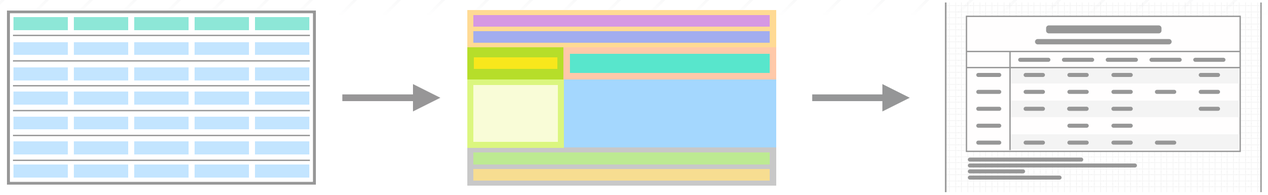
Making Tables with **gt** and **Great Tables**



About Us

Software Engineers at Posit, PBC.

Collectively have:

— 2 PhDs 📜📜

— 2 dogs 🐶🐶

— 5 cats 🐈🐈🐈🐈🐈

We are table display fanatics!

posit::conf(2024)


Michael Chow


Richard Iannone

Workshop Structure

There are four sessions (1.5 h each) we will go through this basic alternation in each session:

— slides

These slides. This will give us a basic understanding of new concepts.

— code along

We'll move to the IDE to work with **gt** or **Great Tables** code.

You'll also get a chance to work semi-independently and figure things out during a few exercise sessions.

Workshop Schedule

This is our schedule for today's workshop

Time	Activity
9:00am – 10:30am	Session 1: Coffee Table **Great Tables**
10:30am – 11:00am	Coffee Break
11:00am – 12:30pm	Session 2: Reactions Table **gt**
12:30pm – 1:30pm	Lunch
1:30pm – 3:00pm	Session 3: Power Generation Table **gt**
3:00pm – 3:30pm	Coffee Break
3:30pm – 5:00pm	Session 4: Make Your Own Excellent Tables

Workshop Materials and Online Discussion

All workshop content is available at:

`https://github.com/posit-conf-2024/tables`

If using Posit Cloud (for R or Python), then follow instructions in

`code-along/py-00-setup.qmd` `code-along/r-00-setup.qmd`

If you'd like to ask questions (and share code) we have a Discord for this:

`https://discord.com/invite/Ux7nrcXHVV`

Once you are inside the server, the channel for discussion is called `#tables-workshop`

Some Other Important Information

Facilities:

Gender-neutral bathrooms on **levels 3–7**

Meditation/prayer room is **Room 503**

Mother's room is **Room 509**

Open times: 7am–7pm (Mon, Tue), 7am–5pm (Wed)

Photography/Recording:

Workshop will not be recorded.

People wearing red lanyards: **don't photograph them**.

WiFi: name: `Posit Conf 2024` password: `conf2024`

Our Code of Conduct: `https://posit.co/code-of-conduct/`

Display Tables... What Do We Want?

shape: (15, 5)

product	revenue_dollars	revenue_pct	margin_dollars	margin_pct
str	f64	f64	f64	f64
"Grinder"	904.5	0.03	567.96	0.04
"Moka pot"	2045.25	0.07	181.08	0.01
"Cold brew"	288.75	0.01	241.77	0.02
"Filter"	404.25	0.01	70.01	0.02
"Drip machine"	2632.0	0.1	1374.45	0.09
…	…	…	…	…
"Dripper"	575.75	0.02	139.02	0.01
"Scale"	3801.0	0.13	2910.29	0.19
"Kettle"	756.25	0.02	617.52	0.04
"Espresso Machi…	8406.0	0.28	3636.44	0.24
"Total"	30284.25	1.0	14932.16	1.0

Less of This

Sales of Coffee Equipment

		Revenue		Profit		
icon	product	Earnings	Fraction	Earnings	Fraction	monthly_sales
	Grinder	$904K	3%	$568K	4%	
	Moka pot	$2.05M	7%	$181K	1%	
	Cold brew	$289K	1%	$242K	2%	
	Filter	$404K	1%	$70.0K	0%	
	Drip machine	$2.63M	9%	$1.37M	9%	
	AeroPress	$2.60M	9%	$1.29M	9%	
	Pour over	$846K	3%	$365K	2%	
	French press	$1.11M	4%	$748K	5%	
	Cezve	$2.51M	9%	$1.97M	13%	
	Chemex	$3.14M	11%	$818K	6%	
	Scale	$3.80M	13%	$2.91M	20%	
	Kettle	$756K	3%	$618K	4%	
	Espresso Machine	$8.41M	29%	$3.64M	25%	
	Total	**$29.4M**	**102%**	**$14.8M**	**100%**	

More of This

Beautiful Tables from the Internet

The heading explains the purpose of the table.

History does not bode well for the Hoosiers

Only one future tournament team made fewer 3PTs through their first six games than Indiana in 2024.

	Shooting					
TEAM	3FG	3FG%	PER GAME	SEED	ROUND	YEAR
Wake Forest	17-61	27.87%	2.83	4	R64	2009
Indiana	**19-79**	**24.05%**	**3.17**	**???**	**???**	**2024**
North Carolina	20-60	33.33%	3.33	6	R32	2014
Coppin St.	21-60	35.00%	3.50	16	R68	2008
Vermont	22-89	24.72%	3.67	16	R64	2010
New Mexico St.	22-72	30.56%	3.67	13	R64	2014

Viz. + Analysis by @andreweatherman

This spanner groups like columns together.

Team logos quickly convey the identity of each row.

Highlighted row draws attention to main subject.

Percentage values formatted for high readability.

Beautiful Tables from the Internet

The rows are subdivided into groups for better organization.

Player Report Card: Kyle Filipowski

Position groups are determined by Barttorvik. Data through Jan 16, 2024

STAT	VALUE	Percentile relative to bigs across... NATIONAL	HIGH-MAJORS	CONF. RANK
Box (Per-Game)				
Minutes	29.8	92%	86%	28
Points	18.2	98%	97%	5
Off. Rebounds	2.3	62%	51%	9
Def. Rebounds	6.5	94%	91%	2
Turnovers	1.7	30%	28%	54
Shooting				
eFG%	56.7%	51%	35%	22
3FG%	41.2%	84%	84%	21
FT%	68.4%	66%	65%	56
3FGA/FGA[1]	24.4	87%	89%	68
Advanced				
Off. Rating[2]	119.9	83%	66%	17
Def. Rating	88.9	95%	89%	2
Usage	28.8	96%	97%	5
PORPAG[3]	5.1	99%	98%	2

[1] 3FG attempts per 100 FGA

[2] Offensive and defensive ratings represent points scored/allowed per 100 possessions

[3] PORPAG represents points above replacement player at that usage

These bar charts enable fast visual comparisons.

The footnotes here provide additional detail.

Beautiful Tables from the Internet

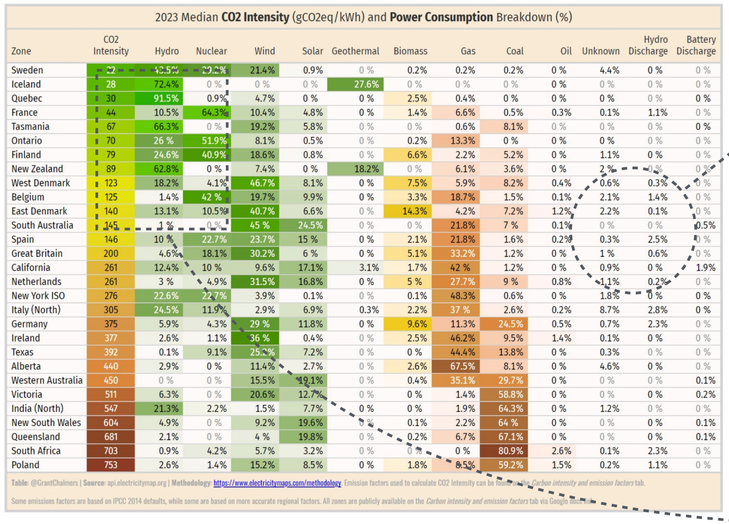
2023 Median **CO2 Intensity** (gCO2eq/kWh) and **Power Consumption** Breakdown (%)

Zone	CO2 Intensity	Hydro	Nuclear	Wind	Solar	Geothermal	Biomass	Gas	Coal	Oil	Unknown	Hydro Discharge	Battery Discharge
Sweden	22	43.5%	29.2%	21.4%	0.9%	0 %	0.2%	0.2%	0.2%	0 %	4.4%	0 %	0 %
Iceland	28	72.4%	0 %	0 %	0 %	27.6%	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Quebec	30	91.5%	0.9%	4.7%	0 %	0 %	2.5%	0.4%	0 %	0 %	0 %	0 %	0 %
France	44	10.5%	64.3%	10.4%	4.8%	0 %	1.4%	6.6%	0.5%	0.3%	0.1%	1.1%	0 %
Tasmania	67	66.3%	0 %	19.2%	5.8%	0 %	0 %	0.6%	8.1%	0 %	0 %	0 %	0 %
Ontario	70	26 %	51.9%	8.1%	0.5%	0 %	0.2%	13.3%	0 %	0 %	0 %	0 %	0 %
Finland	79	24.6%	40.9%	18.6%	0.8%	0 %	6.6%	2.2%	5.2%	0 %	1.1%	0 %	0 %
New Zealand	89	62.8%	0 %	7.4%	0 %	18.2%	0 %	6.1%	3.6%	0 %	2 %	0 %	0 %
West Denmark	123	18.2%	4.1%	46.7%	8.1%	0 %	7.5%	5.9%	8.2%	0.4%	0.6%	0.3%	0 %
Belgium	125	1.4%	42 %	19.7%	9.9%	0 %	3.3%	18.7%	1.5%	0.1%	2.1%	1.4%	0 %
East Denmark	140	13.1%	10.5%	40.7%	6.6%	0 %	14.3%	4.2%	7.2%	1.2%	2.2%	0.1%	0 %
South Australia	145	1 %	0 %	45 %	24.5%	0 %	0 %	21.8%	7 %	0.1%	0 %	0 %	0.5%
Spain	146	10 %	22.7%	23.7%	15 %	0 %	2.1%	21.8%	1.6%	0.2%	0.3%	2.5%	0 %
Great Britain	200	4.6%	18.1%	30.2%	6 %	0 %	5.1%	33.2%	1.2%	0 %	1 %	0.6%	0 %
California	261	12.4%	10 %	9.6%	17.1%	3.1%	1.7%	42 %	1.2%	0 %	0.9%	0 %	1.9%
Netherlands	261	3 %	4.9%	31.5%	16.8%	0 %	5 %	27.7%	9 %	0.8%	1.1%	0.2%	0 %
New York ISO	276	22.6%	22.7%	3.9%	0.1%	0 %	0.1%	48.3%	0.6%	0 %	1.8%	0 %	0 %
Italy (North)	305	24.5%	11.9%	2.9%	6.9%	0.3%	2.2%	37 %	2.6%	0.2%	8.7%	2.8%	0 %
Germany	375	5.9%	4.3%	29 %	11.8%	0 %	9.6%	11.3%	24.5%	0.5%	0.7%	2.3%	0 %
Ireland	377	2.6%	1.1%	36 %	0.4%	0 %	2.5%	46.2%	9.5%	1.4%	0.1%	0 %	0 %
Texas	392	0.1%	9.1%	25.2%	7.2%	0 %	0 %	44.4%	13.8%	0 %	0.3%	0 %	0 %
Alberta	440	2.9%	0 %	11.4%	2.7%	0 %	2.6%	67.5%	8.1%	0 %	4.6%	0 %	0 %
Western Australia	450	0 %	0 %	15.5%	19.1%	0 %	0.4%	35.1%	29.7%	0 %	0 %	0 %	0.1%
Victoria	511	6.3%	0 %	20.6%	12.7%	0 %	0 %	1.4%	58.8%	0 %	0 %	0 %	0.2%
India (North)	547	21.3%	2.2%	1.5%	7.7%	0 %	0 %	1.9%	64.3%	0 %	1.2%	0 %	0 %
New South Wales	604	4.9%	0 %	9.2%	19.6%	0 %	0.1%	2.2%	64 %	0 %	0 %	0 %	0.1%
Queensland	681	2.1%	0 %	4 %	19.8%	0 %	0.2%	6.7%	67.1%	0 %	0 %	0 %	0.1%
South Africa	703	0.9%	4.2%	5.7%	3.2%	0 %	0 %	0 %	80.9%	2.6%	0.1%	2.3%	0 %
Poland	753	2.6%	1.4%	15.2%	8.5%	0 %	1.8%	8.5%	59.2%	1.5%	0.2%	1.1%	0 %

Table: @GrantChalmers | **Source**: api.electricitymap.org | **Methodology**: https://www.electricitymaps.com/methodology Emission factors used to calculate CO2 Intensity can be found on the *Carbon intensity and emission factors* tab.

Some emissions factors are based on IPCC 2014 defaults, while some are based on more accurate regional factors. All zones are publicly available on the *Carbon intensity and emission factors* tab via Google docs link

Beautiful Tables from the Internet



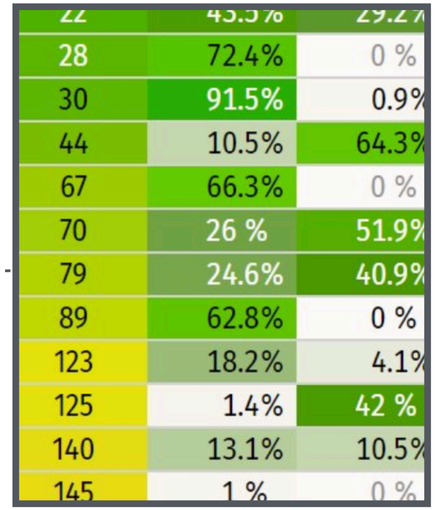
2023 Median **CO2 Intensity** (gCO2eq/kWh) and **Power Consumption** Breakdown (%)

Zone	CO2 Intensity	Hydro	Nuclear	Wind	Solar	Geothermal	Biomass	Gas	Coal	Oil	Unknown	Hydro Discharge	Battery Discharge
Sweden	22	42.5%	29.2%	21.4%	0.9%	0 %	0.2%	0.2%	0.2%	0 %	4.4%	0 %	0 %
Iceland	28	72.4%	0 %	0 %	0 %	27.6%	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Quebec	30	91.5%	0.9%	4.7%	0 %	0 %	2.5%	0.4%	0 %	0 %	0 %	0 %	0 %
France	44	10.5%	64.3%	10.4%	4.8%	0 %	1.4%	6.6%	0.5%	0.3%	0.1%	1.1%	0 %
Tasmania	67	66.3%	0 %	19.2%	5.8%	0 %	0 %	0.6%	8.1%	0 %	0 %	0 %	0 %
Ontario	70	26 %	51.9%	8.1%	0.5%	0 %	0.2%	13.3%	0 %	0 %	0 %	0 %	0 %
Finland	79	24.6%	40.9%	18.6%	0.8%	0 %	6.6%	2.2%	5.2%	0 %	1.1%	0 %	0 %
New Zealand	89	62.8%	0 %	7.4%	0 %	18.2%	0 %	6.1%	3.6%	0 %	2 %	0 %	0 %
West Denmark	123	18.2%	4.1%	46.7%	8.1%	0 %	7.5%	5.9%	8.2%	0.4%	0.6%	0.3%	0 %
Belgium	125	1.4%	42 %	19.7%	9.9%	0 %	3.3%	18.7%	1.5%	0.1%	2.1%	1.4%	0 %
East Denmark	140	13.1%	10.5%	40.7%	6.6%	0 %	14.3%	4.2%	7.2%	1.2%	2.2%	0.1%	0 %
South Australia	145	1 %	0 %	45 %	24.5%	0 %	0 %	21.8%	7 %	0.1%	0 %	0 %	0.5%
Spain	146	10 %	22.7%	23.7%	15 %	0 %	2.1%	21.8%	1.6%	0.2%	0.3%	2.5%	0 %
Great Britain	200	4.6%	18.1%	30.2%	6 %	0 %	5.1%	33.2%	1.2%	0 %	1 %	0.6%	0 %
California	261	12.4%	10 %	9.6%	17.1%	3.1%	1.7%	42 %	1.2%	0 %	0.9%	0 %	1.9%
Netherlands	261	3 %	4.9%	31.5%	16.8%	0 %	5 %	27.7%	9 %	0.8%	1.1%	0.2%	0 %
New York ISO	276	22.6%	22.7%	3.9%	0.1%	0 %	0.1%	48.3%	0.6%	0 %	1.8%	0 %	0 %
Italy (North)	305	24.5%	11.9%	2.9%	6.9%	0.3%	2.2%	37 %	2.6%	0.2%	8.7%	2.8%	0 %
Germany	375	5.9%	4.3%	29 %	11.8%	0 %	9.6%	11.3%	24.5%	0.5%	0.7%	2.3%	0 %
Ireland	377	2.6%	1.1%	36 %	0.4%	0 %	2.5%	46.2%	9.5%	1.4%	0.1%	0 %	0 %
Texas	392	0.1%	9.1%	25.2%	7.2%	0 %	0 %	44.4%	13.8%	0 %	0.3%	0 %	0 %
Alberta	440	2.9%	0 %	11.4%	2.7%	0 %	2.6%	67.5%	8.1%	0 %	4.6%	0 %	0 %
Western Australia	450	0 %	0 %	15.5%	19.1%	0 %	0.4%	35.1%	29.7%	0 %	0 %	0 %	0.1%
Victoria	511	6.3%	0 %	20.6%	12.7%	0 %	0 %	1.4%	58.8%	0 %	0 %	0 %	0.2%
India (North)	547	21.3%	2.2%	1.5%	7.7%	0 %	0 %	1.9%	64.3%	0 %	1.2%	0 %	0 %
New South Wales	604	4.9%	0 %	9.2%	19.6%	0 %	0.1%	2.2%	64 %	0 %	0 %	0 %	0.1%
Queensland	681	2.1%	0 %	4 %	19.8%	0 %	0.2%	6.7%	67.1%	0 %	0 %	0 %	0.1%
South Africa	703	0.9%	4.2%	5.7%	3.2%	0 %	0 %	0 %	80.9%	2.6%	0.1%	2.3%	0 %
Poland	753	2.6%	1.4%	15.2%	8.5%	0 %	1.8%	8.5%	59.2%	1.5%	0.2%	1.1%	0 %

Table: @GrantChalmers | Source: api.electricitymap.org | Methodology: https://www.electricitymaps.com/methodology Emission factors used to calculate CO2 Intensity can be found in the *Carbon intensity and emission factors* tab.

