Create/Modify Parts

The *Create/Modify Parts* Family of Functions

A **gt** table can contain a few useful parts for conveying additional information. These include a header (with a titles and subtitle), a footer (with footnotes and source notes), and additional areas for labels (row group labels, column spanner labels, the stubhead label). We can modify the look of table parts more generally with `tab_options()` and perform styling on targeted table locations with `tab_style()`.

`tab_header()`: Add a table header

We can add a table header to the gt table with a title and even a subtitle. A table header is an optional table part that is positioned above the column labels. We have the flexibility to use Markdown formatting for the header's title and subtitle. Furthermore, if the table is intended for HTML output, we can use HTML in either of the title or subtitle.

EXAMPLE

Use `gtcars` to create a **gt** table; add a header part to contain a `title` and `subtitle`.

```
gtcars %>%
  dplyr::select(mfr, model, msrp) %>%
  dplyr::slice(1:5) %>%
  gt() %>%
  tab_header(
    title = md("Data listing from **gtcars**"),
    subtitle = md("`gtcars` is an R dataset")
  )
```

| Data listing from **gtcars** | | |
| `gtcars` is an R dataset | | |
mfr	model	msrp
Ford	GT	447000
Ferrari	458 Speciale	291744
Ferrari	458 Spider	263553
Ferrari	458 Italia	233509
Ferrari	488 GTB	245400

`tab_spanner()`: Add a spanner column label

Set a spanner column label by mapping it to columns already in the table. This label is placed above one or more column labels, spanning the width of those columns and column labels.

EXAMPLE

Use `gtcars` to create a **gt** table; Group several columns related to car performance under a spanner column with the label `performance` .

```
gtcars %>%
  dplyr::select(
    -mfr, -trim, bdy_style, drivetrain,
    -drivetrain, -trsmn, -ctry_origin
  ) %>%
  dplyr::slice(1:8) %>%
  gt(rowname_col = "model") %>%
  tab_spanner(
    label = "performance",
    columns = vars(
      hp, hp_rpm, trq, trq_rpm,
      mpg_c, mpg_h)
  )
```

	year	bdy_style	performance						msrp
			hp	hp_rpm	trq	trq_rpm	mpg_c	mpg_h	
GT	2017	coupe	647	6250	550	5900	11	18	447000
458 Speciale	2015	coupe	597	9000	398	6000	13	17	291744
458 Spider	2015	convertible	562	9000	398	6000	13	17	263553
458 Italia	2014	coupe	562	9000	398	6000	13	17	233509
488 GTB	2016	coupe	661	8000	561	3000	15	22	245400
California	2015	convertible	553	7500	557	4750	16	23	198973
GTC4Lusso	2017	coupe	680	8250	514	5750	12	17	298000
FF	2015	coupe	652	8000	504	6000	11	16	295000

`tab_spanner_delim()` : Create group names and column labels via delimited names

This function will split selected delimited column names such that the first components (LHS) are promoted to being spanner column labels, and the secondary components (RHS) will become the column labels. Please note that reference to individual columns must continue to be the column names from the input table data (which are unique by necessity).

EXAMPLE

Use `iris` to create a gt table; split any columns that are dot-separated between column spanner labels (first part) and column labels (second part).

```
iris %>%
  dplyr::group_by(Species) %>%
  dplyr::slice(1:4) %>%
  gt() %>%
  tab_spanner_delim(delim = ".")
```

Sepal		Petal	
Length	Width	Length	Width
setosa			
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
versicolor			
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
virginica			
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8

`tab_row_group()` : Add a row group

Create a row group with a collection of rows. This requires specification of the rows to be included, either by supplying row labels, row indices, or through use of a select helper function like `starts_with()` .

