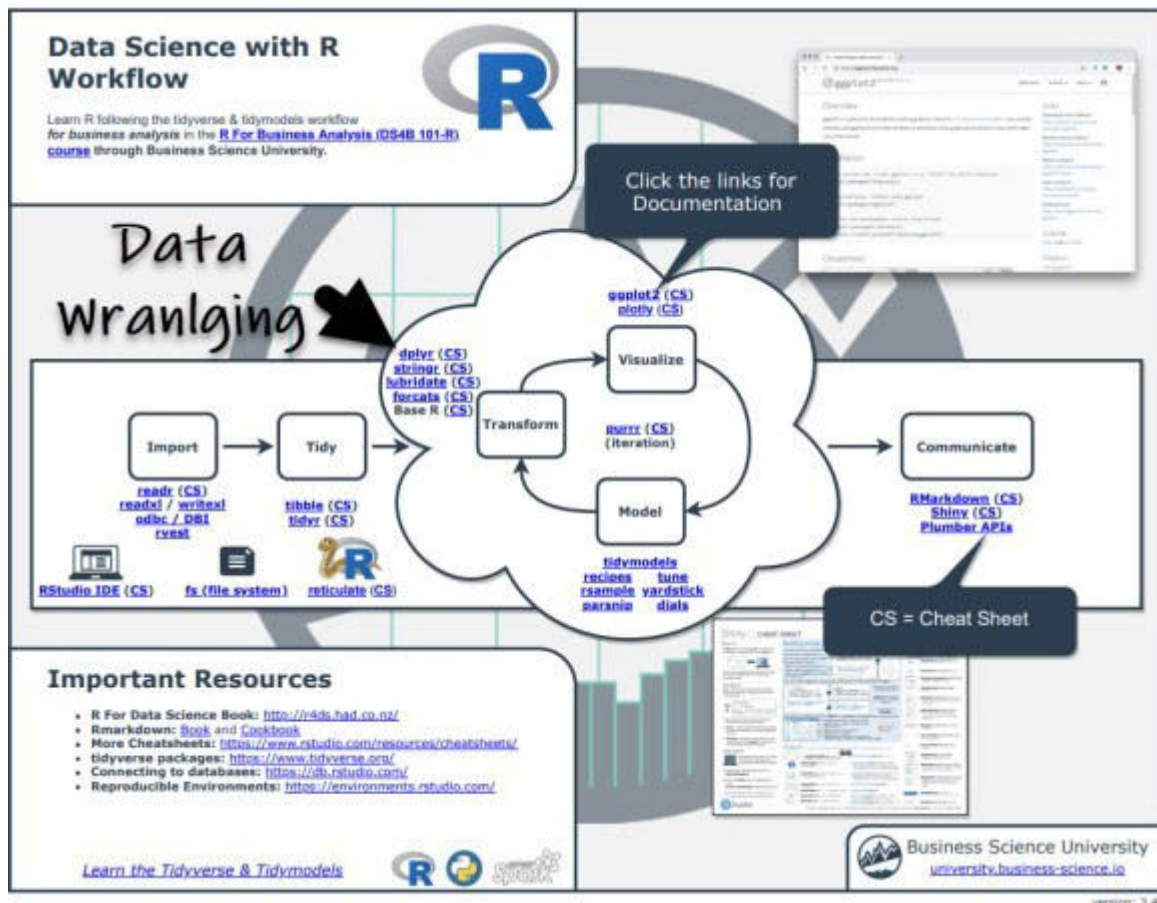


Before we get started, get the Cheat Sheet

Datapasta is great for importing raw data from HTML tables on the web. But, you'll still need to learn how to transform / wrangle the data and produce visualizations. For those topics, I'll use the [Ultimate R Cheat Sheet](#) to refer to `dplyr` and `ggplot2` code in my workflow.

Quick example – Clicking the “CS” next to “dplyr” opens the Data Transformation with Dplyr Cheat Sheet.