Some emissions factors are based on IPCC 2014 defaults, while some are based on more accurate regional factors. All zones are publicly available on the *Carbon intensity and emission factors* tab via Google Sheets.

Nicely formatted percentage values with decimal alignment.



Heat map makes it easier to scan the data values and it aids comparisons.

These Tables: Made from Code

History does not bode well for the Hoosiers

Only one future tournament team made fewer 3PTs through their first six games than Indiana in 2024.

	Shooting					
TEAM	3FG	3FG%	PER GAME	SEED	ROUND	YEAR
Wake Forest	17-61	27.87%	2.83	4	R64	2009
Indiana	**19-79**	**24.05%**	**3.17**	**???**	**???**	**2024**
North Carolina	20-60	33.33%	3.33	6	R32	2014
Coppin St.	21-60	35.00%	3.50	16	R68	2008
Vermont	22-89	24.72%	3.67	16	R64	2010
New Mexico St.	22-72	30.56%	3.67	13	R64	2014

Viz. + Analysis by @andreweatherman

2023 Median **CO2 Intensity** (gCO2eq/kWh) and **Power Consumption** Breakdown (%)

Zone	CO2 Intensity	Hydro	Nuclear	Wind	Solar	Geothermal	Biomass	Gas	Coal	Oil	Unknown	Hydro Discharge	Battery Discharge
Sweden	22	43.5%	29.2%	21.4%	0.9%	0 %	0.2%	0.2%	0.2%	0 %	4.4%	0 %	0 %
Iceland	28	72.4%	0 %	0 %	0 %	27.6%	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Quebec	30	91.5%	0.9%	4.7%	0 %	0 %	2.5%	0.4%	0 %	0 %	0 %	0 %	0 %
France	44	10.5%	64.3%	10.4%	4.8%	0 %	1.4%	6.6%	0.5%	0.3%	0.1%	1.1%	0 %
Tasmania	67	66.3%	0 %	19.2%	5.8%	0 %	0 %	0.6%	8.1%	0 %	0 %	0 %	0 %
Ontario	70	26 %	51.9%	8.1%	0.5%	0 %	0.2%	13.3%	0 %	0 %	0 %	0 %	0 %
Finland	79	24.6%	40.9%	18.6%	0.8%	0 %	6.6%	2.2%	5.2%	0 %	1.1%	0 %	0 %
New Zealand	89	62.8%	0 %	7.4%	0 %	18.2%	0 %	6.1%	3.6%	0 %	2 %	0 %	0 %
West Denmark	123	18.2%	4.1%	46.7%	8.1%	0 %	7.5%	5.9%	8.2%	0.4%	0.6%	0.3%	0 %
Belgium	125	1.4%	42 %	19.7%	9.9%	0 %	3.3%	18.7%	1.5%	0.1%	2.1%	1.4%	0 %
East Denmark	140	13.1%	10.5%	40.7%	6.6%	0 %	14.3%	4.2%	7.2%	1.2%	2.2%	0.1%	0 %
South Australia	145	1 %	0 %	45 %	24.5%	0 %	0 %	21.8%	7 %	0.1%	0 %	0 %	0.5%
Spain	146	10 %	22.7%	23.7%	15 %	0 %	2.1%	21.8%	1.6%	0.2%	0.3%	2.5%	0 %
Great Britain	200	4.6%	18.1%	30.2%	6 %	0 %	5.1%	31.2%	1.2%	0 %	1 %	0.6%	0 %
California	261	12.4%	10 %	9.6%	17.1%	3.1%	1.7%	42 %	1.2%	0 %	0.9%	0 %	1.9%
Netherlands	261	3 %	4.9%	31.5%	16.8%	0 %	5 %	27.7%	9 %	0.8%	1.1%	0.2%	0 %
New York ISO	276	22.6%	22.7%	3.9%	0.1%	0 %	0.1%	48.3%	0.6%	0 %	1.8%	0 %	0 %
Italy (North)	305	24.5%	11.9%	2.9%	6.9%	0.3%	2.2%	37 %	2.6%	0.2%	8.7%	2.8%	0 %
Germany	375	5.9%	4.3%	29 %	11.8%	0 %	9.6%	11.3%	24.5%	0.5%	0.7%	2.3%	0 %
Ireland	377	2.6%	1.1%	36 %	0.4%	0 %	2.5%	46.2%	9.5%	1.4%	0.1%	0 %	0 %
Texas	392	0.1%	9.1%	25.2%	7.2%	0 %	0 %	44.4%	13.8%	0 %	0.3%	0 %	0 %
Alberta	440	2.9%	0 %	11.4%	2.7%	0 %	2.6%	67.5%	8.1%	0 %	4.6%	0 %	0 %
Western Australia	450	0 %	0 %	15.5%	19.1%	0 %	0.4%	35.1%	29.7%	0 %	0 %	0 %	0.1%
Victoria	511	6.3%	0 %	20.6%	12.7%	0 %	0 %	1.4%	58.8%	0 %	0 %	0 %	0.2%
India (North)	547	21.3%	2.2%	1.5%	7.7%	0 %	0 %	1.9%	64.3%	0 %	1.2%	0 %	0 %
New South Wales	604	4.9%	0 %	9.2%	19.6%	0 %	0.1%	2.2%	64 %	0 %	0 %	0 %	0.1%
Queensland	681	2.1%	0 %	4 %	19.8%	0 %	0.2%	6.7%	67.1%	0 %	0 %	0 %	0.1%
South Africa	703	0.9%	4.2%	5.7%	3.2%	0 %	0 %	0 %	80.9%	2.6%	0.1%	2.3%	0 %
Poland	753	2.6%	1.4%	15.2%	8.5%	0 %	1.8%	8.5%	59.2%	1.5%	0.2%	1.1%	0 %

Table: @GrantChalmers | Source: api.electricitymap.org | Methodology: https://www.electricitymaps.com/methodology Emission factors used to calculate CO2 Intensity can be found on the *Carbon intensity and emission factors* tab.

Some emissions factors are based on IPCC 2014 defaults, while some are based on more accurate regional factors. All zones are publicly available on the *Carbon intensity and emission factors* tab via Google docs link

Player Report Card: Kyle Filipowski

Position groups are determined by Barttorvik. Data through Jan 16, 2024

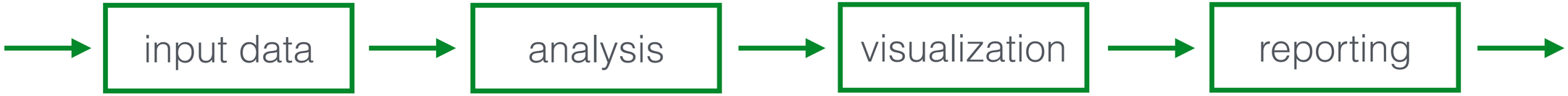
STAT	VALUE	Percentile relative to bigs across... NATIONAL	HIGH-MAJORS	CONF. RANK
Box (Per-Game)				
Minutes	29.8	92%	86%	28
Points	18.2	98%	97%	5
Off. Rebounds	2.3	62%	51%	9
Def. Rebounds	6.5	94%	91%	2
Turnovers	1.7	30%	28%	54
Shooting				
eFG%	56.7%	51%	35%	22
3FG%	41.2%	84%	84%	21
FT%	68.4%	66%	65%	56
3FGA/FGA[1]	24.4	87%	89%	68
Advanced				
Off. Rating[2]	119.9	83%	66%	17
Def. Rating	88.9	95%	89%	2
Usage	28.8	96%	97%	5
PORPAG[3]	5.1	99%	98%	2

[1] 3FG attempts per 100 FGA

[2] Offensive and defensive ratings represent points scored/allowed per 100 possessions

[3] PORPAG represents points above replacement player at that usage

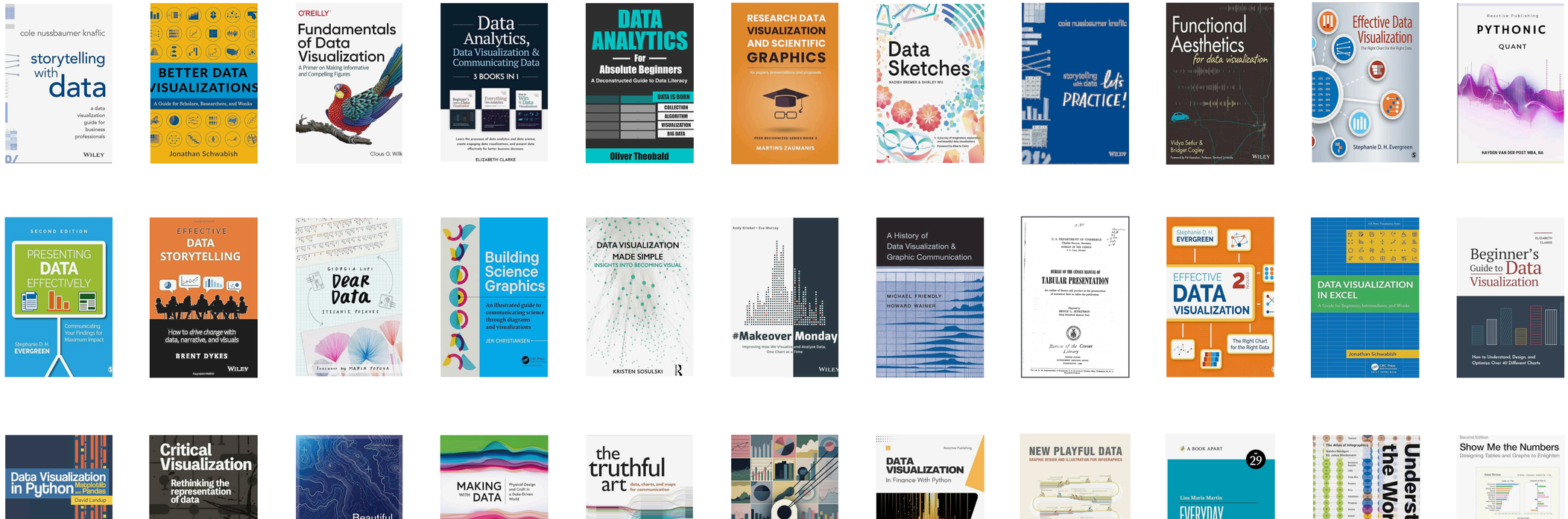
We benefit from a reproducible workflow.



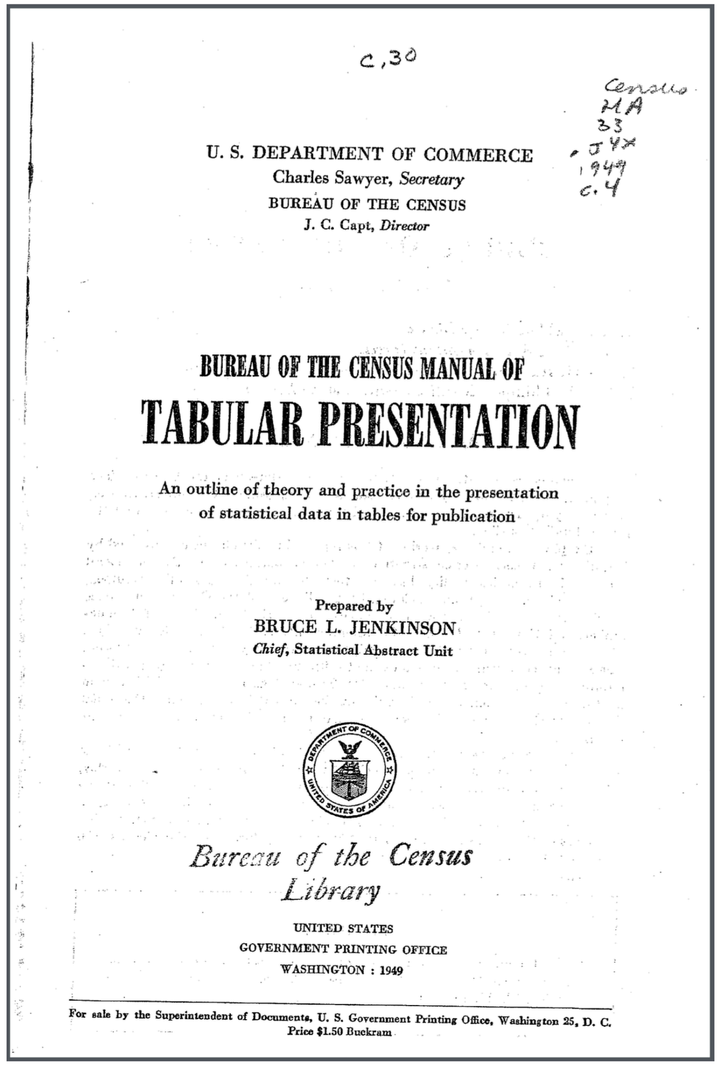
How Did We Get Here?

We needed to get the best ideas on table generation before devising an API.

Surprisingly, there weren't too many authoritative texts on table design, so we had to look hard...



How Did We Get Here?



c.30

U. S. DEPARTMENT OF COMMERCE
Charles Sawyer, *Secretary*
BUREAU OF THE CENSUS
J. C. Capt, *Director*

BUREAU OF THE CENSUS MANUAL OF

TABULAR PRESENTATION

An outline of theory and practice in the presentation of statistical data in tables for publication

Prepared by
BRUCE L. JENKINSON
Chief, Statistical Abstract Unit

UNITED STATES
GOVERNMENT PRINTING OFFICE
WASHINGTON : 1949

For sale by the Superintendent of Documents, U. S. Government Printing Office, Washington 25, D. C.
Price $1.50 Buckram

*This is the **Census Manual of Tabular Presentation**.*

*It dials concepts on table display to **11**.*

*It provides **many** solid and useful recommendations.*

How Did We Get Here?

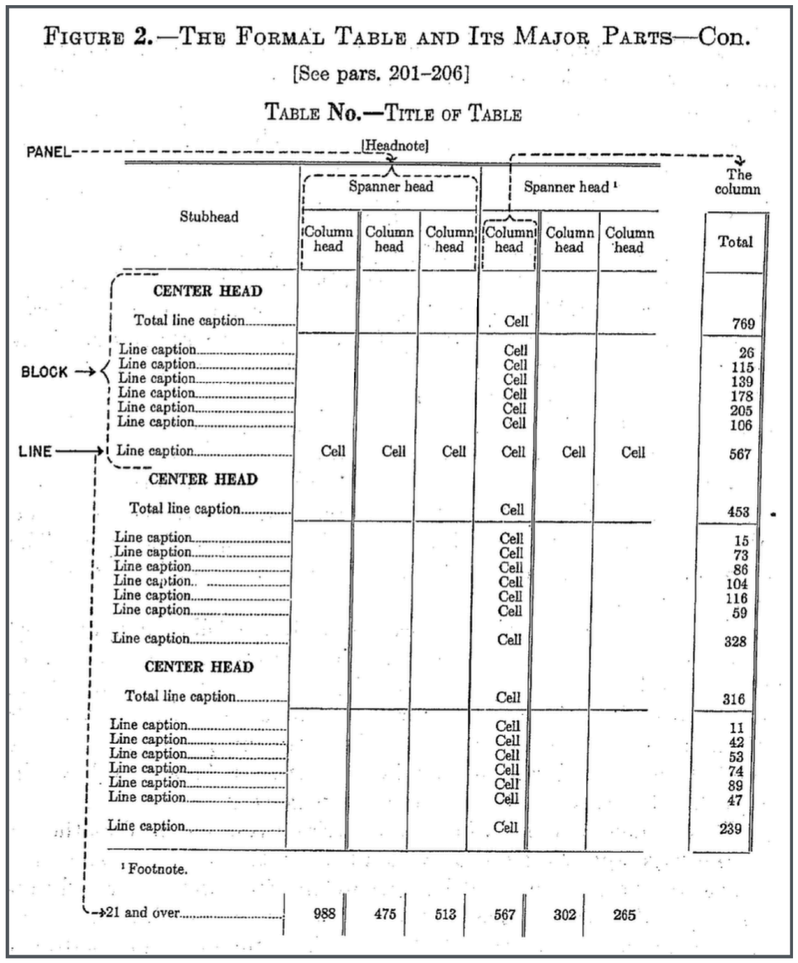


FIGURE 2.—THE FORMAL TABLE AND ITS MAJOR PARTS—Con.