EXAMPLES

Use `gtcars` to create a **gt** table and add two row groups with the labels: `numbered` and `NA` (a group without a title, or, the rest).

```
gtcars %>%
  dplyr::select(model, year, hp, trq) %>%
  dplyr::slice(1:8) %>%
  gt(rowname_col = "model") %>%
  tab_row_group(
    group = "numbered",
    rows = matches("^[0-9]")
  )
```

	year	hp	trq
numbered			
458 Speciale	2015	597	398
458 Spider	2015	562	398
458 Italia	2014	562	398
488 GTB	2016	661	561
GT	2017	647	550
California	2015	553	557
GTC4Lusso	2017	680	514
FF	2015	652	504

Use `gtcars` to create a **gt** table. Add two row groups with the labels `powerful` and `super powerful`: the distinction being `hp` lesser or greater than `600`.

```
gtcars %>%
  dplyr::select(model, year, hp, trq) %>%
  dplyr::slice(1:8) %>%
  gt(rowname_col = "model") %>%
  tab_row_group(
    group = "powerful",
    rows = hp <= 600
  ) %>%
  tab_row_group(
    group = "super powerful",
    rows = hp > 600
  )
```

```
## Warning in min(rows_matched): no non-missing arguments to min; returning Inf
```

```
## Warning in max(rows_matched): no non-missing arguments to max; returning -Inf
```

	year	hp	trq

	year	hp	trq
super powerful			
GT	2017	647	550
488 GTB	2016	661	561
GTC4Lusso	2017	680	514
FF	2015	652	504
powerful			
458 Speciale	2015	597	398
458 Spider	2015	562	398
458 Italia	2014	562	398
California	2015	553	557

`tab_stubhead()` : Add label text to the stubhead

Add a label to the stubhead of a **gt** table. The stubhead is the lone element that is positioned left of the column labels, and above the stub. If a stub does not exist, then there is no stubhead (so no change will be made when using this function in that case). We have the flexibility to use Markdown formatting for the stubhead label. Furthermore, if the table is intended for HTML output, we can use HTML for the stubhead label.

EXAMPLE

Use `gtcars` to create a **gt** table. Add a stubhead label to describe what is in the stub.

```
gtcars %>%
  dplyr::select(model, year, hp, trq) %>%
  dplyr::slice(1:5) %>%
  gt(rowname_col = "model") %>%
  tab_stubhead(label = "car")
```

car	year	hp	trq
GT	2017	647	550
458 Speciale	2015	597	398
458 Spider	2015	562	398
458 Italia	2014	562	398
488 GTB	2016	661	561

`tab_footnote()` : Add a table footnote

The `tab_footnote()` function can make it a painless process to add a footnote to a **gt** table. There are two components to a footnote: (1) a footnote mark that is attached to the targeted cell text, and (2) the footnote text (that starts with the corresponding footnote mark) that is placed in the table's footer area. Each call of `tab_footnote()` will add a different note, and one or more cells can be targeted via the location helper functions (e.g., `cells_body()` , `cells_column_labels()` , etc.).

EXAMPLE

Use `sza` to create a **gt** table; color the `sza` column using the `data_color()` function, then, add a footnote to the `sza` column label explaining what the color scale signifies.

```
sza %>%
  dplyr::filter(
    latitude == 20 &
      month == "jan" &
      !is.na(sza)
  ) %>%
  dplyr::select(-latitude, -month) %>%
  gt() %>%
  data_color(
    columns = vars(sza),
    colors = scales::col_numeric(
      palette = c("white", "yellow", "navyblue"),
      domain = c(0, 90))
  ) %>%
  tab_footnote(
    footnote = "Color indicates height of sun.",
    locations = cells_column_labels(
      columns = vars(sza))
  )
```

tst	sza[1]
0700	84.9
0730	78.7
0800	72.7
0830	66.1
0900	61.5
0930	56.5
1000	52.1
1030	48.3
1100	45.5

[1] Color indicates height of sun.

tst	sza[1]
1130	43.6
1200	43.0

[1] Color indicates height of sun.

`tab_source_note()`: Add a source note citation

Add a source note to the footer part of the gt table. A source note is useful for citing the data included in the table. Several can be added to the footer, simply use multiple calls of `tab_source_note()` and they will be inserted in the order provided. We can use Markdown formatting for the note, or, if the table is intended for HTML output, we can include HTML formatting.