Now you're ready to quickly reference `dplyr` functions. Ok, onto the tutorial.

Data Transformation with dplyr : CHEAT SHEET



dplyr functions work with pipes and expect tidy data. In tidy data:

Each variable is in its own column
Each observation, or case, is in its own row
x %>% f(y) becomes f(x, y)

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function
summarise(data, ...)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))
count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in ... Also tally().
count(mtcars, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))

group_by(data, ..., add = FALSE)
Returns copy of table grouped by ...
g_mtcars <- group_by(mtcars, Species)

ungroup(x, ...)
Returns ungrouped copy of table.
ungroup(g_mtcars)

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(data, ...) Extract rows that meet logical criteria. filter(mtcars, Sepal.Length > 7)
distinct(data, ..., keep_all = FALSE) Remove rows with duplicate values. distinct(mtcars, Species)
sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, env = parent.frame()) Randomly select fraction of rows. sample_frac(mtcars, 0.3, replace = TRUE)
sample_n(tbl, size, replace = FALSE, weight = NULL, env = parent.frame()) Randomly select size rows. sample_n(mtcars, 20, replace = FALSE)
slice(data, ...) Select rows by position. slice(mtcars, 20:25)
top_n(x, n, wt) Select and order top n entries (by group if grouped data). top_n(mtcars, 5, Sepal.Length)

Logical and boolean operators to use with filter()
< <= is.na() %in% | xor()
> >= is.na() !
See ?base::Logic and ?base::Comparison for help.

ARRANGE CASES

arrange(data, ...) Order rows by values of a column or columns (low to high), use with desc() to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(data, ..., before = NULL, after = NULL)
Add one or more rows to a table.
add_row(mtcars, c(10, 10, 10, 10, 10))

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(data, var = 1) Extract column values as a vector. Choose by name or index. pull(mtcars, Sepal.Length)
select(data, ...) Extract columns as a table. Also select_if(). select(mtcars, Sepal.Length, Species)

Use these helpers with select(), e.g. select(mtcars, starts_with("Sepal"))
contains(match) num_range(prefix, range) e.g. mpg:cyl
ends_with(match) one_of(...) e.g. Species
matches(match) starts_with(match)