[See pars. 201–206]

TABLE No.—TITLE OF TABLE

PANEL — [Headnote] — The column

Stubhead	Spanner head: Column head	Column head	Column head	Spanner head [1]: Column head	Column head	Column head	Total
CENTER HEAD							
Total line caption				Cell			769
BLOCK → Line caption				Cell			26
Line caption				Cell			115
Line caption				Cell			139
Line caption				Cell			178
Line caption				Cell			205
Line caption				Cell			106
LINE → Line caption	Cell	Cell	Cell	Cell	Cell	Cell	567
CENTER HEAD							
Total line caption				Cell			453
Line caption				Cell			15
Line caption				Cell			73
Line caption				Cell			86
Line caption				Cell			104
Line caption				Cell			116
Line caption				Cell			59
Line caption				Cell			328
CENTER HEAD							
Total line caption				Cell			316
Line caption				Cell			11
Line caption				Cell			42
Line caption				Cell			53
Line caption				Cell			74
Line caption				Cell			89
Line caption				Cell			47
Line caption				Cell			239
[1] Footnote.							
→21 and over	988	475	513	567	302	265	

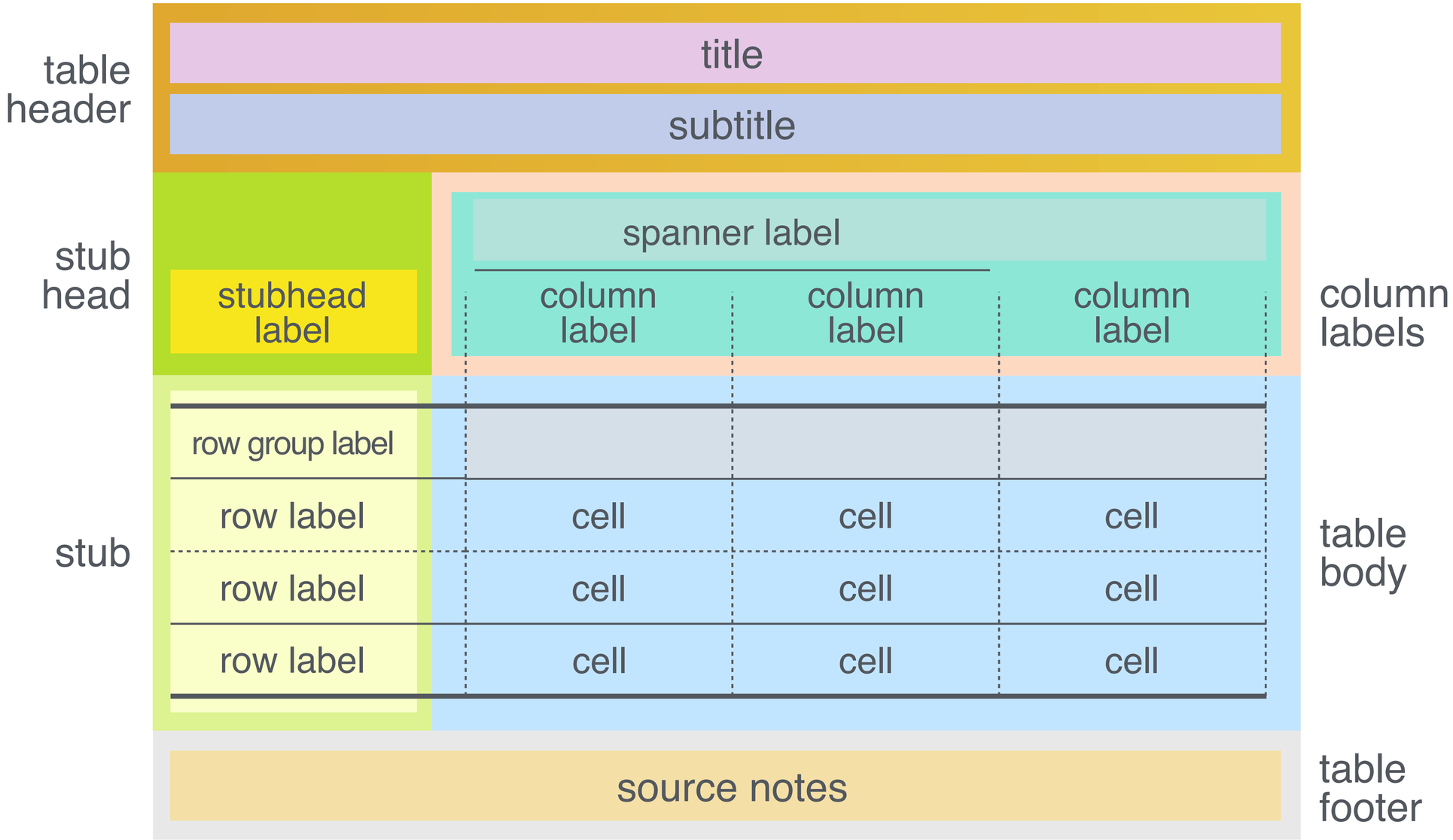
*This is the **Census Manual of Tabular Presentation**.*

*It dials concepts on table display to **11**.*

*It provides **many** solid and useful recommendations.*

*Importantly: it formalizes the **structure of a table**.*

Our Modern Take on a Table Display Framework



How Do You Make Tables Today?

Raw Table

shape: (15, 5)

product	revenue_dollars	revenue_pct	margin_dollars	margin_pct
str	f64	f64	f64	f64
"Grinder"	904.5	0.03	567.96	0.04
"Moka pot"	2045.25	0.07	181.08	0.01
"Cold brew"	288.75	0.01	241.77	0.02
"Filter"	404.25	0.01	70.01	0.02
"Drip machine"	2632.0	0.1	1374.45	0.09
…	…	…	…	…
"Dripper"	575.75	0.02	139.02	0.01
"Scale"	3801.0	0.13	2910.29	0.19
"Kettle"	756.25	0.02	617.52	0.04
"Espresso Machi…	8406.0	0.28	3636.44	0.24
"Total"	30284.25	1.0	14932.16	1.0

You could present this to others,
but it's not recommended.

How Do You Make Tables Today?

Raw Table

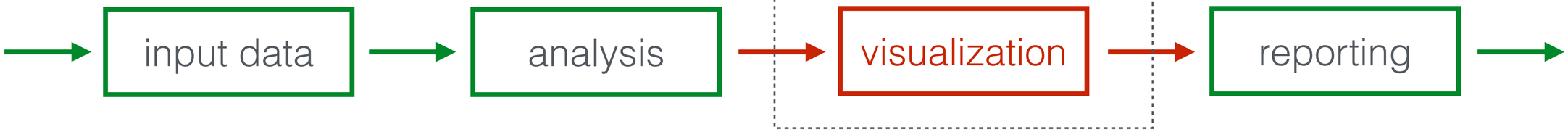
shape: (15, 5)

product	revenue_dollars	revenue_pct	margin_dollars	margin_pct
str	f64	f64	f64	f64
"Grinder"	904.5	0.03	567.96	0.04
"Moka pot"	2045.25	0.07	181.08	0.01
"Cold brew"	288.75	0.01	241.77	0.02
"Filter"	404.25	0.01	70.01	0.02
"Drip machine"	2632.0	0.1	1374.45	0.09
…	…	…	…	…
"Dripper"	575.75	0.02	139.02	0.01
"Scale"	3801.0	0.13	2910.29	0.19
"Kettle"	756.25	0.02	617.52	0.04
"Espresso Machi…	8406.0	0.28	3636.44	0.24
"Total"	30284.25	1.0	14932.16	1.0

Excel

Product	Revenue $ (000's)	Revenue %	Margin $ (000's)	Margin %
Grinder	$904.50	3%	$567.96	4%
Moka pot	$2,045.25	7%	$181.08	1%
Cold brew	$288.75	1%	$241.77	2%
Filter	$404.25	1%	$70.01	0%
Drip machine	$2,632.00	10%	$1,374.45	9%
AeroPress	$2,601.50	9%	$1,293.78	9%
Pour over	$846.00	3%	$364.53	2%
French press	$1,113.25	4%	$748.12	5%
Cezve	$2,512.50	8%	$1,969.52	13%
Chemex	$3,137.25	10%	$817.68	5%
Dripper	$575.75	2%	$139.02	1%
Scale	$3,801.00	13%	$2,910.29	19%
Kettle	$756.25	2%	$617.52	4%
Espresso Machine	$8,406.00	28%	$3,636.44	24%
Total	**$30,284.25**	**100%**	**$14,932.16**	**100%**

You could instead make a nice display table with Excel. But your reproducible workflow is now broken.



How Do You Make Tables Today?

Excel

Product	Revenue $ (000's)	Revenue %	Margin $ (000's)	Margin %
Grinder	$904.50	3%	$567.96	4%
Moka pot	$2,045.25	7%	$181.08	1%
Cold brew	$288.75	1%	$241.77	2%
Filter	$404.25	1%	$70.01	0%
Drip machine	$2,632.00	10%	$1,374.45	9%
AeroPress	$2,601.50	9%	$1,293.78	9%
Pour over	$846.00	3%	$364.53	2%
French press	$1,113.25	4%	$748.12	5%
Cezve	$2,512.50	8%	$1,969.52	13%
Chemex	$3,137.25	10%	$817.68	5%
Dripper	$575.75	2%	$139.02	1%
Scale	$3,801.00	13%	$2,910.29	19%
Kettle	$756.25	2%	$617.52	4%
Espresso Machine	$8,406.00	28%	$3,636.44	24%
Total	**$30,284.25**	**100%**	**$14,932.16**	**100%**

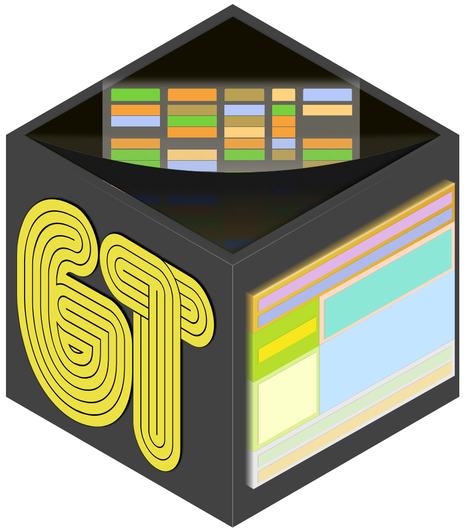
gt / Great Tables

Sales of Coffee Equipment					
	Revenue		Profit		
Product	Amount	Percent	Amount	Percent	Monthly Revenue
Grinder	$904,500	3%	$567,960	4%	
Moka pot	$2,045,250	7%	$181,080	1%	
Cold brew	$288,750	1%	$241,770	2%	
Filter	$404,250	1%	$70,010	0%	
Drip machine	$2,632,000	9%	$1,374,450	9%	
AeroPress	$2,601,500	9%	$1,293,780	9%	
Pour over	$846,000	3%	$364,530	2%	
French press	$1,113,250	4%	$748,120	5%	
Cezve	$2,512,500	9%	$1,969,520	13%	
Chemex	$3,137,250	11%	$817,680	6%	
Scale	$3,801,000	13%	$2,910,290	20%	
Kettle	$756,250	3%	$617,520	4%	
Espresso Machine	$8,406,000	29%	$3,636,440	25%	
Total	**$29,448,500**	**100%**	**$14,793,150**	**100%**	

Using **gt** or **Great Tables**, you work entirely in R or Python!

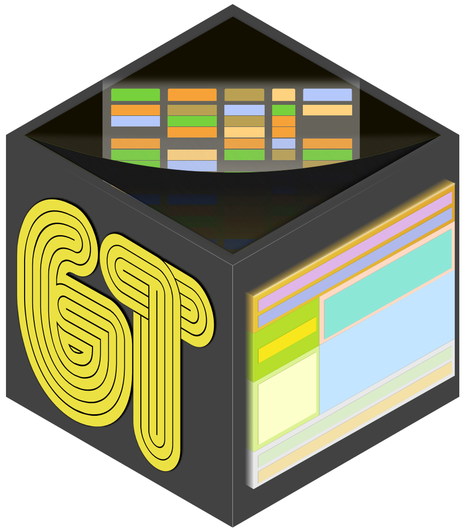
It's reproducible, less effort, and the tables look great!

gt and **Great Tables**



Both **gt** and **Great Tables** are focused purely on the display of tables.

gt and **Great Tables**

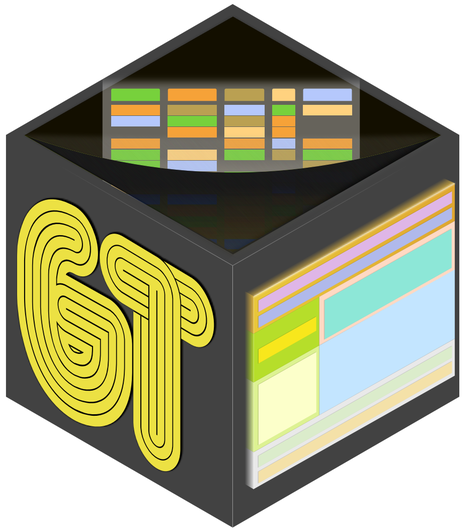


Both **gt** and **Great Tables** are focused purely on the display of tables.

They are not the only approach in their respective language, but they are:

— comprehensive
— actively-developed
— attentive to all table-related problems

gt and **Great Tables**



Both **gt** and **Great Tables** are focused purely on the display of tables.

They are not the only approach in their respective language, but they are:

— comprehensive
— actively-developed
— attentive to all table-related problems

We'll use both packages in this workshop so you can become proficient in either one, or both!

You'll learn the process and design behind making presentation-quality tables.

SETUP

Session 1: **Coffee Table**

Let's Learn the Whole Game of Making Tables with **Coffee Table**

Sales of Coffee Equipment				
	Revenue		Profit	
Product	Amount	Percent	Amount	Percent
Grinder	$904,500	3%	$567,960	4%
Moka pot	$2,045,250	7%	$181,080	1%
Cold brew	$288,750	1%	$241,770	2%
Filter	$404,250	1%	$70,010	0%
Drip machine	$2,632,000	9%	$1,374,450	9%
AeroPress	$2,601,500	9%	$1,293,780	9%
Pour over	$846,000	3%	$364,530	2%
French press	$1,113,250	4%	$748,120	5%
Cezve	$2,512,500	9%	$1,969,520	13%
Chemex	$3,137,250	11%	$817,680	6%
Scale	$3,801,000	13%	$2,910,290	20%
Kettle	$756,250	3%	$617,520	4%
Espresso Machine	$8,406,000	29%	$3,636,440	25%
Total	**$29,448,500**	**100%**	**$14,793,150**	**100%**

Key Ingredients: Structure, Format, Style

Sales of Coffee Equipment				
	Revenue		Profit	
Product	Amount	Percent	Amount	Percent
Grinder	$904,500	3%	$567,960	4%
Moka pot	$2,045,250	7%	$181,080	1%
Cold brew	$288,750	1%	$241,770	2%
Filter	$404,250	1%	$70,010	0%
Drip machine	$2,632,000	9%	$1,374,450	9%
AeroPress	$2,601,500	9%	$1,293,780	9%
Pour over	$846,000	3%	$364,530	2%
French press	$1,113,250	4%	$748,120	5%
Cezve	$2,512,500	9%	$1,969,520	13%
Chemex	$3,137,250	11%	$817,680	6%
Scale	$3,801,000	13%	$2,910,290	20%
Kettle	$756,250	3%	$617,520	4%
Espresso Machine	$8,406,000	29%	$3,636,440	25%
Total	**$29,448,500**	**100%**	**$14,793,150**	**100%**

Title.

Column spanners.

Nice column labels.

STRUCTURE

Key Ingredients: Structure, Format, Style

Sales of Coffee Equipment				
	Revenue		Profit	
Product	Amount	Percent	Amount	Percent
Grinder	$904,500	3%	$567,960	4%
Moka pot	$2,045,250	7%	$181,080	1%
Cold brew	$288,750	1%	$241,770	2%
Filter	$404,250	1%	$70,010	0%
Drip machine	$2,632,000	9%	$1,374,450	9%
AeroPress	$2,601,500	9%	$1,293,780	9%
Pour over	$846,000	3%	$364,530	2%
French press	$1,113,250	4%	$748,120	5%
Cezve	$2,512,500	9%	$1,969,520	13%
Chemex	$3,137,250	11%	$817,680	6%
Scale	$3,801,000	13%	$2,910,290	20%
Kettle	$756,250	3%	$617,520	4%
Espresso Machine	$8,406,000	29%	$3,636,440	25%
Total	**$29,448,500**	**100%**	**$14,793,150**	**100%**

Title.

Column spanners.

Nice column labels.

STRUCTURE

Currency values.