EXAMPLE

Use `gtcars` to create a **gt** table. Add a source note to the table footer that cites the data source.

```
gtcars %>%
  dplyr::select(mfr, model, msrp) %>%
  dplyr::slice(1:5) %>%
  gt() %>%
  tab_source_note(
    source_note = "From edmunds.com"
  )
```

mfr	model	msrp
Ford	GT	447000
Ferrari	458 Speciale	291744
Ferrari	458 Spider	263553
Ferrari	458 Italia	233509
Ferrari	488 GTB	245400

From edmunds.com

`tab_style()`: Add custom styles to one or more cells

With the `tab_style()` function we can target specific cells and apply styles to them. This is best done in conjunction with the helper functions `cell_text()`, `cell_fill()`, and `cell_borders()`. At present this function is focused on the application of styles for HTML output only (as such, other output formats will ignore all `tab_style()` calls). Using the aforementioned helper functions, here are some of the styles we can apply:

- the background color of the cell (`cell_fill()`: `color`)
- the cell's text color, font, and size (`cell_text()`: `color`, `font`, `size`)
- the text style (`cell_text()`: `style`), enabling the use of italics or oblique text.

- the text weight (`cell_text()` : `weight`), allowing the use of thin to bold text (the degree of choice is greater with variable fonts)
- the alignment and indentation of text (`cell_text()` : `align` and `indent`)
- the cell borders (`cell_borders()`)

EXAMPLES

Use `exibble` to create a **gt** table. Add styles that are to be applied to data cells that satisfy a condition (using `tab_style()`).

```
exibble %>%
  dplyr::select(num, currency) %>%
  gt() %>%
  fmt_number(
    columns = vars(num, currency),
    decimals = 1
  ) %>%
  tab_style(
    style = list(
      cell_fill(color = "lightcyan"),
      cell_text(weight = "bold")
      ),
    locations = cells_body(
      columns = vars(num),
      rows = num >= 5000)
  ) %>%
  tab_style(
    style = list(
      cell_fill(color = "#F9E3D6"),
      cell_text(style = "italic")
      ),
    locations = cells_body(
      columns = vars(currency),
      rows = currency < 100)
  )
```

num	currency
0.1	*50.0*
2.2	*17.9*
33.3	*1.4*
444.4	65,100.0
5,550.0	1,325.8
NA	*13.3*
777,000.0	NA
8,880,000.0	*0.4*

Use `sp500` to create a **gt** table. Color entire rows of cells based on values in a particular column.

```
sp500 %>%
  dplyr::filter(
    date >= "2015-12-01" &
    date <= "2015-12-15"
  ) %>%
  dplyr::select(-c(adj_close, volume)) %>%
  gt() %>%
  tab_style(
    style = cell_fill(color = "lightgreen"),
    locations = cells_body(
      rows = close > open)
  ) %>%
  tab_style(
    style = list(
      cell_fill(color = "tomato"),
      cell_text(color = "white")
      ),
    locations = cells_body(
      rows = open > close)
  )
```

date	open	high	low	close
2015-12-15	2025.55	2053.87	2025.55	2043.41
2015-12-14	2013.37	2022.92	1993.26	2021.94
2015-12-11	2047.27	2047.27	2008.80	2012.37
2015-12-10	2047.93	2067.65	2045.67	2052.23
2015-12-09	2061.17	2080.33	2036.53	2047.62
2015-12-08	2073.39	2073.85	2052.32	2063.59
2015-12-07	2090.42	2090.42	2066.78	2077.07
2015-12-04	2051.24	2093.84	2051.24	2091.69
2015-12-03	2080.71	2085.00	2042.35	2049.62
2015-12-02	2101.71	2104.27	2077.11	2079.51
2015-12-01	2082.93	2103.37	2082.93	2102.63

`tab_options()` : Modify the table output options

Modify the options available in a table. These options are named by the components, the subcomponents, and the element that can adjusted.

EXAMPLES

Use `exibble` to create a **gt** table with all the main parts added; we can use this going forward to demo some `tab_options()`.

```
tab_1 <-
  exibble %>%
  dplyr::select(
    -c(fctr, date, time, datetime)
  ) %>%
  gt(
    rowname_col = "row",
    groupname_col = "group"
  ) %>%
  tab_header(
    title = md("Data listing from **exibble**"),
    subtitle = md("`exibble` is an R dataset")
  ) %>%
  fmt_number(columns = vars(num)) %>%
  fmt_currency(columns = vars(currency)) %>%
  tab_footnote(
    footnote = "Using commas for separators.",
    locations = cells_body(
      columns = vars(num),
      rows = num > 1000)
  ) %>%
  tab_footnote(
    footnote = "Using commas for separators.",
    locations = cells_body(
      columns = vars(currency),
      rows = currency > 1000)
  ) %>%
  tab_footnote(
    footnote = "Alphabetical fruit.",
    locations = cells_column_labels(
      columns = vars(char))
  )
```

Modify the table width to `100%` (which spans the entire content width area).

```
tab_1 %>%
  tab_options(
    table.width = pct(100)
  )
```

Data listing from **exibble**

exibble is an R dataset

	num	char[1]	currency
grp_a			
row_1	0.11	apricot	$49.95

[1] Alphabetical fruit.