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

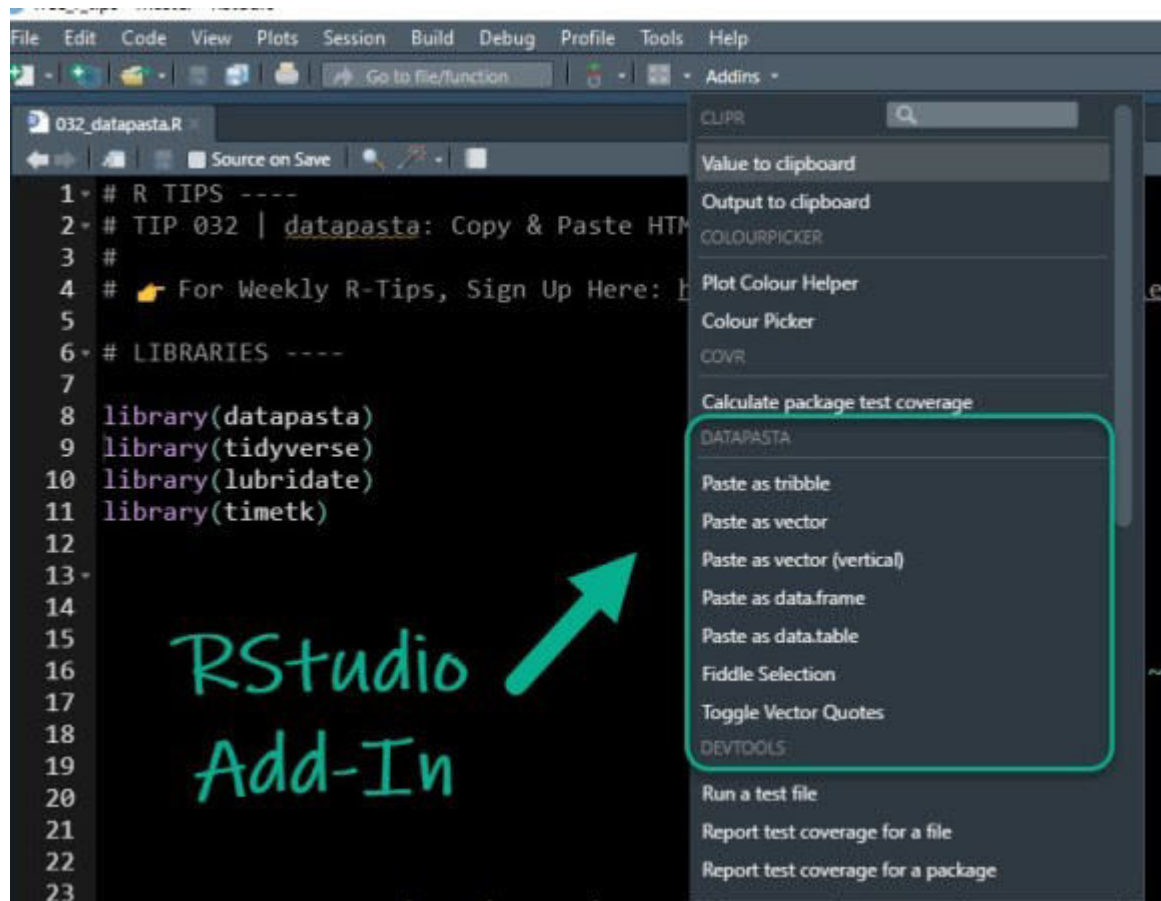
vectorized function
mutate(data, ...) Compute new column(s). mutate(mtcars, gpm = 1/mtg)
transmute(data, ...) Compute new column(s), drop others. transmute(mtcars, gpm = 1/mtg)
mutate_all(tbl, funs, ...) Apply funs to every column. Use with funs(). Also mutate_if(). mutate_all(mtcars, funs(log10, log2))
mutate_at(tbl, cols, funs, ...) Apply funs to specific columns. Use with funs(), vars() and the helper functions for select(). mutate_at(mtcars, vars(Species), funs(log10))
add_column(data, ..., before = NULL, after = NULL) Add new column(s). Also add_count(), add_tally(). add_count(mtcars, new = 2:32)
rename(data, ...) Rename columns. rename(mtcars, Length = Sepal.Length)



RStudio is a trademark of RStudio, Inc. © 2019 RStudio - info@rstudio.com - 844-449-1212 - rstudio.com - Learn more with [install.packages\("dplyr"\)](#) or [install.packages\("tidyverse"\)](#) - dplyr 0.7.6 - 2019-01-10 - Updated 2019-08

