
    footnote = md("Best gas mileage (city) of all the **gtcars**."),
    locations = cells_data(
      columns = vars(mpg_c),
      rows = best_gas_mileage_city)
  ) %>%
  tab_footnote(
    footnote = md("The highest horsepower of all the **gtcars**."),
    locations = cells_data(
      columns = vars(hp),
      rows = highest_horsepower)
  ) %>%
  tab_footnote(
    footnote = "All prices in U.S. dollars (USD).",
    locations = cells_column_labels(
      columns = vars(msrp))
  )

# Show the table
```

```
tab
#> Warning in spanners[is.na(spanners)] <- headings[is.na(spanners)]: number
#> of items to replace is not a multiple of replacement length
#> Warning: HTML tags found, and they will be removed.
#>  * set `options(gt.html_tag_check = FALSE)` to disable this check

#> Warning: HTML tags found, and they will be removed.
#>  * set `options(gt.html_tag_check = FALSE)` to disable this check
```

The Cars of **gtcars**

These are some fine automobiles

	Year	Trim	Transmission	
Germany				
Audi R8	2015	4.2 (Manual) Coupe	6 Speed Manual	11c
Audi S8	2016	Base Sedan	8 Speed Automatic/Manual	15c
Audi RS 7	2016	Quattro Hatchback	8 Speed Automatic/Manual	15c
Audi S7	2016	Prestige quattro Hatchback	7 Speed Automatic	17c
Audi S6	2016	Premium Plus quattro Sedan	7 Speed Automatic	18c
BMW i8	2016	Mega World Coupe	6 Speed Automatic/Manual	28c<
BMW M6	2016	Base Coupe	7 Speed Automatic	15c
BMW M5	2016	Base Sedan	7 Speed Automatic/Manual	15c
BMW 6-Series	2016	640 I Coupe	8 Speed Automatic/Manual	20c
BMW M4	2016	Base Coupe	6 Speed Manual	17c
Mercedes-Benz AMG GT	2016	S Coupe	7 Speed Automatic	16c
Mercedes-Benz SL-Class	2016	SL400 Convertible	7 Speed Automatic/Manual	20c
Porsche 911	2016	Carrera Coupe	7 Speed Manual	20c
Porsche Panamera	2016	Base Sedan	7 Speed Automatic	18c
Porsche 718 Boxster	2017	Base Convertible	6 Speed Manual	21c
Porsche 718 Cayman	2017	Base Coupe	6 Speed Manual	20c
Italy				
Ferrari LaFerrari	2015	Base Coupe	7 Speed Automatic	12c
Ferrari F12Berlinetta	2015	Base Coupe	7 Speed Automatic	11c
Ferrari GTC4Lusso	2017	Base Coupe	7 Speed Automatic	12c
Ferrari FF	2015	Base Coupe	7 Speed Automatic	11c
Ferrari 458 Speciale	2015	Base Coupe	7 Speed Automatic	13c
Ferrari 458 Spider	2015	Base	7 Speed Automatic	13c
Ferrari 488 GTB	2016	Base Coupe	7 Speed Automatic	15c
Ferrari 458 Italia	2014	Base Coupe	7 Speed Automatic	13c
Ferrari California	2015	Base Convertible	7 Speed Automatic	16c
Lamborghini Aventador	2015	LP 700-4 Coupe	7 Speed Automatic	11c
Lamborghini Huracan	2015	LP 610-4 Coupe	7 Speed Automatic	16c
Lamborghini Gallardo	2014	LP 550-2 Coupe	6 Speed Automatic	12c
Maserati Granturismo	2016	Sport Coupe	6 Speed Automatic/Manual	13c
Maserati Quattroporte	2016	S Sedan	8 Speed Automatic/Manual	16c
Maserati Ghibli	2016	Base Sedan	8 Speed Automatic/Manual	17c
United States				
Chevrolet Corvette	2016	Z06 Coupe	7 Speed Manual	15c
Dodge Viper	2017	GT Coupe	6 Speed Manual	12c
Ford GT	2017	Base Coupe	7 Speed Automatic	11c
Tesla Model S	2017	75D	1 Speed Direct Drive	

Japan				
Acura NSX	2017	Base Coupe	9 Speed Automatic	21c
Nissan GT-R	2016	Premium Coupe	6 Speed Automatic	16c
United Kingdom				
Aston Martin Vanquish	2016	Base Coupe	8 Speed Automatic/Manual	13c
Aston Martin DB11	2017	Base Coupe	8 Speed Automatic/Manual	15c
Aston Martin Rapide S	2016	Base Sedan	8 Speed Automatic/Manual	14c
Aston Martin Vantage	2016	V8 GT (Manual) Coupe	6 Speed Manual	13c
Bentley Continental GT	2016	V8 Coupe	8 Speed Automatic/Manual	15c
Jaguar F-Type	2016	Base (Manual) Coupe	6 Speed Manual	16c
Lotus Evora	2017	2+2 Coupe	6 Speed Manual	16c
McLaren 570	2016	Base Coupe	7 Speed Automatic	16c
Rolls-Royce Dawn	2016	Base Convertible	8 Speed Automatic	12c
Rolls-Royce Wraith	2016	Base Coupe	8 Speed Automatic	13c

[1]All prices in U.S. dollars (USD).
[2]Best gas mileage (city) of all the gtcars.
[3]The highest horsepower of all the gtcars.

Source: Various pages within edmunds.com.

That is it. The final table looks pretty good and conveys the additional information we planned for. That table can be used in a lot of different places like R Markdown, Shiny, email messages... wherever HTML is accepted.