Percentages.

FORMAT

Key Ingredients: Structure, Format, Style

Sales of Coffee Equipment				
	Revenue		Profit	
Product	Amount	Percent	Amount	Percent
Grinder	$904,500	3%	$567,960	4%
Moka pot	$2,045,250	7%	$181,080	1%
Cold brew	$288,750	1%	$241,770	2%
Filter	$404,250	1%	$70,010	0%
Drip machine	$2,632,000	9%	$1,374,450	9%
AeroPress	$2,601,500	9%	$1,293,780	9%
Pour over	$846,000	3%	$364,530	2%
French press	$1,113,250	4%	$748,120	5%
Cezve	$2,512,500	9%	$1,969,520	13%
Chemex	$3,137,250	11%	$817,680	6%
Scale	$3,801,000	13%	$2,910,290	20%
Kettle	$756,250	3%	$617,520	4%
Espresso Machine	$8,406,000	29%	$3,636,440	25%
Total	**$29,448,500**	**100%**	**$14,793,150**	**100%**

Title.

Column spanners.

Nice column labels.

STRUCTURE

Currency values.

Percentages.

FORMAT

Fill color.

Bold text.

STYLE

Let’s Talk Imports/Setup: Python

product	revenue_dollars	revenue_pct	profit_dollars	profit_pct
Grinder	904500.0	0.03	567960.0	0.04
Moka pot	2045250.0	0.07	181080.0	0.01
Cold brew	288750.0	0.01	241770.0	0.02
Filter	404250.0	0.01	70010.0	0.0
Drip machine	2632000.0	0.09	1374450.0	0.09
AeroPress	2601500.0	0.09	1293780.0	0.09
Pour over	846000.0	0.03	364530.0	0.02
French press	1113250.0	0.04	748120.0	0.05
Cezve	2512500.0	0.09	1969520.0	0.13
Chemex	3137250.0	0.11	817680.0	0.06
Scale	3801000.0	0.13	2910290.0	0.2
Kettle	756250.0	0.03	617520.0	0.04
Espresso Machine	8406000.0	0.29	3636440.0	0.25
Total	29448500.0	1.02	14793150.0	1.0

```
from great_tables import GT

import polars
import polars.selectors as cs
```

```
cs.starts_with("revenue")

sel_rev = cs.starts_with("revenue")
```

Polars selectors let us target columns, and we need exactly that sort of thing for **Great Tables**.

Let’s Talk Imports/Setup: R

product	revenue_dollars	revenue_pct	profit_dollars	profit_pct
Grinder	904500.0	0.03	567960.0	0.04
Moka pot	2045250.0	0.07	181080.0	0.01
Cold brew	288750.0	0.01	241770.0	0.02
Filter	404250.0	0.01	70010.0	0.0
Drip machine	2632000.0	0.09	1374450.0	0.09
AeroPress	2601500.0	0.09	1293780.0	0.09
Pour over	846000.0	0.03	364530.0	0.02
French press	1113250.0	0.04	748120.0	0.05
Cezve	2512500.0	0.09	1969520.0	0.13
Chemex	3137250.0	0.11	817680.0	0.06
Scale	3801000.0	0.13	2910290.0	0.2
Kettle	756250.0	0.03	617520.0	0.04
Espresso Machine	8406000.0	0.29	3636440.0	0.25
Total	29448500.0	1.02	14793150.0	1.0

```
library(gt)
library(tidyverse)
```

```
starts_with("revenue")
```

We will use tidyselect-style statements in **gt** for targeting columns.

Session 1: Structure Basics

Structure: `gt()` or `GT()` — How to Begin with the GT API

We first need to introduce our data to **gt** or **Great Tables**.

For R that could be a data frame or tibble.
For Python, a Pandas or a Polars DataFrame.

gt – R

```
gt(<data>)
```

Great Tables – Python

```
GT(<data>)
```

Structure: `gt()` or `GT()` — How to Begin with the GT API

Let's use an example dataset, `exibble`, to make a **gt** and **Great Tables** table:

gt – R

```
library(gt)

gt(exibble)
```

Great Tables – Python

```
from great_tables import GT, exibble

GT(exibble)
```

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

Structure: `gt()` or `GT()` — Making a Stub with Row Labels

See this column called `row`? It contains row labels and we could structure this table with a stub (holds row 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

Structure: `gt()` or `GT()` — Making a Stub with Row Labels

Both `gt()` and `GT()` have the `rowname_col` arg. Supply the column name there and it moves into the stub.

gt – R

```
library(gt)

gt(exibble, rowname_col = "row")
```

Great Tables – Python

```
from great_tables import GT, exibble

GT(exibble, rowname_col="row")
```

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

Structure: `gt()` or `GT()` — Making Row Groups

See this column called `group`? It contains categories for grouping rows, so and we could add row groups to the table.

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

Structure: `gt()` or `GT()` — Making Row Groups

Use the `groupname_col` arg to define row groups with a column.

gt – R

```
gt(
  exibble,
  rowname_col = "row",
  groupname_col = "group"
)
```

Great Tables – Python

```
GT(
  exibble,
  rowname_col="row",
  groupname_col="group"
)
```

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

Structure: Comparing a Basic and a More Structured Table

Basic GT Table

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

Table w/ Stub & Row Groups

	num	char	fctr	date
grp_a				
row_1	1.111e-01	apricot	one	2015-01-15
row_2	2.222e+00	banana	two	2015-02-15
row_3	3.333e+01	coconut	three	2015-03-15
row_4	4.444e+02	durian	four	2015-04-15
grp_b				
row_5	5.550e+03	NA	five	2015-05-15
row_6	NA	fig	six	2015-06-15
row_7	7.770e+05	grapefruit	seven	NA
row_8	8.880e+06	honeydew	eight	2015-08-15

Structure: `tab_header()` — Adding a Title to Your Table

Adding a title to the GT table in a header component can be good for presentation. We do that with `tab_header()`.

gt – R

```
gt(exibble) |>
  tab_header(title = "Table Title")
```

Great Tables – Python

```
GT(exibble) \
.tab_header(title="Table Title")
```

Table Title

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

Structure: `tab_header()` — You Can Also Add a Subtitle

Having a subtitle is also an option here.

gt – R

```
gt(exibble) |>
  tab_header(
    title = "Table Title",
    subtitle = "The Subtitle"
  )
```

Great Tables – Python

```
GT(exibble) \
.tab_header(
  title="Table Title",
  subtitle="The Subtitle"
)
```

Table Title

The Subtitle

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

Structure: `tab_spanner()` — Spanners Above Column Labels

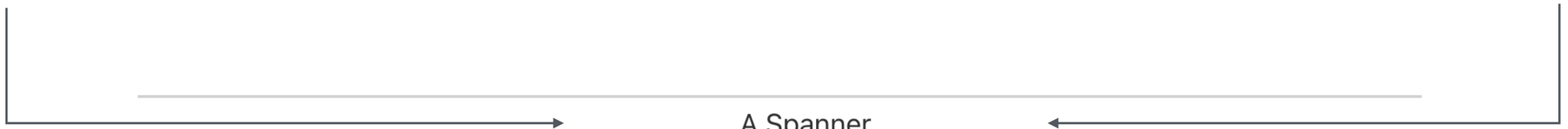
Another way to add structure is to add spanner labels above sets of column labels. We do this with `tab_spanner()`.

gt – R

```
gt(exibble) |>
  tab_spanner(
    columns = c(date, time, datetime),
    label = "A Spanner"
  )
```

Great Tables – Python

```
GT(exibble) \
.tab_spanner(
  columns=["date", "time", "datetime"],
  label="A Spanner"
)
```



			A Spanner					
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

Structure: `cols_label()` — Making Column Labels Much Nicer

The column labels are derived from the column names. We usually want to make them more presentable and it's done with `cols_label()`.

gt – R

```
gt(exibble) |>
  cols_label(
    num = "Numbers",
    char = "Fruits"
  )
```

Great Tables – Python

```
GT(exibble) \
.cols_label(
  num="Numbers",
  char="Fruits"
)
```

Numbers	Fruits	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

Let's Try It Out!

py-01-coffee-table.qmd

```
## Start

## Structure Basics
```

Session 1: Formatting Basics

Format: the family of `fmt_*()` Functions

There is a huge number of formatting functions!

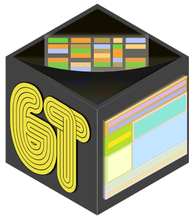
They all begin with `fmt_` and they format values in the table body.

They they typically operate on whole columns of data but
you can subset the columns' rows.

Here's a sampling of what is available:

```
fmt_number()
fmt_integer()
fmt_scientific()
fmt_engineering()
fmt_percent()
fmt_currency()
fmt_date()
fmt_time()
fmt_datetime()
fmt_markdown()
fmt_image()
fmt()
```

\+ many more.
Especially in:



Format: `fmt_currency()` — Formatting Monetary Values

Let's again use the example dataset, **`exibble`**, and see what we're starting with.

gt – R

```
library(gt)

gt(exibble)
```

Great Tables – Python

```
from great_tables import GT, exibble

GT(exibble)
```

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

Format: `fmt_currency()` — Formatting Monetary Values

There's a column called `currency` here. Let's format that with `fmt_currency()`

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

Format: `fmt_currency()` — Formatting Monetary Values

We've got to specify the columns here when using any formatter. In this case it is the one column called `currency`.

gt – R

```
gt(exibble) |>
  fmt_currency(columns = "currency")
```

Great Tables – Python

```
GT(exibble) \
.fmt_currency(columns="currency")
```

num	char	fctr	date	time	datetime	currency	row	group
1.111e-01	apricot	one	2015-01-15	13:35	2018-01-01 02:22	$49.95	row_1	grp_a
2.222e+00	banana	two	2015-02-15	14:40	2018-02-02 14:33	$17.95	row_2	grp_a
3.333e+01	coconut	three	2015-03-15	15:45	2018-03-03 03:44	$1.39	row_3	grp_a
4.444e+02	durian	four	2015-04-15	16:50	2018-04-04 15:55	$65,100.00	row_4	grp_a
5.550e+03	NA	five	2015-05-15	17:55	2018-05-05 04:00	$1,325.81	row_5	grp_b
NA	fig	six	2015-06-15	NA	2018-06-06 16:11	$13.26	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	$0.44	row_8	grp_b

Format: `fmt_currency()` — Formatting Monetary Values

Before Formatting

currency
49.950
17.950
1.390
65100.000
1325.810
13.255
NA
0.440

After Formatting

currency
$49.95
$17.95
$1.39
$65,100.00
$1,325.81
$13.26
NA
$0.44

What changes can you see in the *before* and *after* of this formatting?

Format: `fmt_percent()` — Formatting as Percentages

There's not really a great column to demonstrate percentage formatting in `exibble`.
Let's make up a new table with five rows:

row	value
1	0.0200
2	0.4345
3	0.0520
4	0.7530
5	1.0234

Format: `fmt_percent()` — Formatting as Percentages

We will format that called `value` column with `fmt_percent()`. Let's use the defaults.

gt – R

```
gt(<data>) |>
  fmt_percent(columns = "value")
```

Great Tables – Python

```
GT(<data>) \
.fmt_percent(columns="value")
```

row	value
1	2.00%
2	43.45%
3	5.20%
4	75.30%
5	102.34%

Format: `fmt_percent()` — Formatting as Percentages

Before Formatting

row	value
1	0.0200
2	0.4345
3	0.0520
4	0.7530
5	1.0234

After Formatting

row	value
1	2.00%
2	43.45%
3	5.20%
4	75.30%
5	102.34%

What changes can you see in the *before* and *after* of this formatting?

Let's Try It Out!

py-01-coffee-table.qmd

```
## Formatting Basics
```

Session 1: Styling with `tab_style()`

Style: `tab_style()` — Styling the Table Cells

The `tab_style()` function is a bit more complicated than the previous ones we tried out.

gt – R

```
library(gt)

gt(<data>) |>
  tab_style(
    style = ...,
    locations = ...
  )
```

Great Tables – Python

```
from great_tables import GT

GT(<data>) \
.tab_style(
  style=...,
  locations=...
)
```

The additional complication is in the `...` parts. They require the use of helper functions.

Style: `tab_style()` — Styling the Table Cells

There are two arguments in `tab_style()`: `style` and `locations`.

gt – R

```
library(gt)

gt(<data>) |>
  tab_style(
    style = ...,
    locations = ...
  )
```

what → `style = ...,`

where → `locations = ...`

Great Tables – Python

```
from great_tables import GT

GT(<data>) \
.tab_style(
  style=...,
  locations=...
)
```

what → `style=...,`

where → `locations=...`

Guide to args:

`style`: what is the styling we are going to use?

`locations`: where is the styling going to be used? Or, which cells receive the styles?

Style: `tab_style()` — Styling the Table Cells

Helpers used to define the `style` and `locations`.

gt – R

`style` use `cell_*()` functions

- `cell_fill()`
- `cell_text()`
- `cell_borders()`

`locations` use `cells_*()` functions

- `cells_body()`
- `cells_stub()`
- `cells_column_labels()` + many more.

Before Styling

num	char
1.111e-01	apricot
2.222e+00	banana
3.333e+01	coconut
4.444e+02	durian
5.550e+03	NA
NA	fig
7.770e+05	grapefruit
8.880e+06	honeydew

After Styling ×2

num	char
1.111e-01	apricot
2.222e+00	banana
3.333e+01	coconut
4.444e+02	durian
5.550e+03	NA
NA	fig
7.770e+05	grapefruit
8.880e+06	honeydew

Style: `tab_style()` — Styling the Table Cells

These helpers are analogous but named differently in **gt** and **Great Tables**.

gt – R

`style` use `cell_*()` functions

- `cell_fill()`
- `cell_text()`
- `cell_borders()`

`locations` use `cells_*()` functions

- `cells_body()`
- `cells_stub()`
- `cells_column_labels()` + many more.

Great Tables – Python

`style` use `style` class

- `style.fill()`
- `style.text()`
- `style.borders()`

`locations` use `loc` class

- `loc.body()`

(Others in development.)

Style: `tab_style()` — Styling the Table Cells

IMPORTANT: for **Great Tables**, we need two extra imports if using `.tab_style()`.

Great Tables – Python

```
from great_tables import GT, style, loc

GT(<data>) \
.tab_style(
  style = style...,
  locations = loc...
)
```

Importing `style` and `loc` is important here. Otherwise, you can't use `tab_style()`.

Style: `tab_style()` — Styling the Table Cells

We will go back to the baseline table (using `exibble` for this).

gt – R

```
library(gt)

gt(exibble)
```

Great Tables – Python

```
from great_tables import GT, exibble

GT(exibble)
```

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

Style: `tab_style()` — Styling the Table Cells

Let's style the entire **num** column with a light blue background color.

gt – R

```
gt(exibble) |>
  tab_style(
    style = cell_fill(color = "lightblue"),
    locations = cells_body(columns = num)
  )
```

Great Tables – Python

```
GT(exibble) \
.tab_style(
  style=style.fill(color="lightblue"),
  locations=loc.body(columns="num")
)
```

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

Let's Try It Out!

py-01-coffee-table.qmd

```
## Styling with tab_style()
```

Session 1: Three Last Things

Three Last Things: Nanoplots, Images, and Missing Values

After we do all these things, we'll get this table:

added icons

`fmt_image()`

`sub_missing()`

addressed missing val

added nanoplots

`fmt_nanoplot()`

Sales of Coffee Equipment

	Product	Revenue		Profit		
		Amount	Percent	Amount	Percent	Monthly Revenue
	Grinder	$904,500	3%	$567,960	4%	
	Moka pot	$2,045,250	7%	$181,080	1%	
	Cold brew	$288,750	1%	$241,770	2%	
	Filter	$404,250	1%	$70,010	0%	
	Drip machine	$2,632,000	9%	$1,374,450	9%	
	AeroPress	$2,601,500	9%	$1,293,780	9%	
	Pour over	$846,000	3%	$364,530	2%	
	French press	$1,113,250	4%	$748,120	5%	
	Cezve	$2,512,500	9%	$1,969,520	13%	
	Chemex	$3,137,250	11%	$817,680	6%	
	Scale	$3,801,000	13%	$2,910,290	20%	
	Kettle	$756,250	3%	$617,520	4%	
	Espresso Machine	$8,406,000	29%	$3,636,440	25%	
	Total	**$29,448,500**	**100%**	**$14,793,150**	**100%**	

Let's Try It Out!

