Introduction to Data Visualization with ggplot2

Code

1 Objectives

This document outlines and introduction to data visualization with `ggplot2`.

2 Materials

The slides for this presentation are here (https://mjfrigaard.github.io/data-viz-intro/Index.html)

There is also an accompanying RStudio.Cloud project:

3 Previous lessons

All of the exercises and lessons are available here (https://mjfrigaard.github.io/r-meetup-tutorials/), but you can also read more about `dplyr` (https://dplyr.tidyverse.org/index.html) and `tidyr` (https://tidyr.tidyverse.org/) on the tidyverse website, and in the Data Transformation (https://r4ds.had.co.nz/transform.html) and Tidy Data (https://r4ds.had.co.nz/tidy-data.html) chapters of R for Data Science.

4 Load the packages

The main packages we're going to use are `dplyr`, `tidyr`, and `ggplot2`. These are all part of the `tidyverse`, so we'll import this package below:

Hide

```
install.packages("tidyverse")
library(tidyverse)
```

5 Largest Pharmaceutical Companies

In the slides we had well-formatted dataset. In these exercises, we're going to import some data from the wild (Wikipedia), because most data aren't ready to visualize and model when we first get them.

5.1 Import the data

I Googled "largest pharmaceutical companies" and found this wikipedia page (https://en.wikipedia.org/wiki/List_of_largest_biotechnology_and_pharmaceutical_companies).

These packages will help us scrape the data in the table and manipulate it in R.

Hide

```
library(xml2)
library(rvest)
```

5.1.1 Read HTML | 5.1.2 HTML Nodes (css) | 5.1.3 Extract HTML table

The first function I'll use comes from the `xml2` package (https://blog.rstudio.com/2015/04/21/xml2/). `xml2::read_html()` loads the html from the wikipedia page into an R object I call `wiki_html`.

Hide

```
wiki_html <- xml2::read_html("https://en.wikipedia.org/wiki/List_of_large
st_biotechnology_and_pharmaceutical_companies")
```

We should check the structure of new objects, so we know what we're working with.

Hide

```
wiki_html %>% str()
```

```
## List of 2
##  $ node:<externalptr>
##  $ doc :<externalptr>
##  - attr(*, "class")= chr [1:2] "xml_document" "xml_node"
```

I can see this is a list of two objects (a `node` and a `doc`).

5.2 Data wrangling (outline)

We have a raw dataset now, and we should make a 'to-do' list of what we want the data to look like. I start with data tidying, then move into changes for the individual variables.

6 Tidy Data

These data are in a wide format, with each `market_cap_in_` spread across columns.

Hide

```
TopPharmCompRaw
```

Rank[N 1] <chr>
1
2 1
3 2
4
5 3
6 2
7
8 2
9 3
10 2

1-10 of 33 rows | 1-1 of 10 columns Previous **1** 2 3 4 Next

6.1 Pivot data

Ideally, we want two variables, `year` and `market_cap_us_bil`.

6.1.1 `pivot_longer`

We can re-shape `TopPharmComp` with `tidyr::pivot_longer()`, but first we need all the variables in the *same* format. We can do this with `mutate_if()`.

Hide

```
TopPharmCompRaw %>%
  mutate_if(is.numeric, as.character) %>%
  tidyr::pivot_longer(cols = starts_with("Market Cap in"),
                      names_to = "year",
                      values_to = "market_cap_us_bil")
```

Rank[N 1]
<chr>
1
1
1
1
1
1
1
2 1
2 1
2 1

1-10 of 231 rows | 1-1 of 5 columns Previous **1** 2 3 4 … 24 Next

This looks correct–let's assign it to `TidyTopPharmComp` (because it's a new data structure).

Hide

```
TidyTopPharmComp <- TopPharmCompRaw %>%
  mutate_if(is.numeric, as.character) %>%
  tidyr::pivot_longer(cols = starts_with("Market Cap in"),
                      names_to = "year",
                      values_to = "market_cap_us_bil")
```

6.2 Wrangle Columns

Follow along on each tab for the steps to wrangle the column names:

1. Each variable in it's own column:
    - i.e. type of company in it's own column (`B` – Biotechnology company, `P` – Pharmaceutical company)
    - stock exchange identifier in it's own column (i.e. `NYSE` , `NASDAQ` , etc.)
2. Properly formatted values (numeric, factor, etc.)
3. Missing variables formatted correctly (replace `-` with `NA`)

6.2.1 String manipulations | 6.2.2 `janitor::clean_names` | 6.2.3 Assign to data frame

---

We can use `janitor::clean_names()` , but first we should do some string manipulation to remove the extra characters from the column names.

