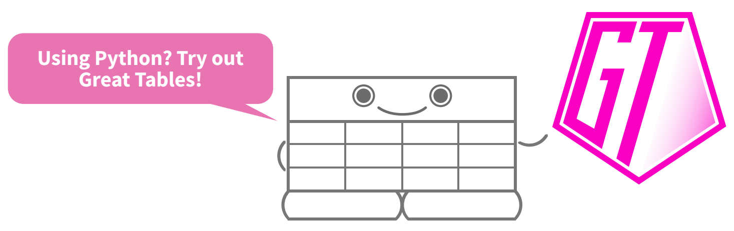
gt : : CHEAT SHEET

Introduction

The gt package allows you to make beautiful, informative table displays from R data frames. The package provides functions for you to **structure**, **format**, and **style** the table to your heart's content.

The resulting tables can be easily displayed in Shiny apps and Quarto documents. And there are quite a few output formats for your tables, including HTML, LaTeX, RTF, and Word.

INSTALLATION `install.packages("gt")`



Adding Structure

ADDING A HEADER AND FOOTER

`tab_header()` → add title and optional subtitle, can use `md()` or `html()` helpers

`tab_source_note()` → add text to table footer (above helpers can be used here too)

`tab_footnote()` → add footnote to footer; needs a *location* (see *Styling...* for this)

ADDING A STUB AND ROW GROUPS

`gt(rowname_col = "column")` → assign a column with row names to the stub

`gt(groupname_col = "column")` → use column with categoricals to group rows

ADDING COLUMN SPANNERS

`tab_spanner()` → add spanner over cols

`tab_spanner_delim()` → use delimited column names to set up multiple spanners

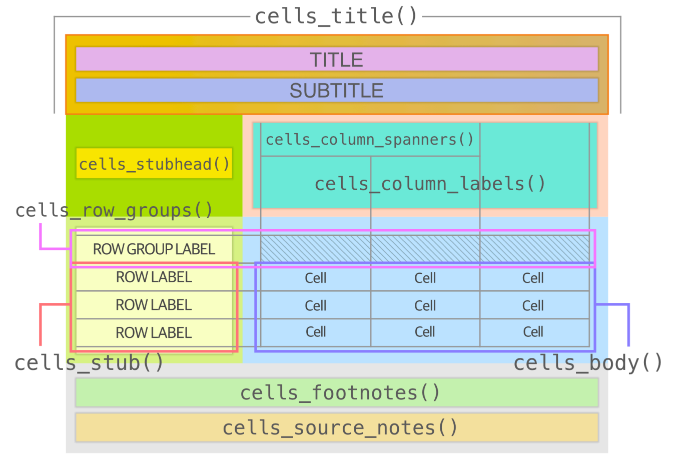
Styling the Table

ONE FUNCTION FOR STYLING EVERYWHERE

`tab_style()` → apply styles to various locations

```
gt_tbl |> tab_style(style = list(styles), locations = list(locations))
```

THE DIFFERENT LOCATIONS OF A TABLE



THREE WAYS TO STYLE

`cell_fill()` → fill location with color

`cell_text()` → style location's text

`cell_borders()` → set up borders and define styles for them

YOU ALSO HAVE OTHER OPTIONS

`tab_options()` → many, many options for styling that applies to whole table

`opt_stylize()` → set a theme, 36 looks

`opt_table_font()` → set a font to use

Creating a gt Table

BASIC EXAMPLE WITH SAMPLE DATA

```
library(gt)

gt_tbl = gt(exibble)
gt_tbl
```

create GT object

display HTML table

OUTPUT

column labels

num	char	fctr	date	time	datetime	currency	row	group
1.111e-01	apricot	one	2015-01-15	13:35	2018-01-01 02:22	49.950	row_1	grp_a
2.222e+00	banana	two	2015-02-15	14:40	2018-02-02 14:33	17.950	row_2	grp_a
3.333e+01	coconut	three	2015-03-15	15:45	2018-03-03 03:44	1.390	row_3	grp_a
4.444e+02	durian	four	2015-04-15	16:50	2018-04-04 15:55	65100.000	row_4	grp_a
5.550e+03	NA	five	2015-05-15	17:55	2018-05-05 04:00	1325.810	row_5	grp_b
NA	fig	six	2015-06-15	NA	2018-06-06 16:11	13.255	row_6	grp_b
7.770e+05	grapefruit	seven	NA	19:10	2018-07-07 05:22	NA	row_7	grp_b
8.880e+06	honeydew	eight	2015-08-15	20:20	NA	0.440	row_8	grp_b