py-01-coffee-table.qmd

```
## Three Last Things
```

Session 2: **Reactions Table**

This Is What the **Reactions Table** Will Look Like in the End

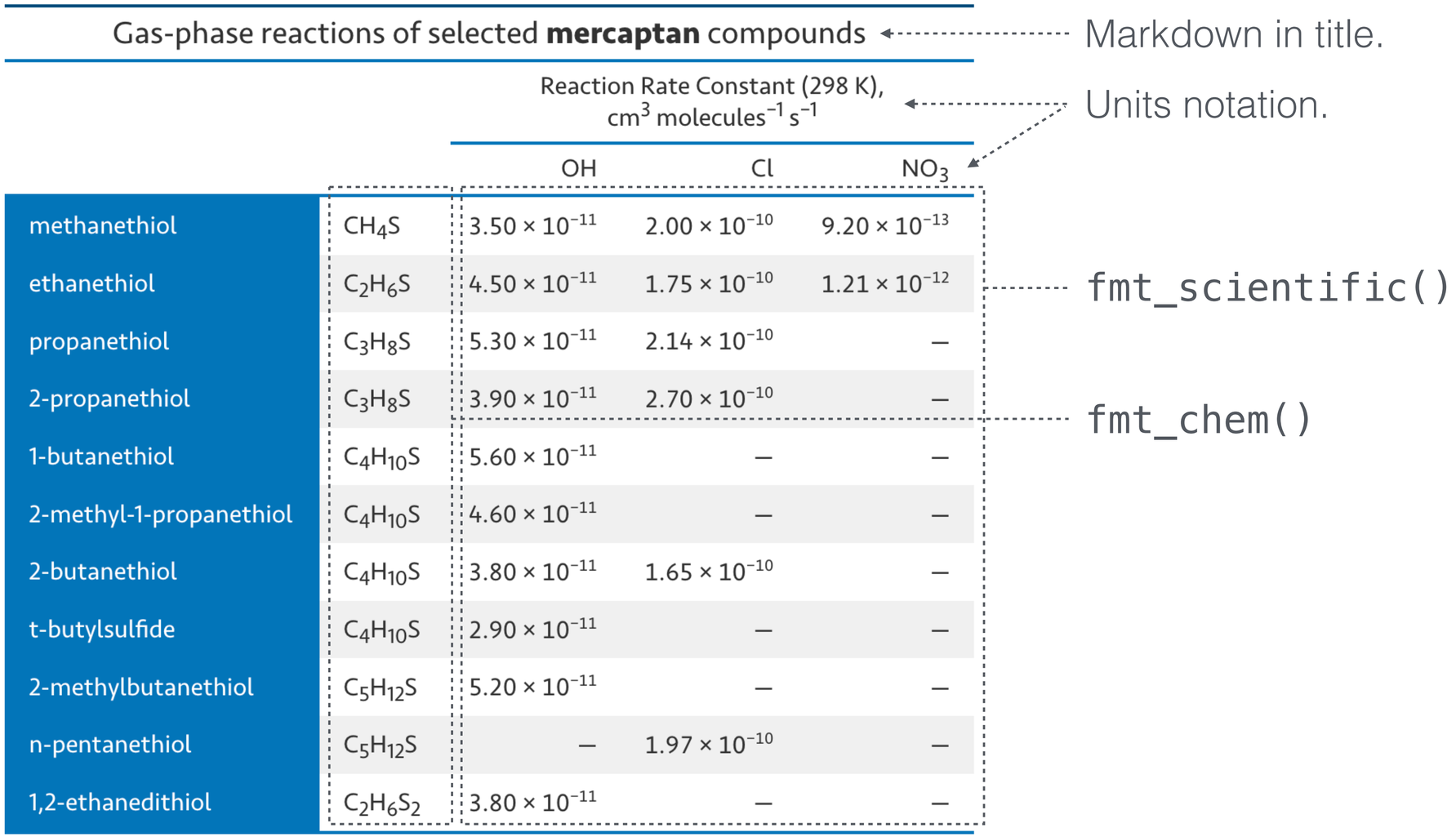
Gas-phase reactions of selected **mercaptan** compounds				
		Reaction Rate Constant (298 K), cm^3 $molecules^{-1}$ s^{-1}		
		OH	Cl	NO_3
methanethiol	CH_4S	3.50×10^{-11}	2.00×10^{-10}	9.20×10^{-13}
ethanethiol	C_2H_6S	4.50×10^{-11}	1.75×10^{-10}	1.21×10^{-12}
propanethiol	C_3H_8S	5.30×10^{-11}	2.14×10^{-10}	–
2-propanethiol	C_3H_8S	3.90×10^{-11}	2.70×10^{-10}	–
1-butanethiol	$C_4H_{10}S$	5.60×10^{-11}	–	–
2-methyl-1-propanethiol	$C_4H_{10}S$	4.60×10^{-11}	–	–
2-butanethiol	$C_4H_{10}S$	3.80×10^{-11}	1.65×10^{-10}	–
t-butylsulfide	$C_4H_{10}S$	2.90×10^{-11}	–	–
2-methylbutanethiol	$C_5H_{12}S$	5.20×10^{-11}	–	–
n-pentanethiol	$C_5H_{12}S$	–	1.97×10^{-10}	–
1,2-ethanedithiol	$C_2H_6S_2$	3.80×10^{-11}	–	–

Key Ingredients: Structure, Format, Style

Gas-phase reactions of selected **mercaptan** compounds ← Markdown in title.

		Reaction Rate Constant (298 K), cm^3 $molecules^{-1}$ s^{-1} ← Units notation.		
		OH	Cl	NO_3
methanethiol	CH_4S	3.50×10^{-11}	2.00×10^{-10}	9.20×10^{-13}
ethanethiol	C_2H_6S	4.50×10^{-11}	1.75×10^{-10}	1.21×10^{-12}
propanethiol	C_3H_8S	5.30×10^{-11}	2.14×10^{-10}	–
2-propanethiol	C_3H_8S	3.90×10^{-11}	2.70×10^{-10}	–
1-butanethiol	$C_4H_{10}S$	5.60×10^{-11}	–	–
2-methyl-1-propanethiol	$C_4H_{10}S$	4.60×10^{-11}	–	–
2-butanethiol	$C_4H_{10}S$	3.80×10^{-11}	1.65×10^{-10}	–
t-butylsulfide	$C_4H_{10}S$	2.90×10^{-11}	–	–
2-methylbutanethiol	$C_5H_{12}S$	5.20×10^{-11}	–	–
n-pentanethiol	$C_5H_{12}S$	–	1.97×10^{-10}	–
1,2-ethanedithiol	$C_2H_6S_2$	3.80×10^{-11}	–	–

Key Ingredients: Structure, Format, Style



Gas-phase reactions of selected **mercaptan** compounds

Markdown in title.

Reaction Rate Constant (298 K), $cm^3\ molecules^{-1}\ s^{-1}$

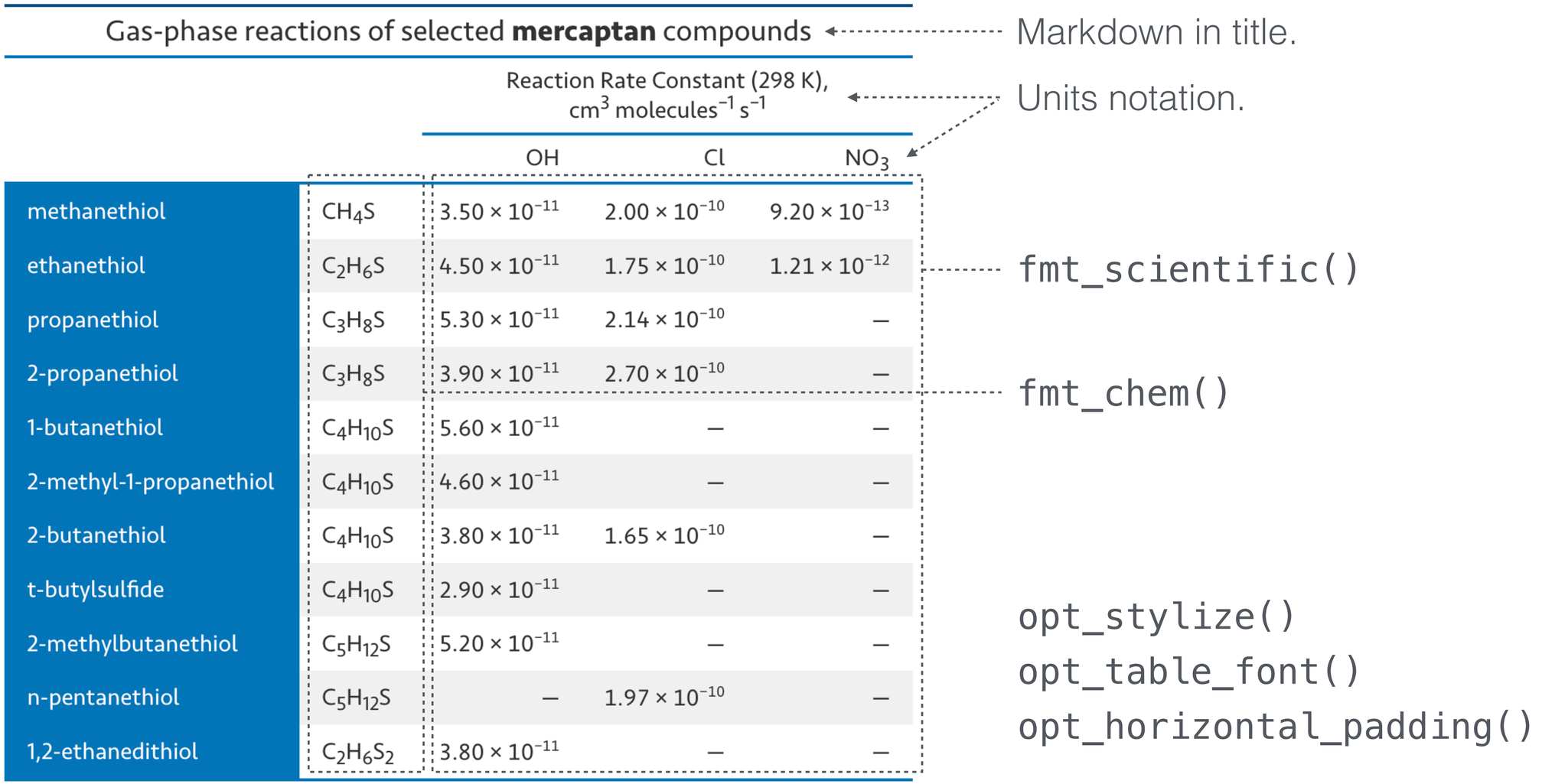
Units notation.

		OH	Cl	NO_3
methanethiol	CH_4S	3.50×10^{-11}	2.00×10^{-10}	9.20×10^{-13}
ethanethiol	C_2H_6S	4.50×10^{-11}	1.75×10^{-10}	1.21×10^{-12}
propanethiol	C_3H_8S	5.30×10^{-11}	2.14×10^{-10}	–
2-propanethiol	C_3H_8S	3.90×10^{-11}	2.70×10^{-10}	–
1-butanethiol	$C_4H_{10}S$	5.60×10^{-11}	–	–
2-methyl-1-propanethiol	$C_4H_{10}S$	4.60×10^{-11}	–	–
2-butanethiol	$C_4H_{10}S$	3.80×10^{-11}	1.65×10^{-10}	–
t-butylsulfide	$C_4H_{10}S$	2.90×10^{-11}	–	–
2-methylbutanethiol	$C_5H_{12}S$	5.20×10^{-11}	–	–
n-pentanethiol	$C_5H_{12}S$	–	1.97×10^{-10}	–
1,2-ethanedithiol	$C_2H_6S_2$	3.80×10^{-11}	–	–

`fmt_scientific()`

`fmt_chem()`

Key Ingredients: Structure, Format, Style



Gas-phase reactions of selected **mercaptan** compounds

Markdown in title.

Reaction Rate Constant (298 K), $cm^3\ molecules^{-1}\ s^{-1}$

Units notation.

		OH	Cl	NO_3
methanethiol	CH_4S	3.50×10^{-11}	2.00×10^{-10}	9.20×10^{-13}
ethanethiol	C_2H_6S	4.50×10^{-11}	1.75×10^{-10}	1.21×10^{-12}
propanethiol	C_3H_8S	5.30×10^{-11}	2.14×10^{-10}	—
2-propanethiol	C_3H_8S	3.90×10^{-11}	2.70×10^{-10}	—
1-butanethiol	$C_4H_{10}S$	5.60×10^{-11}	—	—
2-methyl-1-propanethiol	$C_4H_{10}S$	4.60×10^{-11}	—	—
2-butanethiol	$C_4H_{10}S$	3.80×10^{-11}	1.65×10^{-10}	—
t-butylsulfide	$C_4H_{10}S$	2.90×10^{-11}	—	—
2-methylbutanethiol	$C_5H_{12}S$	5.20×10^{-11}	—	—
n-pentanethiol	$C_5H_{12}S$	—	1.97×10^{-10}	—
1,2-ethanedithiol	$C_2H_6S_2$	3.80×10^{-11}	—	—

`fmt_scientific()`

`fmt_chem()`

`opt_stylize()`
`opt_table_font()`
`opt_horizontal_padding()`

Session 2: Introducing Units Notation

Structure: Adding Measurement Units to the Table

Tables are often full of values... but we also need to know what the values signify.

It's common to provide the *measurement units* of values.
We see this both in tables and plots. Let's look at some table examples.

name	Population	Density, *persons* km^{-2}	Area, km^2	Latitude, °N	Longitude, °W
Toronto	2,794,356	4,427.8	631.1	43.74	79.37
Ottawa	1,017,449	364.9	2,788.2	45.42	75.69
Mississauga	717,961	2,452.6	292.7	43.60	79.65
Brampton	656,480	2,469.0	265.9	43.69	79.76
Hamilton	569,353	509.1	1,118.3	43.26	79.87
London	422,324	1,004.3	420.5	42.97	81.23
Markham	338,503	1,604.8	210.9	43.88	79.26
Vaughan	323,103	1,186.0	272.4	43.83	79.50
Kitchener	256,885	1,877.7	136.8	43.42	80.47
Windsor	229,660	1,572.8	146.0	42.28	83.00

Laboratory Findings for the YF Patient

		Day			
Test	Units	3	4	5	6
Viral load	copies per mL	12,000	4,200	1,600	830
WBC	$\times 10^9$/L	5.26	4.26	9.92	10.49
Neutrophils	$\times 10^9$/L	4.87	4.72	7.92	18.21
RBC	$\times 10^{12}$/L	5.72	5.98	4.23	4.83
Hb	g/L	153	135	126	115
PLT	$\times 10^9$/L	67	38.6	27.4	26.2
ALT	U/L	12,835	12,632	6,426.7	4,263.1
AST	U/L	23,672	21,368	14,730	8,691
TBIL	µmol/L	117.2	143.8	137.2	158.1
DBIL	µmol/L	71.4	104.6	94.6	143.9
NH3	mmol/L	115.2	135.2	131	176.7
PT	s	24.6	42.4	53.7	54
APTT	s	39.2	57.2	65.9	68.3
PTA	%	41	25	19	14
DD	mg/L	32.9	35.1	24.5	25.6
FDP	µg/mL	84.7	92.5	77.2	—
Fibrinogen	mg/dL	238.1	216.8	135	85.2
LDH	U/L	5,727.3	2,622.8	2,418.7	546.3

Physical Constants Having a Molar Basis	
alpha particle molar mass	4.002×10^{-3} kg mol^{-1}
deuteron molar mass	2.014×10^{-3} kg mol^{-1}
electron molar mass	5.486×10^{-7} kg mol^{-1}
helion molar mass	3.015×10^{-3} kg mol^{-1}
molar gas constant	8.314 J mol^{-1} K^{-1}
molar mass constant	1.000×10^{-3} kg mol^{-1}
molar mass of carbon-12	1.200×10^{-2} kg mol^{-1}
molar Planck constant	3.990×10^{-10} J Hz^{-1} mol^{-1}

Structure: Adding Measurement Units to the Table

Laboratory Findings for the YF Patient

		Day			
Test	Units	3	4	5	6
Viral load	copies per mL	12,000	4,200	1,600	830
WBC	$\times 10^9$/L	5.26	4.26	9.92	10.49
Neutrophils	$\times 10^9$/L	4.87	4.72	7.92	18.21
RBC	$\times 10^{12}$/L	5.72	5.98	4.23	4.83
Hb	g/L	153	135	126	115
PLT	$\times 10^9$/L	67	38.6	27.4	26.2
ALT	U/L	12,835	12,632	6,426.7	4,263.1
AST	U/L	23,672	21,368	14,730	8,691
TBIL	µmol/L	117.2	143.8	137.2	158.1
DBIL	µmol/L	71.4	104.6	94.6	143.9
NH3	mmol/L	115.2	135.2	131	176.7
PT	s	24.6	42.4	53.7	54
APTT	s	39.2	57.2	65.9	68.3
PTA	%	41	25	19	14
DD	mg/L	32.9	35.1	24.5	25.6
FDP	µg/mL	84.7	92.5	77.2	—
Fibrinogen	mg/dL	238.1	216.8	135	85.2
LDH	U/L	5,727.3	2,622.8	2,418.7	546.3

Structure: Adding Measurement Units to the Table

name	Population	Density, *persons* km^{-2}	Area, km^2	Latitude, °N	Longitude, °W
Toronto	2,794,356	4,427.8	631.1	43.74	79.37
Ottawa	1,017,449	364.9	2,788.2	45.42	75.69
Mississauga	717,961	2,452.6	292.7	43.60	79.65
Brampton	656,480	2,469.0	265.9	43.69	79.76
Hamilton	569,353	509.1	1,118.3	43.26	79.87
London	422,324	1,004.3	420.5	42.97	81.23
Markham	338,503	1,604.8	210.9	43.88	79.26
Vaughan	323,103	1,186.0	272.4	43.83	79.50
Kitchener	256,885	1,877.7	136.8	43.42	80.47
Windsor	229,660	1,572.8	146.0	42.28	83.00