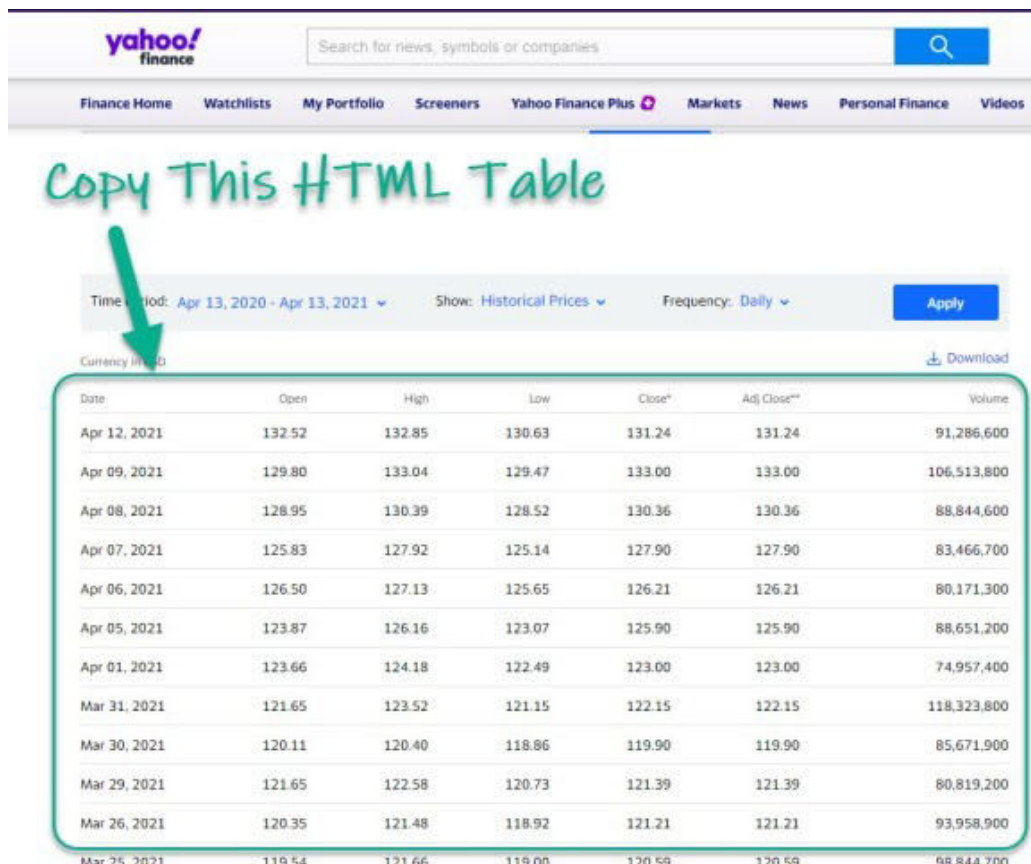
Datapasta RStudio Addin

Datapasta contains an RStudio Add-In for Pasting web-tables stored in your "clipboard" (what happens when you "copy" something).



Example 1: Copying Stock Data from Yahoo! Finance

Let's go through a quick example. We can navigate to Yahoo! Finance and search for a ticker symbol like AAPL.



Copy This HTML Table

Time Period: Apr 13, 2020 - Apr 13, 2021 Show: Historical Prices Frequency: Daily Apply