[2] Using commas for separators.

Data listing from **exibble**

exibble is an R dataset

	num	char	currency
row_2	2.22	banana	$17.95
row_3	33.33	coconut	$1.39
row_4	444.40	durian	$65,100.00[2]
grp_b			
row_5	5,550.00[2]	NA	$1,325.81[2]
row_6	NA	fig	$13.26
row_7	777,000.00[2]	grapefruit	NA
row_8	8,880,000.00[2]	honeydew	$0.44

[1] Alphabetical fruit.

[2] Using commas for separators.

Modify the table's background color to be "lightcyan".

```
tab_1 %>%
  tab_options(
    table.background.color = "lightcyan"
  )
```

Data listing from **exibble**

exibble is an R dataset

	num	char[1]	currency
grp_a			
row_1	0.11	apricot	$49.95
row_2	2.22	banana	$17.95
row_3	33.33	coconut	$1.39
row_4	444.40	durian	$65,100.00[2]
grp_b			
row_5	5,550.00[2]	NA	$1,325.81[2]

[1] Alphabetical fruit.

[2] Using commas for separators.

Data listing from **exibble**

exibble is an R dataset

row_6	NA	fig	$13.26
row_7	777,000.00[2]	grapefruit	NA
row_8	8,880,000.00[2]	honeydew	$0.44

[1] Alphabetical fruit.

[2] Using commas for separators.

Use letters as the glyphs for footnote references; also, separate footnotes in the footer by spaces instead of newlines.

```
tab_1 %>%
  tab_options(
    footnotes.sep = " ",
    footnotes.marks = letters
  )
```

Data listing from **exibble**

exibble is an R dataset

	num	char[a]	currency
grp_a			
row_1	0.11	apricot	$49.95
row_2	2.22	banana	$17.95
row_3	33.33	coconut	$1.39
row_4	444.40	durian	$65,100.00[b]
grp_b			
row_5	5,550.00[b]	NA	$1,325.81[b]
row_6	NA	fig	$13.26
row_7	777,000.00[b]	grapefruit	NA
row_8	8,880,000.00[b]	honeydew	$0.44

[a] Alphabetical fruit.

[b] Using commas for separators.

Change the padding of data rows to `5px` .

```
tab_1 %>% tab_options(data_row.padding = px(5))
```

Data listing from **exibble**

exibble is an R dataset

	num	char[1]	currency
grp_a			
row_1	0.11	apricot	$49.95
row_2	2.22	banana	$17.95
row_3	33.33	coconut	$1.39
row_4	444.40	durian	$65,100.00[2]
grp_b			
row_5	5,550.00[2]	NA	$1,325.81[2]
row_6	NA	fig	$13.26
row_7	777,000.00[2]	grapefruit	NA
row_8	8,880,000.00[2]	honeydew	$0.44

[1] Alphabetical fruit.

[2] Using commas for separators.

Reduce the size of the title and the subtitle text.

```
tab_1 %>%
  tab_options(
    heading.title.font.size = "small",
    heading.subtitle.font.size = "small"
  )
```

Data listing from **exibble**

exibble is an R dataset

	num	char[1]	currency
grp_a			
row_1	0.11	apricot	$49.95
row_2	2.22	banana	$17.95

[1] Alphabetical fruit.

[2] Using commas for separators.

Data listing from **exibble**

`exibble` is an R dataset

row_3	33.33	coconut	$1.39
row_4	444.40	durian	$65,100.00[2]
grp_b			
row_5	5,550.00[2]	NA	$1,325.81[2]
row_6	NA	fig	$13.26
row_7	777,000.00[2]	grapefruit	NA
row_8	8,880,000.00[2]	honeydew	$0.44

[1] Alphabetical fruit.

[2] Using commas for separators.