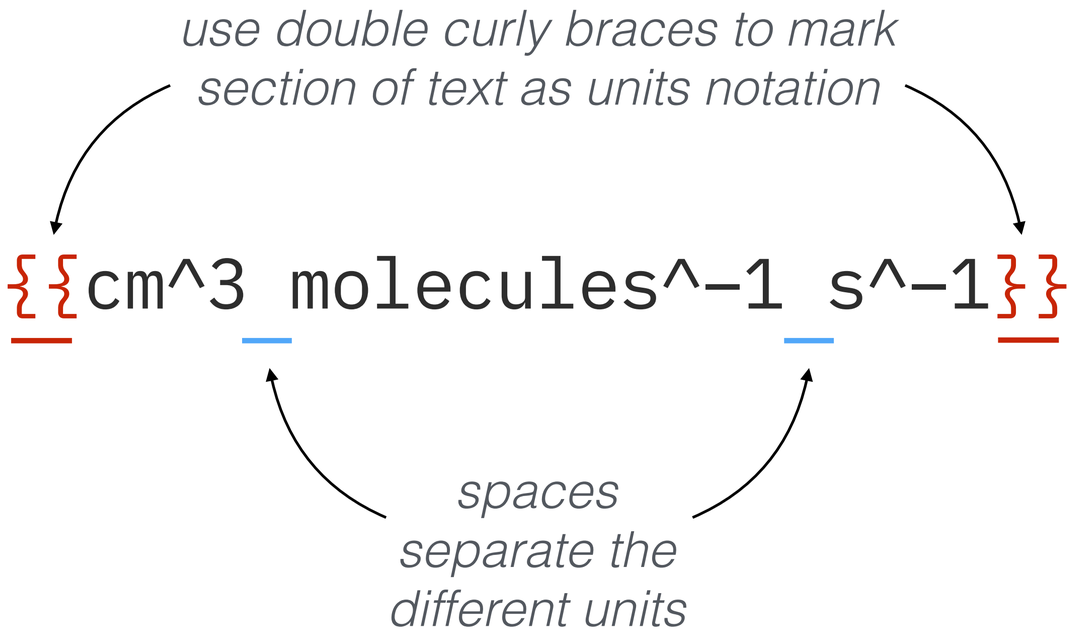
Structure: Adding Measurement Units to the Table

Physical Constants Having a Molar Basis		
alpha particle molar mass	4.002×10^{-3}	$\mathrm{kg\ mol^{-1}}$
deuteron molar mass	2.014×10^{-3}	$\mathrm{kg\ mol^{-1}}$
electron molar mass	5.486×10^{-7}	$\mathrm{kg\ mol^{-1}}$
helion molar mass	3.015×10^{-3}	$\mathrm{kg\ mol^{-1}}$
molar gas constant	8.314	$\mathrm{J\ mol^{-1}\ K^{-1}}$
molar mass constant	1.000×10^{-3}	$\mathrm{kg\ mol^{-1}}$
molar mass of carbon-12	1.200×10^{-2}	$\mathrm{kg\ mol^{-1}}$
molar Planck constant	3.990×10^{-10}	$\mathrm{J\ Hz^{-1}\ mol^{-1}}$

Structure: Adding Measurement Units to the Table

How to do this in **gt** and **Great Tables**? Use *units notation*. Helps you create inline units.

Let's try to understand the specialized syntax.



`cm^3` → cm^3

name exponent

`s^-1` → s^{-1}

name exponent

Some basic guidance on expressing units available at:

`https://physics.nist.gov/cuu/Units/checklist.html`

Structure: Adding Measurement Units to the Table

Basic guide to units notation

rule	input	output
'^' creates a superscript	`m^2`	m^2
'_' creates a subscript	`h_0`	h_0
subscripts and superscripts can be combined	`h_0^3`	h_0^3
use '[_subscript^superscript]' to create an overstrike	`h[_0^3]`	h_0^3
a '/' at the beginning adds the superscript '-1'	`/s`	s^{-1}
hyphen is transformed to minus sign when preceding a unit	`-h^2`	$-h^2$
'x' at the beginning is transformed to '×'	`x10^3 kg^2 m^-1`	$\times 10^3\ kg^2\ m^{-1}$
ASCII terms from biology/chemistry turned into terminology forms	`ug`	µg
can create italics with '*' or '_'; create bold text with '**' or '__'	`*m*^**2**`	$\mathit{m}^{\mathbf{2}}$
special symbol set surrounded by colons	`:permille:C`	‰C
chemistry notation: '%C6H6%'	`g/L %C6H12O6%`	g/L $C_6H_{12}O_6$

Structure: Adding Measurement Units to the Table

You can experiment with units notation! Here's how:

gt – R

```
library(gt)
library(tidyverse)

tibble(
    units = "x^-2",
    out = units
) |>
    gt() |>
    fmt_units(columns = out)
```

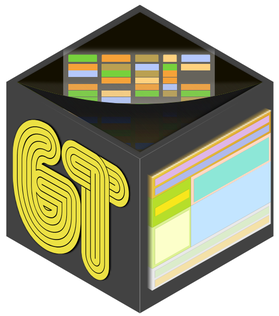
Great Tables – Python

```
from great_tables import GT, define_units

define_units("x^-2")
```

NOTE there is no dedicated function like `define_units()` in **gt** yet.

Let's Try It Out!



r-02-reactions-table.qmd

```
## Start

## Introducing Units Notation
```

Session 2: Formatting for Science with `fmt_chem()` and `fmt_scientific()`

Format: the family of `fmt_*()` Functions

From the family of formatting functions (`fmt_*()`) we will use two here:

- `fmt_chem()`: takes chemical formula text and properly formats that

  `H2O2` ⟶ H_2O_2

  `C4H10S` ⟶ $C_4H_{10}S$

- `fmt_scientific()`: takes numerical values formats as scientific notation

  `0.0002368` ⟶ 2.37×10^{-4}

  `4364000000000` ⟶ 4.36×10^{12}

  `1.5632` ⟶ 1.56×10^{0} ⟶ 1.56

  Convention is to avoid null exponents.

Format: Taking Care of Missing Values

We use the `sub_missing()` function to replace any **NA**/**None** values.

`sub_missing()` has a `columns` argument but, by default, targeting is for all columns.

gt – R

```
gt(exibble) |> sub_missing()
```

Great Tables – Python

```
GT(exibble).sub_missing()
```

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	—	five	2015-05-15	17:55	2018-05-05 04:00	1325.810	row_5	grp_b
—	fig	six	2015-06-15	—	2018-06-06 16:11	13.255	row_6	grp_b
7.770e+05	grapefruit	seven	—	19:10	2018-07-07 05:22	—	row_7	grp_b
8.880e+06	honeydew	eight	2015-08-15	20:20	—	0.440	row_8	grp_b

Format: Taking Care of Missing Values

As an alternative to the default **"—"** character, you can specify you own text for the replacement.

gt – R

```
gt(exibble) |>
  sub_missing(missing_text = "missing")
```

Great Tables – Python

```
GT(exibble) \
.sub_missing(missing_text="missing")
```

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	missing	five	2015-05-15	17:55	2018-05-05 04:00	1325.810	row_5	grp_b
missing	fig	six	2015-06-15	missing	2018-06-06 16:11	13.255	row_6	grp_b
7.770e+05	grapefruit	seven	missing	19:10	2018-07-07 05:22	missing	row_7	grp_b
8.880e+06	honeydew	eight	2015-08-15	20:20	missing	0.440	row_8	grp_b

Format: Hiding Columns from Final Output

You can elect to hide entire columns from the final output with `cols_hide()`.

Why do this? You may need a column for an expression but it's not really suitable for display.

gt – R

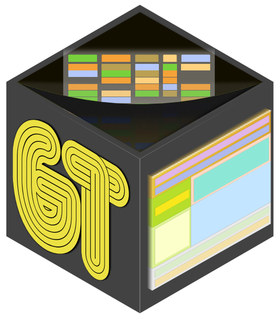
```
gt(exibble) |> cols_hide(c(date, time))
```

Great Tables – Python

```
GT(exibble).cols_hide(["date", "time"])
```

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

Let's Try It Out!



r-02-reactions-table.qmd

Formatting for Science with fmt_chem() and fmt_scientific()

Session 2: Using the `opt_*()` Functions for Quick Styling

Style: `opt_stylize()` — Complete Styling with a Theme

We can shortcut the styling of a table with `opt_stylize()`.

gt – R

```
<gt_data> |>
  opt_stylize()
```

Great Tables – Python

```
<gt_data> \
.opt_stylize()
```

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

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

Style: `opt_stylize()` — Complete Styling with a Theme

There are stylistic options (in `style`) and color possibilities (in `color`). Try them out!

gt – R

```
<gt_data> |>
  opt_stylize(style = 3, color = "green")
```

Great Tables – Python

```
<gt_data> \
.opt_stylize(style=3, color="green")
```

The title of the table

The table's subtitle

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

This is a source note.

The title of the table

The table's subtitle

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

This is a source note.

Style: `opt_table_font()` — Change the Table Font

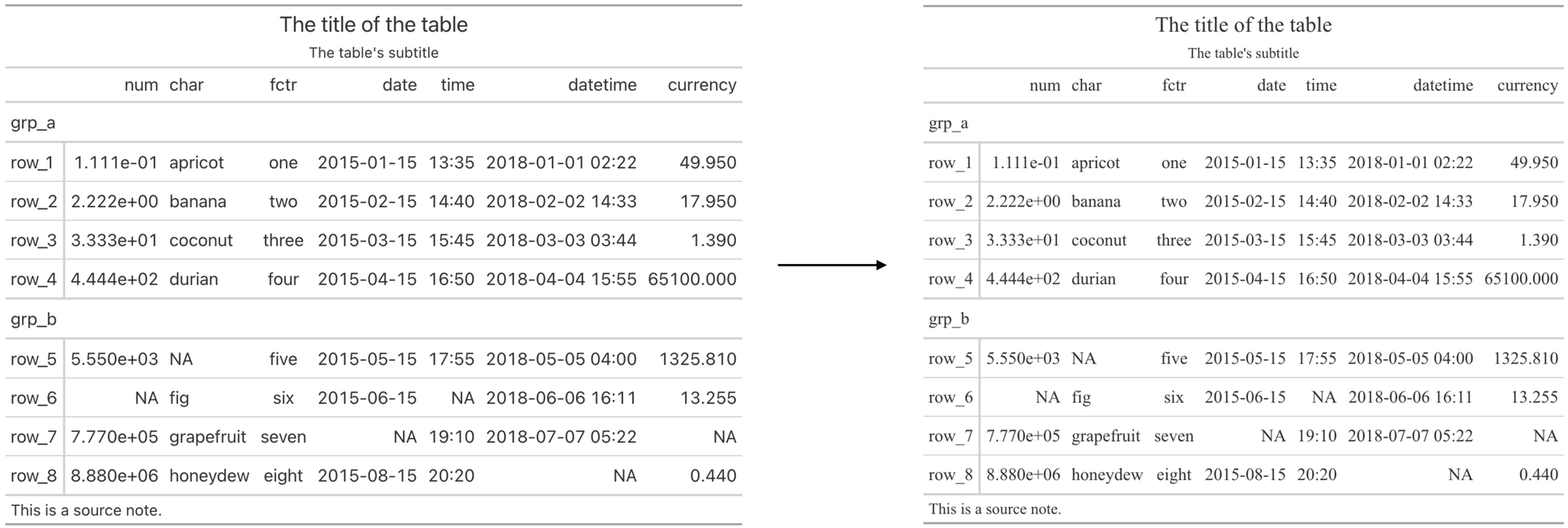
It's pretty easy to change the table's font with `opt_table_font()`.

gt – R

```
<gt_data> |>
  opt_table_font(font = "Times New Roman")
```

Great Tables – Python

```
<gt_data> \
.opt_table_font(font="Times New Roman")
```



The title of the table

The table's subtitle

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

This is a source note.

The title of the table

The table's subtitle

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

This is a source note.

Style: `opt_table_font()` — Change the Table Font

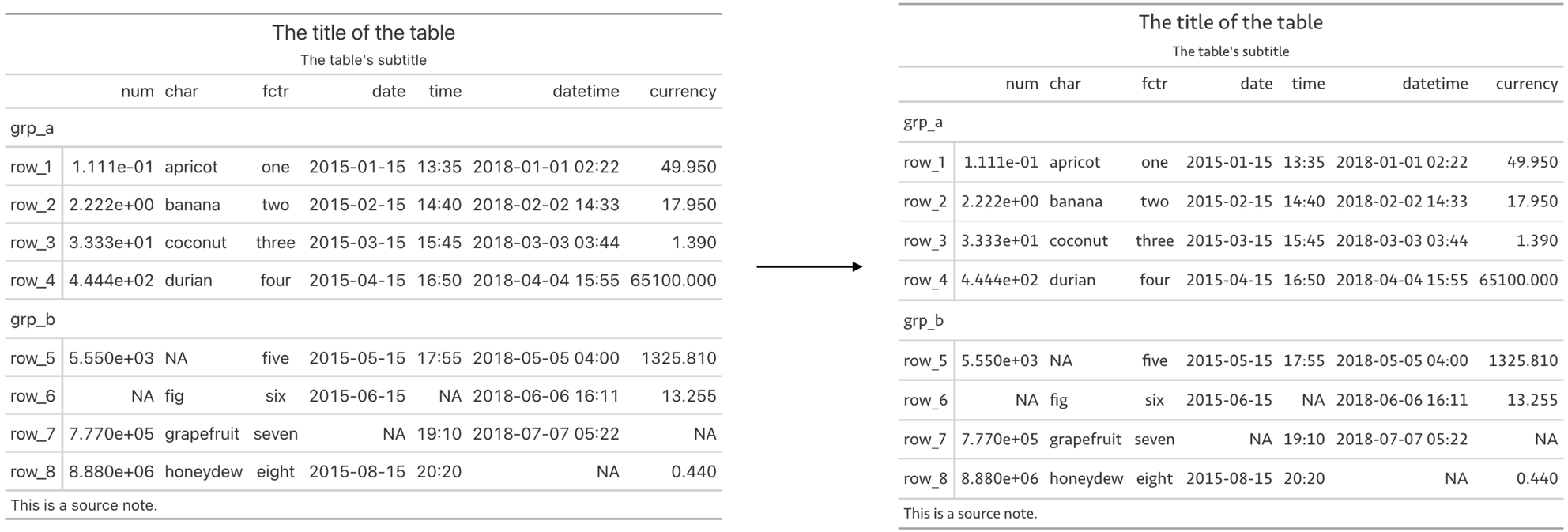
You can also specify a `stack`. This is a themed set of fonts that work well across systems.

gt – R

```
<gt_data> |>
  opt_table_font(stack = "humanist")
```

Great Tables – Python

```
<gt_data> \
.opt_table_font(stack="humanist")
```



The title of the table

The table's subtitle

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

This is a source note.

The title of the table

The table's subtitle

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

This is a source note.

Style: `opt_table_font()` — Change the Table Font

You should experiment with the different `stack` types. Here's the list of them.

```
system-ui   transitional   old-style   humanist

geometric-humanist   classical-humanist   neo-grotesque

monospace-slab-serif   monospace-code   industrial

rounded-sans   slab-serif   antique   didone   handwritten
```

For more information on these, look at the help page from `?system_fonts`.

Style: `opt_*_padding()` — Change the Table Padding

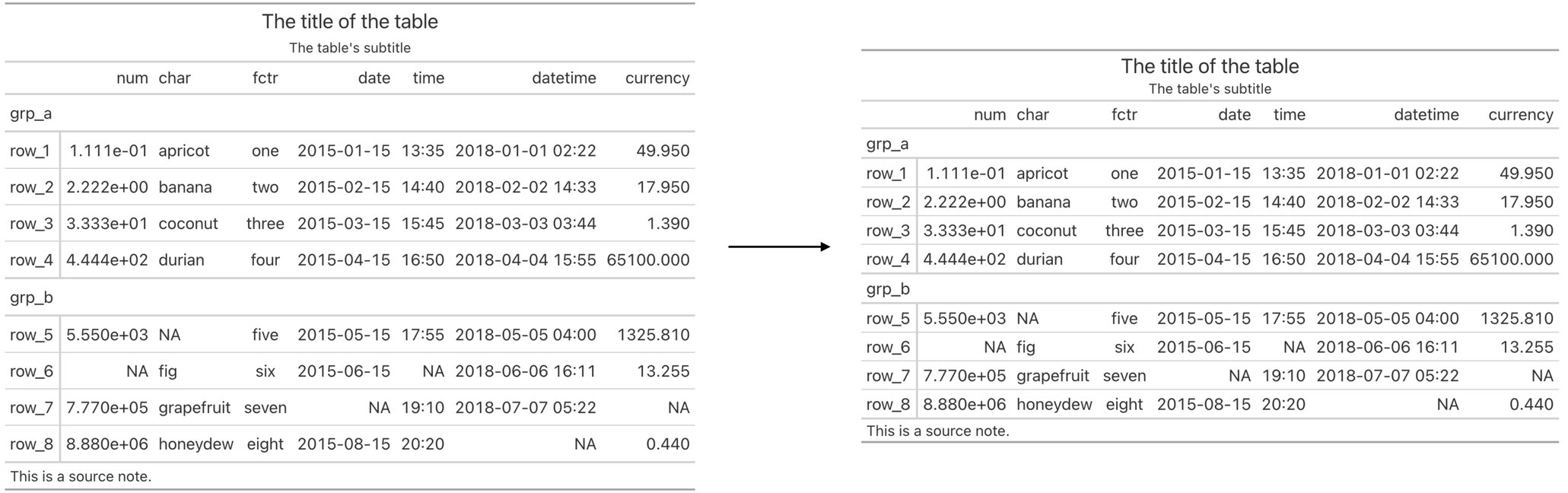
You can modify the vertical table padding with `opt_vertical_padding()`.

gt – R

```
<gt_data> |>
  opt_vertical_padding(scale = 0.5)
```

Great Tables – Python

```
<gt_data> \
.opt_vertical_padding(scale=0.5)
```



The title of the table

The table's subtitle

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

This is a source note.