Currency: USD Download

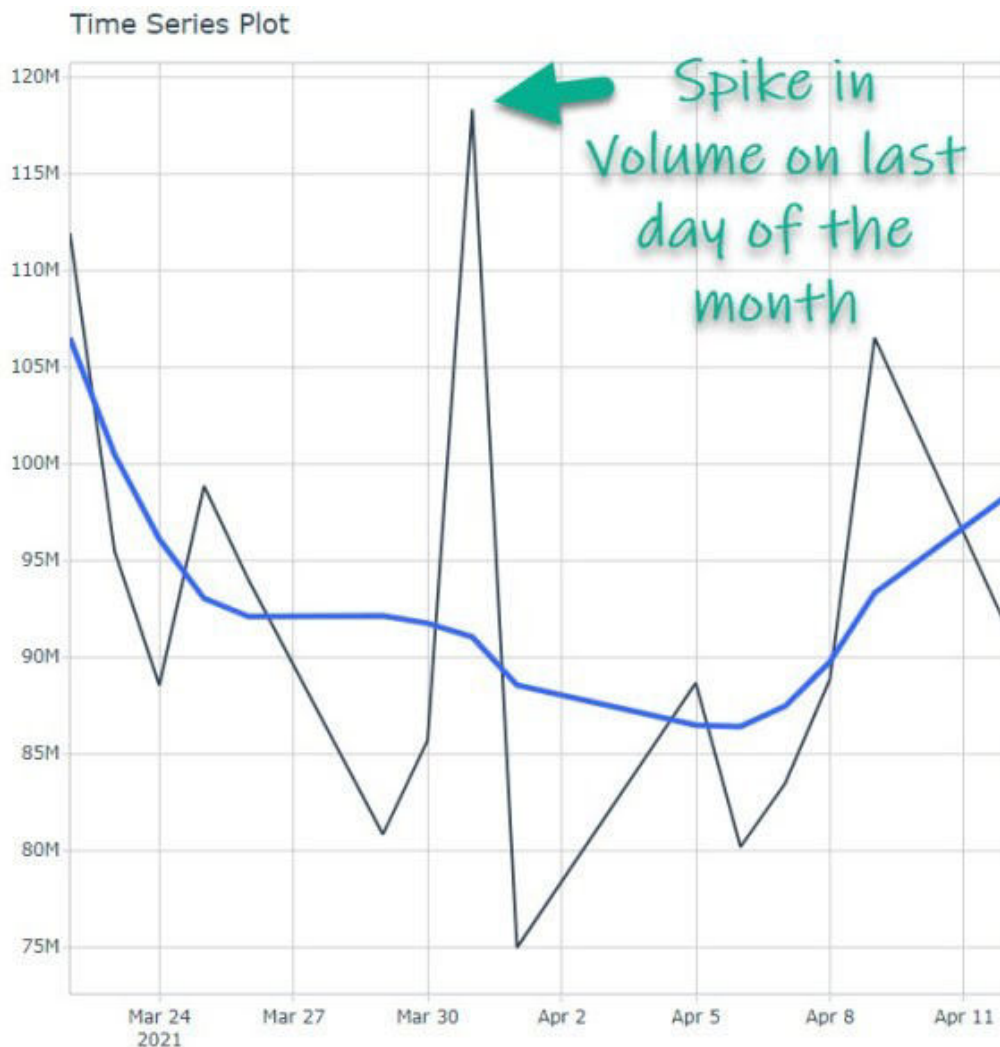
Date	Open	High	Low	Close*	Adj Close**	Volume
Apr 12, 2021	132.52	132.85	130.63	131.24	131.24	91,286,600
Apr 09, 2021	129.80	133.04	129.47	133.00	133.00	106,513,800
Apr 08, 2021	128.95	130.39	128.52	130.36	130.36	88,844,600
Apr 07, 2021	125.83	127.92	125.14	127.90	127.90	83,466,700
Apr 06, 2021	126.50	127.13	125.65	126.21	126.21	80,171,300
Apr 05, 2021	123.87	126.16	123.07	125.90	125.90	88,651,200
Apr 01, 2021	123.66	124.18	122.49	123.00	123.00	74,957,400
Mar 31, 2021	121.65	123.52	121.15	122.15	122.15	118,323,800
Mar 30, 2021	120.11	120.40	118.86	119.90	119.90	85,671,900
Mar 29, 2021	121.65	122.58	120.73	121.39	121.39	80,819,200
Mar 26, 2021	120.35	121.48	118.92	121.21	121.21	93,958,900
Mar 25, 2021	119.54	121.66	119.00	120.59	120.59	98,844,700
Mar 24, 2021	122.82	122.90	120.07	120.09	120.09	88,530,500
Mar 23, 2021	123.33	124.24	122.14	122.54	122.54	95,467,100
Mar 22, 2021	120.33	123.87	120.26	123.39	123.39	111,912,300