Hide

```
TidyTopPharmComp %>%
    # get a vector of 'dirty' names
    names() %>%
    # remove citations in brackets [ ]
    stringr::str_remove_all(string = ., pattern = "\\[[0-9]\\]") %>%
    # replace (USD billions)
    stringr::str_replace_all(string = ., pattern = "\\s*\\([^\\)]+\\)",
                             replacement = "_us_bil") %>%
    # remove bracket from Rank
    stringr::str_remove_all(string = ., pattern = "\\[|\\]") %>%
    # remove alpha numeric from end of Rank
    stringr::str_remove_all(string = ., pattern = "N 1$") %>%
    # make all lowercase
    stringr::str_to_lower() -> cleaned_names
cleaned_names
```

```
## [1] "rank"                      "company"
## [3] "largest market cap_us_bil" "year"
## [5] "market_cap_us_bil"
```

6.3 Define Variables

We want to split up the following variables into their own columns:

1. Year as a four-number digit `year`
2. `company_type` = `[P]` for `Pharmaceutical` or `[B]` for `Biotechnology`
3. `stock_exch` = `NSYE` (New York Stock Exchange), `NASDAQ` (National Association of Securities Dealers Automated Quotations), `FWB` (Frankfurt Stock Exchange), `TYO` (Australian Securities Exchange), `TSX` (Toronto Stock Exchange), and `SIX` (Swiss Exchange).
4. `stock_id` = acronym for each company on `stock_exch`
5. `largest_market_cap_date` = date from `largest_market_cap_us_bil`
6. `company_name` a variable containing *only* the company name
7. a `ranking` variable with a numerically coded rank

6.3.1 `year` 6.3.2 `company_type` 6.3.3 `stock_exch` & `stock_id`

6.3.4 `largest_market_cap_date` 6.3.5 `company_name` 6.3.6 `ranking`

6.3.7 Missing Values

We can also wrangle the `market_cap_year` variable so it only contains the four-number year.

Hide

```
TopPharmComp %>%
  # remove characters
  dplyr::mutate(year = str_remove_all(string = year, pattern = "\\D"),
                year = str_sub(string = year, start = 1L, end = 4L),
                # make numeric
                year = as.integer(year))
```

rank <chr>	company <chr>
1	Johnson & Johnson[P]NYSE: JNJ
1	Johnson & Johnson[P]NYSE: JNJ
1	Johnson & Johnson[P]NYSE: JNJ
1	Johnson & Johnson[P]NYSE: JNJ
1	Johnson & Johnson[P]NYSE: JNJ
1	Johnson & Johnson[P]NYSE: JNJ
1	Johnson & Johnson[P]NYSE: JNJ
2 1	Roche[P]SIX: ROG
2 1	Roche[P]SIX: ROG
2 1	Roche[P]SIX: ROG

1-10 of 231 rows | 1-2 of 5 columns Previous **1** 2 3 4 … 24 Next

Let's assign `year` to `TopPharmComp`

Hide

```
TopPharmComp %>%
  # remove characters
  dplyr::mutate(year = str_remove_all(string = year, pattern = "\\D"),
                year = str_sub(string = year, start = 1L, end = 4L),
                # make numeric
                year = as.integer(year)) -> TopPharmComp
```

6.4 Format Values

Let's take a look with `glimpse()` to see how these are formatted.

Hide

```
# set width
options(width = 60)
# view transposed data
TopPharmComp %>% glimpse(60)
```

```
## Rows: 231
## Columns: 9
## $ ranking                  <chr> "1", "1", "1", "1", "1"…
## $ company_name             <chr> "Johnson & Johnson", "J…
## $ company_type             <chr> "Pharma", "Pharma", "Ph…
## $ stock_exch               <chr> "NYSE", "NYSE", "NYSE",…
## $ stock_id                 <chr> " JNJ", " JNJ", " JNJ",…
## $ largest_market_cap_us_bil <chr> "397.4 ", "397.4 ", "39…
## $ largest_market_cap_date  <chr> "Jan 2018", "Jan 2018",…
## $ year                     <int> 2019, 2018, 2017, 2016,…
## $ market_cap_us_bil        <chr> "385", "346.1", "375.4"…
```

We can see the `market_cap_us_bil` and `largest_market_cap_us_bil` are formatted as `character` s (but they should be `double`).