The title of the table

The table's subtitle

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

This is a source note.

Style: `opt_*_padding()` — Change the Table Padding

You can likewise work in the horizontal direction with `opt_horizontal_padding()`.

gt – R

```
<gt_data> |>
  opt_horizontal_padding(scale = 3)
```

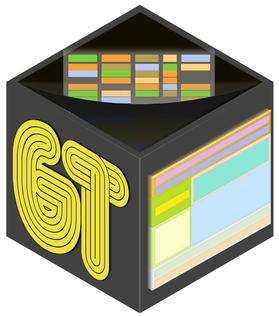
Great Tables – Python

```
<gt_data> \
.opt_horizontal_padding(scale=3)
```

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

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

Let's Try It Out!



r-02-reactions-table.qmd

Using the opt_*() Functions for Quick Styling

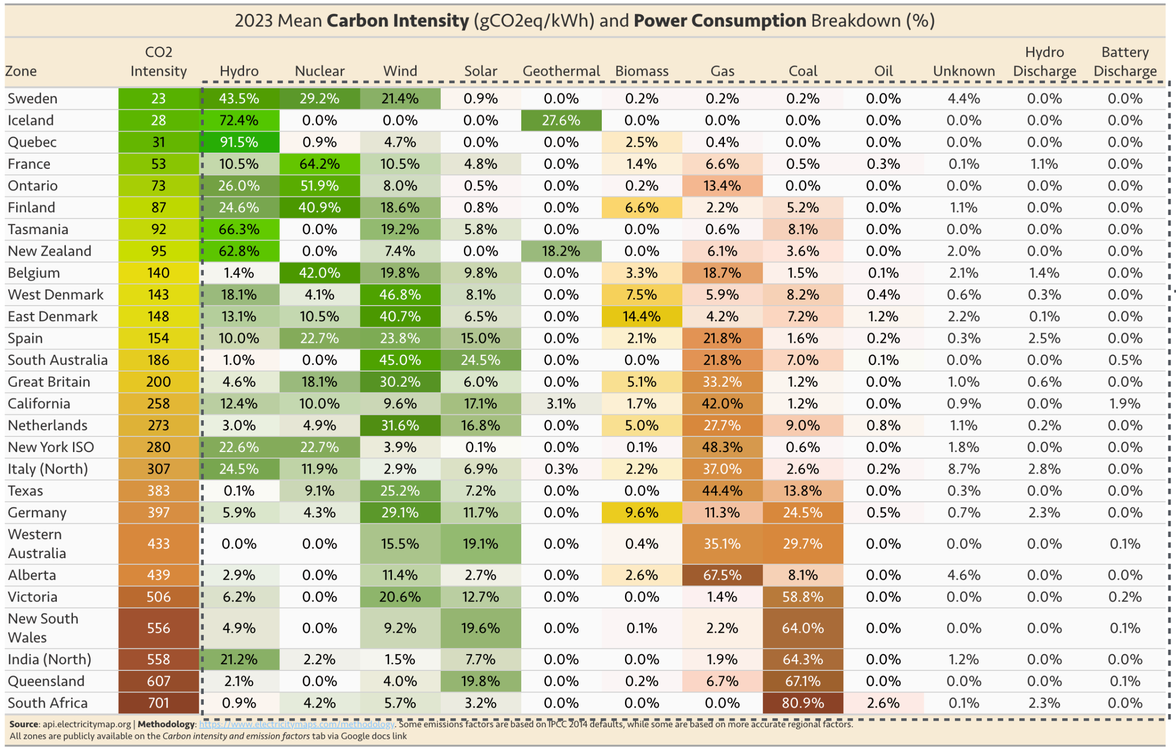
Session 3: **Power Generation Table**

This Is What the **Power Generation Table** Will Look Like in the End

2023 Mean **Carbon Intensity** (gCO2eq/kWh) and **Power Consumption** Breakdown (%)													
Zone	CO2 Intensity	Hydro	Nuclear	Wind	Solar	Geothermal	Biomass	Gas	Coal	Oil	Unknown	Hydro Discharge	Battery Discharge
Sweden	23	43.5%	29.2%	21.4%	0.9%	0.0%	0.2%	0.2%	0.2%	0.0%	4.4%	0.0%	0.0%
Iceland	28	72.4%	0.0%	0.0%	0.0%	27.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Quebec	31	91.5%	0.9%	4.7%	0.0%	0.0%	2.5%	0.4%	0.0%	0.0%	0.0%	0.0%	0.0%
France	53	10.5%	64.2%	10.5%	4.8%	0.0%	1.4%	6.6%	0.5%	0.3%	0.1%	1.1%	0.0%
Ontario	73	26.0%	51.9%	8.0%	0.5%	0.0%	0.2%	13.4%	0.0%	0.0%	0.0%	0.0%	0.0%
Finland	87	24.6%	40.9%	18.6%	0.8%	0.0%	6.6%	2.2%	5.2%	0.0%	1.1%	0.0%	0.0%
Tasmania	92	66.3%	0.0%	19.2%	5.8%	0.0%	0.0%	0.6%	8.1%	0.0%	0.0%	0.0%	0.0%
New Zealand	95	62.8%	0.0%	7.4%	0.0%	18.2%	0.0%	6.1%	3.6%	0.0%	2.0%	0.0%	0.0%
Belgium	140	1.4%	42.0%	19.8%	9.8%	0.0%	3.3%	18.7%	1.5%	0.1%	2.1%	1.4%	0.0%
West Denmark	143	18.1%	4.1%	46.8%	8.1%	0.0%	7.5%	5.9%	8.2%	0.4%	0.6%	0.3%	0.0%
East Denmark	148	13.1%	10.5%	40.7%	6.5%	0.0%	14.4%	4.2%	7.2%	1.2%	2.2%	0.1%	0.0%
Spain	154	10.0%	22.7%	23.8%	15.0%	0.0%	2.1%	21.8%	1.6%	0.2%	0.3%	2.5%	0.0%
South Australia	186	1.0%	0.0%	45.0%	24.5%	0.0%	0.0%	21.8%	7.0%	0.1%	0.0%	0.0%	0.5%
Great Britain	200	4.6%	18.1%	30.2%	6.0%	0.0%	5.1%	33.2%	1.2%	0.0%	1.0%	0.6%	0.0%
California	258	12.4%	10.0%	9.6%	17.1%	3.1%	1.7%	42.0%	1.2%	0.0%	0.9%	0.0%	1.9%
Netherlands	273	3.0%	4.9%	31.6%	16.8%	0.0%	5.0%	27.7%	9.0%	0.8%	1.1%	0.2%	0.0%
New York ISO	280	22.6%	22.7%	3.9%	0.1%	0.0%	0.1%	48.3%	0.6%	0.0%	1.8%	0.0%	0.0%
Italy (North)	307	24.5%	11.9%	2.9%	6.9%	0.3%	2.2%	37.0%	2.6%	0.2%	8.7%	2.8%	0.0%
Texas	383	0.1%	9.1%	25.2%	7.2%	0.0%	0.0%	44.4%	13.8%	0.0%	0.3%	0.0%	0.0%
Germany	397	5.9%	4.3%	29.1%	11.7%	0.0%	9.6%	11.3%	24.5%	0.5%	0.7%	2.3%	0.0%
Western Australia	433	0.0%	0.0%	15.5%	19.1%	0.0%	0.4%	35.1%	29.7%	0.0%	0.0%	0.0%	0.1%
Alberta	439	2.9%	0.0%	11.4%	2.7%	0.0%	2.6%	67.5%	8.1%	0.0%	4.6%	0.0%	0.0%
Victoria	506	6.2%	0.0%	20.6%	12.7%	0.0%	0.0%	1.4%	58.8%	0.0%	0.0%	0.0%	0.2%
New South Wales	556	4.9%	0.0%	9.2%	19.6%	0.0%	0.1%	2.2%	64.0%	0.0%	0.0%	0.0%	0.1%
India (North)	558	21.2%	2.2%	1.5%	7.7%	0.0%	0.0%	1.9%	64.3%	0.0%	1.2%	0.0%	0.0%
Queensland	607	2.1%	0.0%	4.0%	19.8%	0.0%	0.2%	6.7%	67.1%	0.0%	0.0%	0.0%	0.1%
South Africa	701	0.9%	4.2%	5.7%	3.2%	0.0%	0.0%	0.0%	80.9%	2.6%	0.1%	2.3%	0.0%

Source: api.electricitymap.org | **Methodology**: https://www.electricitymaps.com/methodology. Some emissions factors are based on IPCC 2014 defaults, while some are based on more accurate regional factors.
All zones are publicly available on the *Carbon intensity and emission factors* tab via Google docs link

Key Ingredients: Structure, Format, Style



2023 Mean **Carbon Intensity** (gCO2eq/kWh) and **Power Consumption** Breakdown (%)

Zone	CO2 Intensity	Hydro	Nuclear	Wind	Solar	Geothermal	Biomass	Gas	Coal	Oil	Unknown	Hydro Discharge	Battery Discharge
Sweden	23	43.5%	29.2%	21.4%	0.9%	0.0%	0.2%	0.2%	0.2%	0.0%	4.4%	0.0%	0.0%
Iceland	28	72.4%	0.0%	0.0%	0.0%	27.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Quebec	31	91.5%	0.9%	4.7%	0.0%	0.0%	2.5%	0.4%	0.0%	0.0%	0.0%	0.0%	0.0%
France	53	10.5%	64.2%	10.5%	4.8%	0.0%	1.4%	6.6%	0.5%	0.3%	0.1%	1.1%	0.0%
Ontario	73	26.0%	51.9%	8.0%	0.5%	0.0%	0.2%	13.4%	0.0%	0.0%	0.0%	0.0%	0.0%
Finland	87	24.6%	40.9%	18.6%	0.8%	0.0%	6.6%	2.2%	5.2%	0.0%	1.1%	0.0%	0.0%
Tasmania	92	66.3%	0.0%	19.2%	5.8%	0.0%	0.0%	0.6%	8.1%	0.0%	0.0%	0.0%	0.0%
New Zealand	95	62.8%	0.0%	7.4%	0.0%	18.2%	0.0%	6.1%	3.6%	0.0%	2.0%	0.0%	0.0%
Belgium	140	1.4%	42.0%	19.8%	9.8%	0.0%	3.3%	18.7%	1.5%	0.1%	2.1%	1.4%	0.0%
West Denmark	143	18.1%	4.1%	46.8%	8.1%	0.0%	7.5%	5.9%	8.2%	0.4%	0.6%	0.3%	0.0%
East Denmark	148	13.1%	10.5%	40.7%	6.5%	0.0%	14.4%	4.2%	7.2%	1.2%	2.2%	0.1%	0.0%
Spain	154	10.0%	22.7%	23.8%	15.0%	0.0%	2.1%	21.8%	1.6%	0.2%	0.3%	2.5%	0.0%
South Australia	186	1.0%	0.0%	45.0%	24.5%	0.0%	0.0%	21.8%	7.0%	0.1%	0.0%	0.0%	0.5%
Great Britain	200	4.6%	18.1%	30.2%	6.0%	0.0%	5.1%	33.2%	1.2%	0.0%	1.0%	0.6%	0.0%
California	258	12.4%	10.0%	9.6%	17.1%	3.1%	1.7%	42.0%	1.2%	0.0%	0.9%	0.0%	1.9%
Netherlands	273	3.0%	4.9%	31.6%	16.8%	0.0%	5.0%	27.7%	9.0%	0.8%	1.1%	0.2%	0.0%
New York ISO	280	22.6%	22.7%	3.9%	0.1%	0.0%	0.1%	48.3%	0.6%	0.0%	1.8%	0.0%	0.0%
Italy (North)	307	24.5%	11.9%	2.9%	6.9%	0.3%	2.2%	37.0%	2.6%	0.2%	8.7%	2.8%	0.0%
Texas	383	0.1%	9.1%	25.2%	7.2%	0.0%	0.0%	44.4%	13.8%	0.0%	0.3%	0.0%	0.0%
Germany	397	5.9%	4.3%	29.1%	11.7%	0.0%	9.6%	11.3%	24.5%	0.5%	0.7%	2.3%	0.0%
Western Australia	433	0.0%	0.0%	15.5%	19.1%	0.0%	0.4%	35.1%	29.7%	0.0%	0.0%	0.0%	0.1%
Alberta	439	2.9%	0.0%	11.4%	2.7%	0.0%	2.6%	67.5%	8.1%	0.0%	4.6%	0.0%	0.0%
Victoria	506	6.2%	0.0%	20.6%	12.7%	0.0%	0.0%	1.4%	58.8%	0.0%	0.0%	0.0%	0.2%
New South Wales	556	4.9%	0.0%	9.2%	19.6%	0.0%	0.1%	2.2%	64.0%	0.0%	0.0%	0.0%	0.1%
India (North)	558	21.2%	2.2%	1.5%	7.7%	0.0%	0.0%	1.9%	64.3%	0.0%	1.2%	0.0%	0.0%
Queensland	607	2.1%	0.0%	4.0%	19.8%	0.0%	0.2%	6.7%	67.1%	0.0%	0.0%	0.0%	0.1%
South Africa	701	0.9%	4.2%	5.7%	3.2%	0.0%	0.0%	0.0%	80.9%	2.6%	0.1%	2.3%	0.0%

Source: api.electricitymap.org | **Methodology**: https://www.electricitymaps.com/methodology. Some emissions factors are based on IPCC 2014 defaults, while some are based on more accurate regional factors.
All zones are publicly available on the *Carbon intensity and emission factors* tab via Google docs link

Session 3: `tab_source_note()` and `cols_width()`

Structure: `tab_source_note()` — Adding Notes to Your Table

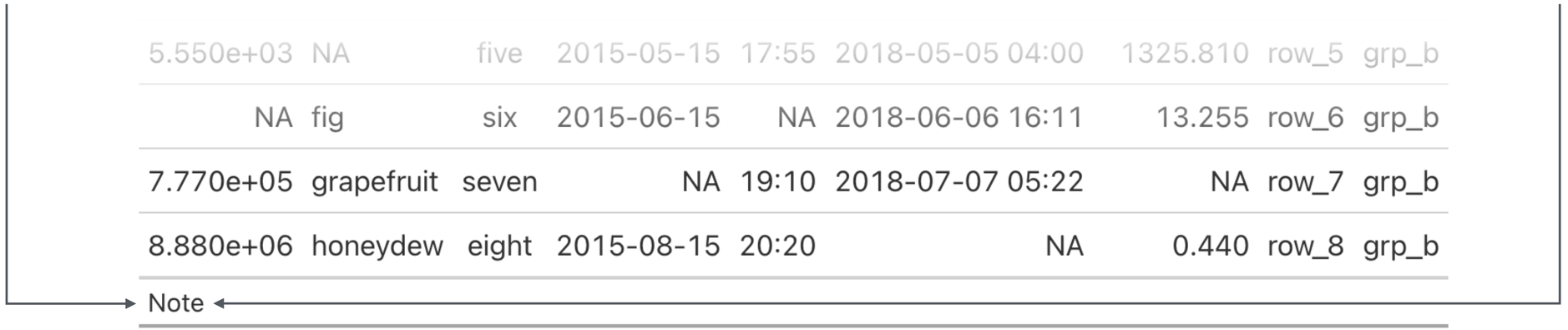
Adding notes to the GT table in a footer component can help explain the table. We do that with `tab_source_note()`.

gt – R

```
gt(exibble) |>
  tab_source_note(source_note = "Note")
```

Great Tables – Python

```
GT(exibble) \
.tab_source_note(source_note="Note")
```



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

Note

Structure: `cols_width()` — Adjusting the Widths of Columns

We can manually specify the widths of columns by using `cols_width()`.

gt – R

```
exibble |>
  dplyr::select(num, char, date) |>
  gt() |>
  cols_width(
    num ~ "150px",
    char ~ "100px",
    date ~ "200px"
  )
```

num	char	date
1.111e-01	apricot	2015-01-15
2.222e+00	banana	2015-02-15
3.333e+01	coconut	2015-03-15
4.444e+02	durian	2015-04-15
5.550e+03	NA	2015-05-15
NA	fig	2015-06-15
7.770e+05	grapefruit	NA
8.880e+06	honeydew	2015-08-15

num	char	date
1.111e-01	apricot	2015-01-15
2.222e+00	banana	2015-02-15
3.333e+01	coconut	2015-03-15
4.444e+02	durian	2015-04-15
5.550e+03	NA	2015-05-15
NA	fig	2015-06-15
7.770e+05	grapefruit	NA
8.880e+06	honeydew	2015-08-15