Source: Yahoo! Finance

Next, use the Datapasta Addin to “paste as tibble”. This pastes our data into our R script file.

```
tibble::tribble(
  ~Date, ~Open, ~High, ~Low, ~Close*, ~Adj.Close**, ~Volume,
  "Apr 12, 2021", 132.52, 132.85, 130.63, 131.24, 131.24, 91286600,
  "Apr 09, 2021", 129.8, 133.04, 129.47, 133, 133, 106513800,
  "Apr 08, 2021", 128.95, 130.39, 128.52, 130.36, 130.36, 88844600,
  "Apr 07, 2021", 125.83, 127.92, 125.14, 127.9, 127.9, 83466700,
  "Apr 06, 2021", 126.5, 127.13, 125.65, 126.21, 126.21, 80171300,
  "Apr 05, 2021", 123.87, 126.16, 123.07, 125.9, 125.9, 88651200,
  "Apr 01, 2021", 123.66, 124.18, 122.49, 123, 123, 74957400,
  "Mar 31, 2021", 121.65, 123.52, 121.15, 122.15, 122.15, 118323800,
  "Mar 30, 2021", 120.11, 120.4, 118.86, 119.9, 119.9, 85671900,
  "Mar 29, 2021", 121.65, 122.58, 120.73, 121.39, 121.39, 80819200,
  "Mar 26, 2021", 120.35, 121.48, 118.92, 121.21, 121.21, 93958900,
  "Mar 25, 2021", 119.54, 121.66, 119, 120.59, 120.59, 98844700,
  "Mar 24, 2021", 122.82, 122.9, 120.07, 120.09, 120.09, 88530500,
  "Mar 23, 2021", 123.33, 124.24, 122.14, 122.54, 122.54, 95467100,
  "Mar 22, 2021", 120.33, 123.87, 120.26, 123.39, 123.39, 111912300
)
```

Next, use `dplyr` and `timetk` to wrangle and visualize the data. (Refer to the [ultimate R cheat sheet for documentation on dplyr and timetk](#)). We can see a spike in volume on last day of the month.



Code available in our [Free R-Tips Github Repository](#)

Example 2: Getting Revenue Data for World Largest Companies From Wikipedia

First, head over to Wikipedia and search for the “list of largest companies”.

en.wikipedia.org/wiki/List_of_largest_companies_by_revenue

Contents [hide]

- 1 List
- 2 Notes
- 3 See also
- 4 References
- 5 External links

List [edit]

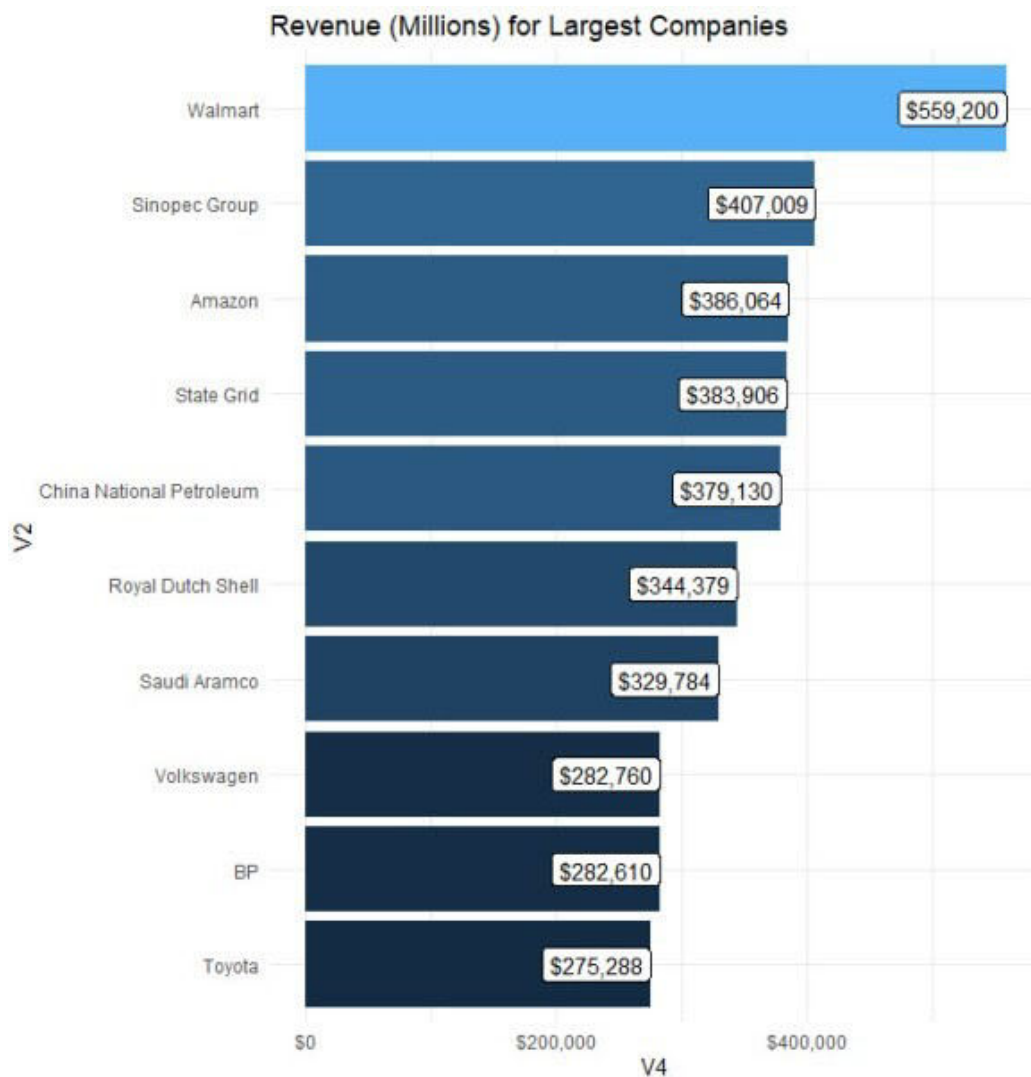
☐ State-owned enterprise (government owns 50% or more)

