RStudio theme

Adam Shen

About the data

The production and sales of different types of greenhouse flowers and plants has been collected annually since 2007 for both Canada as a whole and its individual provinces (no territories).

More information on this record can be found here.

The data and accompanying metadata can be downloaded together as a .zip file from here.

Packages

```
library(tidyverse)
library(gghighlight)
theme_set(theme_bw())
```

Obtaining the data

For these demos, I will be storing all data files in a folder called `data` with subfolders for each topic.

```
if (!dir.exists("./data/flowers")) {
  dir.create("./data/flowers", recursive=TRUE)
}

download.file(
  "https://www150.statcan.gc.ca/n1/tbl/csv/32100246-eng.zip",
  destfile = "./data/flowers/flowers.zip"
)

unzip(
  "./data/flowers/flowers.zip",
  exdir = "./data/flowers"
)
```

```
list.files("./data/flowers", pattern="csv")
```

Reading in the data

```
flowers <- read_csv("./data/flowers/32100246.csv")
```

```
glimpse(flowers)
```

The variables of interest are:

Variable	Description
`REF_DATE`	Year of record
`GEO`	Country or province
`VALUE`	Number of plants produced or monetary value of plants sold
`Output`	Whether `VALUE` corresponds to production (number) or sales (dollars)
`Flowers and plants`	Flower and plant types

```
flowers <- flowers %>%
  select(REF_DATE, GEO, VALUE, Output, `Flowers and plants`) %>%
  rename(
    year = REF_DATE,
    location = GEO,
    value = VALUE,
    output = Output,
    type = `Flowers and plants`
  )
```

Plant sales in Canada

```
flower_sales_can <- flowers %>%
  filter(location == "Canada", output == "Sales")

ggplot(flower_sales_can, aes(x=year, y=value, colour=type))+
  geom_line()
```

Improving our plot

- Axis titles and legend title should be capitalised
- Legend taking up too much of the plotting area due to excessive information
- Y-axis is in scientific notation which requires the reader to do math
- It is unclear what the y-axis is measuring

Cleaning up the flower types

We can overwrite the existing `flowers` data with these changes since we'll want to use the cleaned data again later. We'll remove the word *Total* with the space that comes after, the space before the square brackets, and the square brackets with all of its contents. Finally, we will capitalise the first letter of each flower type.

```
flowers <- flowers %>%
  mutate(
    type = str_remove(type, pattern="Total\\s"),
    type = str_remove(type, pattern="\\s\\[.*\\]"),
    type = str_to_sentence(type)
  )
```

We can check our work using:

```
flowers %>%
  distinct(type)
```

Now, to re-obtain the data we originally had:

```
flower_sales_can <- flowers %>%
  filter(location == "Canada", output == "Sales")
```

Avoiding scientific notation on the y-axis

We can avoid the scientific notation on the y-axis by simply scaling the values.

```
flower_sales_can <- flower_sales_can %>%
  mutate(value = value / 1e6)
```

Now our y-axis will have the unit of millions of dollars rather than single dollars.

Remaking the plot

We can remove the legend entirely and just label the lines directly on the plot using `gghighlight()`. `gghighlight()` highlights five levels by default. Coincidentally, we have exactly five lines!

```
ggplot(flower_sales_can, aes(x=year, y=value, colour=type))+
  geom_line()+
  gghighlight(label_params = list(alpha=0.6))+
  labs(
    x="Year", y="Value of plants sold (million dollars)",
    title="Total value of plants sold in Canada",
    subtitle="2007 to 2020"
  )
```

Plant production in Canada in 2020

```
flower_prod_can <- flowers %>%
  filter(year == 2020, location == "Canada", output == "Production (number)")

ggplot(flower_prod_can, aes(x=type, y=value))+
  geom_col()
```

Improving our plot

- Axis titles should be capitalised
- Y-axis is in scientific notation which requires the reader to do math
- It is unclear what the y-axis is measuring
- Bars are unordered

Reordering the flower types by number of plants produced

```
flower_prod_can <- flower_prod_can %>%
  mutate(type = fct_reorder(type, value, .desc=TRUE))
```

Avoiding scientific notation on the y-axis

We can avoid the scientific notation on the y-axis by once again scaling the values.

```
flower_prod_can <- flower_prod_can %>%
  mutate(value = value / 1e6)
```

Now our y-axis will have the unit of millions of units rather than single units.

Remaking the plot

```
ggplot(flower_prod_can, aes(x=type, y=value))+
  geom_col()+
  labs(
    x="Plant type", y="Number of plants produced (millions)",
```

```
    title="Number of plants produced in Canada in 2020"
  )
```

Flipping the axes

When making plots for webpages, we can make them as wide as we want. For print, we may be limited to the printing margins. However, in reducing the width of the plot,

- Plot labels on the x-axis may overlap, making them unreadable
- Reducing the font size of the x-axis labels so that they don't overlap can make them difficult to read

The solution here is to flip the axes — put the plant types on the y-axis and their counts on the x-axis.

Reordering the flower types by number of plants produced (again)

If we are making a horizontal bar plot and we want the values to be sorted from largest to smallest, in `fct_reorder()`, we would use the default `.desc=FALSE`.

```
flower_prod_can <- flower_prod_can %>%
  mutate(type = fct_reorder(type, value))
```

Remaking the plot

Since `ggplot2` v3.3.0, we no longer need to build a vertical bar plot and add `coord_flip()` to it. We can now build a horizontal bar plot by supplying the values to the x-aesthetic and the flower types to the y-aesthetic.

```
ggplot(flower_prod_can, aes(x=value, y=type))+
  geom_col()+
  labs(
    x="Number of plants produced (millions)", y="Plant type",
    title="Number of plants produced in Canada in 2020"
  )

# This is a comment
## Documentation
```