6.4.1 Remove whitespace | 6.4.2 Dates | 6.4.3 Factors

Remove the whitespace from `largest_market_cap_us_bil` with `stringr::str_trim()` .

Hide

```
TopPharmComp %>%
  mutate(
    largest_market_cap_us_bil = str_trim(string = largest_market_cap_us_b
il,
                                         side = "both"),
    largest_market_cap_us_bil = as.numeric(largest_market_cap_us_bil))
```

ranking	company_name
<chr>	<chr>
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
2	Roche
2	Roche
2	Roche

1-10 of 231 rows | 1-2 of 9 columns

This is working on the `largest_market_cap_us_bil`. We will also format `market_cap_us_bil` as a numeric value.

Hide

```
TopPharmComp %>%
  mutate(
    largest_market_cap_us_bil = str_trim(string = largest_market_cap_us_b
il,
                                         side = "both"),
    largest_market_cap_us_bil = as.numeric(largest_market_cap_us_bil),
    market_cap_us_bil = as.numeric(market_cap_us_bil))
```

ranking	company_name
<chr>	<chr>
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
1	Johnson & Johnson
2	Roche
2	Roche
2	Roche

1-10 of 231 rows | 1-2 of 9 columns Previous **1** 2 3 4 … 24 Next

Assign this to `TopPharmComp`.

Hide

```
TopPharmComp <- TopPharmComp %>%
  mutate(
    largest_market_cap_us_bil = str_trim(string = largest_market_cap_us_b
il,
                                         side = "both"),
    largest_market_cap_us_bil = as.numeric(largest_market_cap_us_bil),
    market_cap_us_bil = as.numeric(market_cap_us_bil))
```

7 Visualize

We will start by looking at the trends of `market_cap_us_bil` over time (using `year`).

7.1 Labels

I suggest building labels **first** when making a figure or graph, because it forces us to think about what we should expect to see. For example, if we want to see `market_cap_us_bil` on the `y` and `market_cap_year` on the `x`, we can create these with a title using the `ggplot2::labs()` function below.

Hide

```
lab_year_x_mrktcap <- ggplot2::labs(title = "Market Cap Trends Over Time"
,
                                    subtitle = "Largest Pharmaceutical Companie
s",
                                    x = "Year", y = "Market cap (USD in billion
s)")
```

7.2 Map Variables to Positions

The exercises below are a refresher on mapping variables to aesthetics and picking geoms.

7.2.1 exercise | 7.2.2 solution | 7.2.3 exercise | 7.2.4 solution | 7.2.5 exercise | 7.2.6 solution

Map the `year` to the `x` and `market_cap_us_bil` to the `y` and add a `geom_point()`

Hide

```
TopPharmComp %>%
  ggplot(aes(x = ___________, y = ___________)) +
  geom______() +
  lab_year_x_mrktcap
```

7.3 Map Variables to Aesthetics

In the previous graph, we were able to get a different line per company. However, we want to identify more about these companies with color.

We know there are 33 levels in `company_name`, and this is too many to map with color (check this below with `distinct()`)

Hide

```
TopPharmComp %>% distinct(company_name)
```

company_name <chr>
Johnson & Johnson
Roche
Novartis
Merck & Co.
Pfizer
Abbott Laboratories
Amgen
Novo Nordisk
AbbVie
AstraZeneca

1-10 of 33 rows Previous **1** 2 3 4 Next

7.3.1 exercise | 7.3.2 solution

Use the `stock_exch` to color the lines, assign this graph to `gg_trend_line`

Hide

```
gg_trend_line <- TopPharmComp %>%
  ggplot(aes(x = year, y = market_cap_us_bil)) +
  geom_line(aes(group = company_name, color = __________)) +
  lab_year_x_mrktcap
gg_trend_line
```

7.4 Graphing Data Summaries

We're going to summarize the data in `TopPharmComp`. Summarizations are helpful when we're interested in comparing variables across groups.