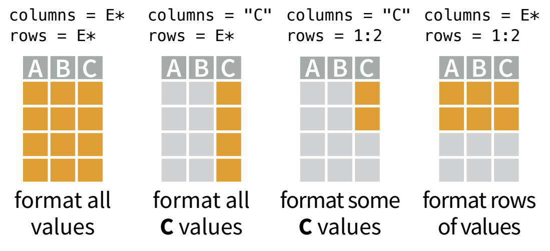
table body

SAMPLE DATASETS `countrypops`, `sza`, `gtcars`, `sp500`, `films`, `pizzaplace`, `exibble`, `towny`, `peeps` + more

Formatting Data Values

FORMATTING BASICS

Formatting functions have the `columns` and `rows` options to specify which body values should be formatted.



`E* means everything()`

NUMERIC FORMATTING

1 `fmt_number()` → decimal values
2 `fmt_integer()` → integer values
3 `fmt_percent()` → percentage values
4 `fmt_scientific()` → scientific not'n
5 `fmt_engineering()` → engineering not'n

common options: `decimals` in 1,3,4,5 for fixed # of decimals / `scale_by` in 1,2,4,5 to multiply number before formatting / `accounting` in 1,2,3 to display in that notation / `suffixing` in 1,2 to condense large numbers / `locale` in 1–5 to localize formatted values to language/region.

DATE/TIME FORMATTING

`fmt_date()` → format with a date style preset

```
iso: "2000-02-29" (the default)
wday_month_day_year: "Tuesday, February 29, 2000"
wd_m_day_year: "Tue, Feb 29, 2000"
wday_day_month_year: "Tuesday 29 February 2000"
month_day_year: "February 29, 2000"
```

`fmt_time()` → format with a time style preset

```
iso: "14:35:00" (the default)
h_m_s_p: "2:35:00 PM"
h_m_p: "2:35 PM"
```

`fmt_datetime()` → format with date/time style presets or with `format` option for more control

FORMATTING INVOLVING GRAPHICS

`fmt_image()` → insert images to cells with filenames; `path` / `file_pattern` help resolve

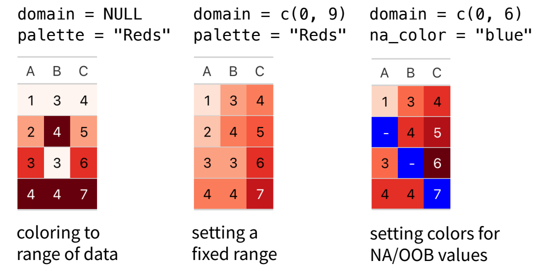
`fmt_icon()` → add icons to cells containing FontAwesome short icon names (e.g., `"cat"`)

`fmt_flag()` → add flag icons to cells w/ 2- or 3-letter country codes (e.g., `"LU"` or `"LUX"`)

common to all: cells may have multiple instances of replaceable text for these formatters, they must be comma-separated though.

COLORIZING BODY CELLS

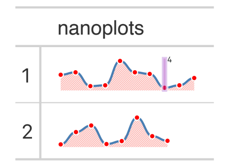
`data_color()` → apply color to a series of cells according to their value


domain = NULL palette = "Reds"
domain = c(0, 9) palette = "Reds"
domain = c(0, 6) na_color = "blue"
A B C
1 3 4
2 4 5
3 3 6
4 4 7
1 3 4
2 4 5
3 3 6
4 4 7
1 3 4
- 4 5
3 - 6
4 4 7
coloring to range of data
setting a fixed range
setting colors for NA/OOB values

ADDING NANOPLOTS

`cols_nanoplot()` → convert series of values (strings w/ vals or a list column) to tiny interactive plots in a new column

	vals
1	20 23 6 7 37 23 21 4 7 16
2	2.3 6.8 9.2 2.42 3.5 12.1 5.3 3.6


nanoplots