Copy table rows 1-10
(skipping the table header)

Rank	Name	Industry	Revenue	Profit	Employees	Headquarters	Ref
			USD Millions				
1	Walmart	Retail	▲ \$559,200	\$19,742	2,200,000	 United States	[4]
2	Sinopec Group	Oil and gas	▼ \$407,009	\$6,793	582,648	 China	[5]
3	Amazon	Retail	▲ \$386,064	\$17,377	1,125,300	 United States	[6]
4	State Grid	Electricity	▼ \$383,906	\$7,970	907,677	 China	[7]
5	China National Petroleum	Oil and gas	▼ \$379,130	\$4,433	1,344,410	 China	[8]
6	Royal Dutch Shell	Oil and gas	▼ \$344,379	\$15,842	83,000	 Netherlands	[9]
7	Saudi Aramco	Oil and gas	▼ \$329,784	\$88,211	79,000	 Saudi Arabia	[10]
8	Volkswagen	Automotive	▲ \$282,760	\$15,542	671,205	 Germany	[11]
9	BP	Oil and gas	▼ \$282,610	\$4,026	72,500	 United Kingdom	[12]
10	Toyota	Automotive	▲ \$275,288	\$19,096	359,542	 Japan	[13]
11	Apple	Electronics	▼ \$274,515	\$55,256	137,000	 United States	[14]
12	ExxonMobil	Oil and gas	▼ \$264,938	\$14,340	74,900	 United States	[15]
13	CVS Health	Healthcare	▲ \$256,776	\$6,634	290,000	 United States	[16]
14	Berkshire Hathaway	Financials	▲ \$254,616	\$81,417	391,500	 United States	[17]

Source: Wikipedia

Use **datapasta** to “paste as data.table”. Then do some data wrangling with **dplyr**. Then visualize with **ggplot2**. And in a few lines of code you can create this chart showing that Walmart is dominating in Revenue. (Refer to the [ultimate R cheat sheet for documentation on dplyr and ggplot2](#)).



Code available in our [Free R-Tips Github Repository](#)

In Summary

You just quickly scraped HTML tables using the copy-and-paster Rstudio Add-In known as **datapasta** . This is an amazing productivity boost!!