Structure: `cols_width()` — Adjusting the Widths of Columns

Widths are determined by content size.

num	char	date
1.111e-01	apricot	2015-01-15
2.222e+00	banana	2015-02-15
3.333e+01	coconut	2015-03-15
4.444e+02	durian	2015-04-15
5.550e+03	NA	2015-05-15
NA	fig	2015-06-15
7.770e+05	grapefruit	NA
8.880e+06	honeydew	2015-08-15

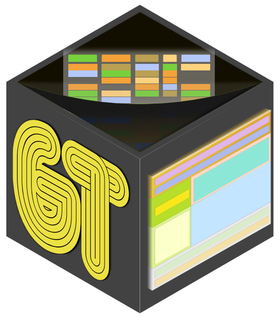
Default Widths

150px 100px 200px

num	char	date
1.111e-01	apricot	2015-01-15
2.222e+00	banana	2015-02-15
3.333e+01	coconut	2015-03-15
4.444e+02	durian	2015-04-15
5.550e+03	NA	2015-05-15
NA	fig	2015-06-15
7.770e+05	grapefruit	NA
8.880e+06	honeydew	2015-08-15

Customized Widths

Let's Try It Out!



r-03-power-gen-table.qmd

Start

tab_source_note() and cols_width()

Formatting Values Within the Table Body

Session 3: `data_color()`

`data_color()`: Heat Maps in Tables

2023 Mean **Carbon Intensity** (gCO2eq/kWh) and **Power Consumption** Breakdown (%)

Zone	CO2 Intensity	Hydro	Nuclear	Wind	Solar	Geothermal	Biomass	Gas	Coal	Oil	Unknown	Hydro Discharge	Battery Discharge
Sweden	23	43.5%	29.2%	21.4%	0.9%	0.0%	0.2%	0.2%	0.2%	0.0%	4.4%	0.0%	0.0%
Iceland	28	72.4%	0.0%	0.0%	0.0%	27.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Quebec	31	91.5%	0.9%	4.7%	0.0%	0.0%	2.5%	0.4%	0.0%	0.0%	0.0%	0.0%	0.0%
France	53	10.5%	64.2%	10.5%	4.8%	0.0%	1.4%	6.6%	0.5%	0.3%	0.1%	1.1%	0.0%
Ontario	73	26.0%	51.9%	8.0%	0.5%	0.0%	0.2%	13.4%	0.0%	0.0%	0.0%	0.0%	0.0%
Finland	87	24.6%	40.9%	18.6%	0.8%	0.0%	6.6%	2.2%	5.2%	0.0%	1.1%	0.0%	0.0%
Tasmania	92	66.3%	0.0%	19.2%	5.8%	0.0%	0.0%	0.6%	8.1%	0.0%	0.0%	0.0%	0.0%
New Zealand	95	62.8%	0.0%	7.4%	0.0%	18.2%	0.0%	6.1%	3.6%	0.0%	2.0%	0.0%	0.0%
Belgium	140	1.4%	42.0%	19.8%	9.8%	0.0%	3.3%	18.7%	1.5%	0.1%	2.1%	1.4%	0.0%
West Denmark	143	18.1%	4.1%	46.8%	8.1%	0.0%	7.5%	5.9%	8.2%	0.4%	0.6%	0.3%	0.0%
East Denmark	148	13.1%	10.5%	40.7%	6.5%	0.0%	14.4%	4.2%	7.2%	1.2%	2.2%	0.1%	0.0%
Spain	154	10.0%	22.7%	23.8%	15.0%	0.0%	2.1%	21.8%	1.6%	0.2%	0.3%	2.5%	0.0%
South Australia	186	1.0%	0.0%	45.0%	24.5%	0.0%	0.0%	21.8%	7.0%	0.1%	0.0%	0.0%	0.5%
Great Britain	200	4.6%	18.1%	30.2%	6.0%	0.0%	5.1%	33.2%	1.2%	0.0%	1.0%	0.6%	0.0%
California	258	12.4%	10.0%	9.6%	17.1%	3.1%	1.7%	42.0%	1.2%	0.0%	0.9%	0.0%	1.9%
Netherlands	273	3.0%	4.9%	31.6%	16.8%	0.0%	5.0%	27.7%	9.0%	0.8%	1.1%	0.2%	0.0%
New York ISO	280	22.6%	22.7%	3.9%	0.1%	0.0%	0.1%	48.3%	0.6%	0.0%	1.8%	0.0%	0.0%
Italy (North)	307	24.5%	11.9%	2.9%	6.9%	0.3%	2.2%	37.0%	2.6%	0.2%	8.7%	2.8%	0.0%
Texas	383	0.1%	9.1%	25.2%	7.2%	0.0%	0.0%	44.4%	13.8%	0.0%	0.3%	0.0%	0.0%
Germany	397	5.9%	4.3%	29.1%	11.7%	0.0%	9.6%	11.3%	24.5%	0.5%	0.7%	2.3%	0.0%
Western Australia	433	0.0%	0.0%	15.5%	19.1%	0.0%	0.4%	35.1%	29.7%	0.0%	0.0%	0.0%	0.1%
Alberta	439	2.9%	0.0%	11.4%	2.7%	0.0%	2.6%	67.5%	8.1%	0.0%	4.6%	0.0%	0.0%
Victoria	506	6.2%	0.0%	20.6%	12.7%	0.0%	0.0%	1.4%	58.8%	0.0%	0.0%	0.0%	0.2%
New South Wales	556	4.9%	0.0%	9.2%	19.6%	0.0%	0.1%	2.2%	64.0%	0.0%	0.0%	0.0%	0.1%
India (North)	558	21.2%	2.2%	1.5%	7.7%	0.0%	0.0%	1.9%	64.3%	0.0%	1.2%	0.0%	0.0%
Queensland	607	2.1%	0.0%	4.0%	19.8%	0.0%	0.2%	6.7%	67.1%	0.0%	0.0%	0.0%	0.1%
South Africa	701	0.9%	4.2%	5.7%	3.2%	0.0%	0.0%	0.0%	80.9%	2.6%	0.1%	2.3%	0.0%

Source: api.electricitymap.org | **Methodology**: https://www.electricitymaps.com/methodology. Some emissions factors are based on IPCC 2014 defaults, while some are based on more accurate regional factors.
All zones are publicly available on the *Carbon intensity and emission factors* tab via Google docs link

`data_color()`: generate a heat map with a palette.

It makes the large amount of information here much easier to digest at a glance.

Larger values can be seen right away. Negligible values lack color.

`data_color()`: Heat Maps in Tables

Two big advantages from plots:

1. emphasizes differences in values
2. reveals trends in the data

`data_color()`: Heat Maps in Tables

Two big advantages from plots:

1. emphasizes differences in values
2. reveals trends in the data

Row: compare across measures.

Zone	Hydro	Nuclear	Wind	Solar	Geothermal
France	10.5%	64.3%	10.4%	4.8%	0 %

`data_color()`: Heat Maps in Tables

Two big advantages from plots:

1. emphasizes differences in values
2. reveals trends in the data

Row: compare across measures.

Zone	Hydro	Nuclear	Wind	Solar	Geothermal
France	10.5%	64.3%	10.4%	4.8%	0 %

Column: compare across observations.

Zone	CO2 Intensity
Sweden	22
Iceland	28
Quebec	30
France	44
Tasmania	67
Ontario	70

data_color(): Heat Maps in Tables

2023 Mean **Carbon Intensity** (gCO2eq/kWh) and **Power Consumption** Breakdown (%)

Zone	CO2 Intensity	Hydro	Nuclear	Wind	Solar	Geothermal	Biomass	Gas	Coal	Oil	Unknown	Hydro Discharge	Battery Discharge
Sweden	23	43.5%	29.2%	21.4%	0.9%	0.0%	0.2%	0.2%	0.2%	0.0%	4.4%	0.0%	0.0%
Iceland	28	72.4%	0.0%	0.0%	0.0%	27.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Quebec	31	91.5%	0.9%	4.7%	0.0%	0.0%	2.5%	0.4%	0.0%	0.0%	0.0%	0.0%	0.0%
France	53	10.5%	64.2%	10.5%	4.8%	0.0%	1.4%	6.6%	0.5%	0.3%	0.1%	1.1%	0.0%
Ontario	73	26.0%	51.9%	8.0%	0.5%	0.0%	0.2%	13.4%	0.0%	0.0%	0.0%	0.0%	0.0%
Finland	87	21.6%	40.9%	18.6%	0.8%	0.0%	6.6%	2.2%	5.2%	0.0%	1.1%	0.0%	0.0%
Tasmania	92	66.3%	0.0%	19.2%	5.8%	0.0%	0.0%	0.6%	8.1%	0.0%	0.0%	0.0%	0.0%
New Zealand	95	62.8%	0.0%	7.4%	0.0%	18.2%	0.0%	6.1%	3.6%	0.0%	2.0%	0.0%	0.0%
Belgium	140	1.4%	42.0%	19.8%	9.8%	0.0%	3.3%	18.7%	1.5%	0.1%	2.1%	1.4%	0.0%
West Denmark	143	18.1%	4.1%	46.8%	8.1%	0.0%	7.5%	5.9%	8.2%	0.4%	0.6%	0.3%	0.0%
East Denmark	148	13.1%	10.5%	40.7%	6.5%	0.0%	14.4%	4.2%	7.2%	1.2%	2.2%	0.1%	0.0%
Spain	154	10.0%	22.7%	23.8%	15.0%	0.0%	2.1%	21.8%	1.6%	0.2%	0.3%	2.5%	0.0%
South Australia	186	1.0%	0.0%	45.0%	29.5%	0.0%	0.0%	21.8%	7.0%	0.1%	0.0%	0.0%	0.5%
Great Britain	200	4.6%	18.1%	30.2%	6.0%	0.0%	5.1%	33.2%	1.2%	0.0%	1.0%	0.6%	0.0%
California	258	12.4%	10.0%	9.6%	17.1%	3.1%	1.7%	42.0%	1.2%	0.0%	0.9%	0.0%	1.9%
Netherlands	273	3.0%	4.9%	31.6%	16.8%	0.0%	5.0%	27.7%	9.0%	0.8%	1.1%	0.2%	0.0%
New York ISO	280	22.6%	22.7%	3.9%	0.1%	0.0%	0.1%	48.3%	0.6%	0.0%	1.8%	0.0%	0.0%
Italy (North)	307	24.5%	11.9%	2.9%	6.9%	0.3%	2.2%	37.0%	2.6%	0.2%	8.7%	2.8%	0.0%
Texas	383	0.1%	9.1%	25.2%	7.2%	0.0%	0.0%	44.4%	13.8%	0.0%	0.3%	0.0%	0.0%
Germany	397	5.9%	4.3%	29.1%	11.7%	0.0%	9.6%	11.3%	24.5%	0.5%	0.7%	2.3%	0.0%
Western Australia	433	0.0%	0.0%	15.5%	19.1%	0.0%	0.4%	35.1%	29.7%	0.0%	0.0%	0.0%	0.1%
Alberta	439	2.9%	0.0%	11.4%	2.7%	0.0%	2.6%	67.5%	8.1%	0.0%	4.6%	0.0%	0.0%
Victoria	506	6.2%	0.0%	20.6%	12.7%	0.0%	0.0%	1.4%	58.8%	0.0%	0.0%	0.0%	0.2%
New South Wales	556	4.9%	0.0%	9.2%	19.6%	0.0%	0.1%	2.2%	64.0%	0.0%	0.0%	0.0%	0.1%
India (North)	558	21.2%	2.2%	1.5%	7.7%	0.0%	0.0%	1.9%	64.3%	0.0%	1.2%	0.0%	0.0%
Queensland	607	2.1%	0.0%	4.0%	19.8%	0.0%	0.2%	6.7%	67.1%	0.0%	0.0%	0.0%	0.1%
South Africa	701	0.9%	4.2%	5.7%	3.2%	0.0%	0.0%	0.0%	80.9%	2.6%	0.1%	2.3%	0.0%

Source: api.electricitymap.org | **Methodology**: https://www.electricitymaps.com/methodology. Some emissions factors are based on IPCC 2014 defaults, while some are based on more accurate regional factors.
All zones are publicly available on the *Carbon intensity and emission factors* tab via Google docs link

We can see a global pattern of values in the table.

Trend here: more energy from columns to the right leads to higher CO_2 intensity values.

`data_color()`: Heat Maps in Tables

`data_color()` can be used without any supplied arguments to colorize a gt table

gt – R

```
exibble |>
  gt() |>
  data_color()
```

Great Tables – Python

```
GT(exibble) \
.data_color()
```

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

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

`data_color()`: Heat Maps in Tables

For numeric columns, let's supply a palette of two colors: red and green.

gt – R

```
exibble |>
  gt() |>
  data_color(
    columns = c(num, currency),
    palette = c("red", "green")
  )
```

Great Tables – Python

```
GT(exibble) \
.data_color(
  columns=["num", "currency"]),
  palette=["red", "green"])
)
```

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

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

`data_color()`: Heat Maps in Tables

We can constrain the cells to which coloring will be applied with the `columns` and `rows` arguments. Also, we can manually set the limits of the data with the `domain` argument

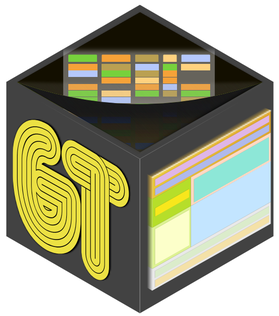
gt – R

```
exibble |>
  gt() |>
  data_color(
    columns = currency,
    rows = currency < 50,
    method = "numeric",
    palette = c("red", "green"),
    domain = c(0, 50)
  )
```

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

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

Let's Try It Out!



r-03-power-gen-table.qmd

data_color()

Session 3: `cols_align()` and `tab_options()`

Style: `cols_align()` — Changing the Alignment of Columns

You can modify the alignment of values in a column with `cols_align()`.

gt – R

```
exibble |>
  gt() |>
  cols_align(
    columns = char,
    align = "right"
  )
```

Great Tables – Python

```
GT(exibble) \
  .cols_align(
    columns="char",
    align="right"
)
```

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

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

Style: `cols_align()` — Changing the Alignment of Columns

num	char
1.111e-01	apricot
2.222e+00	banana
3.333e+01	coconut
4.444e+02	durian
5.550e+03	NA
NA	fig
7.770e+05	grapefruit
8.880e+06	honeydew

Before Change

num	char
1.111e-01	apricot
2.222e+00	banana
3.333e+01	coconut
4.444e+02	durian
5.550e+03	NA
NA	fig
7.770e+05	grapefruit
8.880e+06	honeydew

After Change

Style: `tab_options()` — Many, Many Options for Table Modding

While the number of options available in `tab_options()` is huge, it's always very useful.

gt – R

```
exibble |>
  gt() |>
  tab_options(
    ...
  )
```

Great Tables – Python

```
GT(exibble) \
  .tab_options(
    ...
)
```

Best thing to do is consult the online documentation...

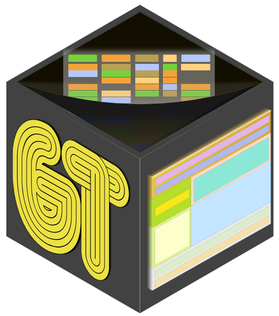
...for **gt**

`https://gt.rstudio.com/reference/tab_options.html`

...for **Great Tables**

`https://posit-dev.github.io/great-tables/reference/GT.tab_options.html`

Let's Try It Out!



`r-03-power-gen-table.qmd`

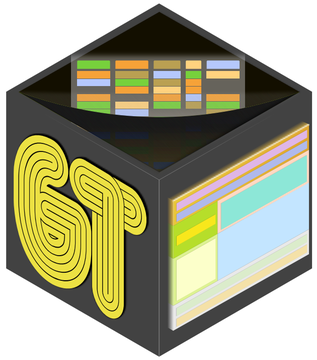
```
## cols_align() and tab_options()
```

Session 4: Make Your Own Excellent Tables

Make Your Own Excellent Tables

Now it's time to break free of the instruction and code along loop and DIY your own tables.

Use **gt** or **Great Tables**. Ask questions. Get some feedback.



By the end of this session you should have one or more tables you can be proud of!

Suggested Datasets for Table Work

The datasets from **gt** and **Great Tables**.

`countrypops` `sza` `gtcars` `sp500` `pizzaplace`
`exibble` `towny` `peeps` `films` `metro` `gibraltar`
`constants` `illness` `reactions` `photolysis` `nuclides`

TidyTuesday's catalog of data.

`https://github.com/rfordatascience/tidytuesday`

Kaggle's open datasets.

`https://www.kaggle.com/datasets`

Plotly datasets.

`https://github.com/plotly/datasets`

The Workshop Survey

We'd really appreciate your feedback!

So please fill this quick survey and let us know your thoughts.

(All answers will be anonymous.)

`https://pos.it/conf-workshop-survey`