7.4.1 Graph Labels | 7.4.2 exercise | 7.4.3 solution | 7.4.4 exercise | 7.4.5 solution | 7.4.6 exercise | 7.4.7 solution

We want the `TopPharmComp` dataset to have one variable per `company_name` name and `stock_exch`. A good measure of central tendency for this is the `mean()` (or average). First we define the new labels.

Hide

```
lab_avgmrktcap_compname <- ggplot2::labs(title = "Average Market Cap (201
3-2019)",
                                subtitle = "Largest Pharmaceutical Companie
s",
                                x = "Average Market cap (USD in billions)",
                                y = "Company")
```

The code below groups the `TopPharmComp` data by `company_name` and `stock_exch` and summarizes the mean `market_cap_us_bil`.

Hide

```
TopPharmComp %>%
  group_by(company_name, stock_exch) %>%
  summarize(
    avg_market_cap = mean(market_cap_us_bil, na.rm = TRUE)) %>%
  ungroup()
```

company_name <chr>	stock_exch <chr>
GlaxoSmithKline	NYSE
Teva Pharmaceuticals	NYSE
Abbott Laboratories	NYSE
AbbVie	NYSE
Alexion Pharmaceuticals	NASDAQ
Amgen	NASDAQ
Astellas	TYO
AstraZeneca	NYSE
Bausch Health	TSX
Bayer	FWB

1-10 of 33 rows | 1-2 of 3 columns Previous **1** 2 3 4 Next

This dataset will allow us to show more information per company.

7.5 Labelling Values

We're going to focus on the Largest Market Cap (`largest_market_cap_us_bil`) and Largest Market Cap Date (`largest_market_cap_date`) variables.

7.5.1 Graph Lables | 7.5.2 exercise | 7.5.3 solution | 7.5.4 exercise | 7.5.5 solution

We will define a new set of labels below.

Hide

```
lab_lrg_mrkt_cap_trend <- labs(title = "Largest Market Cap (USD in billio
ns)",
                               subtitle = "Largest Pharmaceutical Companie
s",
                               x = "Largest Market Cap Date",
                               y = "Largest Market Cap (USD in billions)",
                               color = "Company Type")
```

7.6 Adding Layers

We're going to add another layer to this plot that will allow us to plot text values onto their points.

7.6.1 exercise | 7.6.2 solution | 7.6.3 exercise | 7.6.4 solution

Add another layer to the plot above using `geom_text()`

- map `largest_market_cap_date` to the `x`
- map `largest_market_cap_us_bil` to the `y`
- map `stock_id` to `label`
- set `size` to `3` (outside `aes()`)

Hide

```
TopPharmComCaps %>%

  ggplot() +

  geom_point(aes(x = largest_market_cap_date,
                 y = largest_market_cap_us_bil,
                 color = company_type)) +

  geom_text(aes(x = _____________,
                y = _____________,
                label = _____________), size = _) +

  lab_lrg_mrkt_cap_trend
```

7.7 Using Facets

In the next few exercises we're going to see how to use facets to explore values across different levels and scales.

Facets show subplots for different levels of a grouping variable.

7.7.1 exercise | 7.7.2 solution | 7.7.3 exercise | 7.7.4 solution | 7.7.5 exercise | 7.7.6 solution

Use `facet_wrap()` to split the graphs by `company_type`

- set the `scales` argument to `"free"` and `nrow` to `2`.

Hide

```
TopPharmComCaps %>%
  ggplot(aes(x = largest_market_cap_date,
             y = largest_market_cap_us_bil)) +
  geom_point(aes(color = company_type), show.legend = FALSE) +
  geom_text_repel(aes(label = stock_id), size = 3) +

  facet_wrap(. ~ company_type, scales = ______, nrow = _) +

  lab_lrg_mrkt_cap_trend
```

8 Recap

These exercises have covered:

1. Wrangling data with `dplyr`, `stringr` and `janitor`
2. Tidying data with `tidyr`
3. Graphing data using positions, aesthetics, and geoms
4. Using `facet`s to explore graphs across different scales for the `x` and `y` axes