R: Visualization

Sakol Suethanapornkul

10 December 2020

Preamble

Thus far, we have looked at several verbs for data manipulation. In this part, we are going to shift gear.

We will talk about data visualization. Specifically, we will look at `ggplot2`, which is part of `tidyverse`.

```r
library(tidyverse)
```

EXERCISE: What purpose does data visualization serve? How can data visualization be effective?

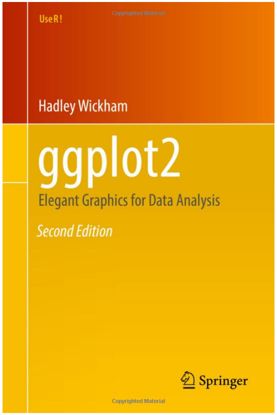
Preamble

Visualization helps us explore data and learn of any systematic patterns between variables that might otherwise go unnoticed.

Sometimes, these patterns may help us generate new hypotheses or shed new light on existing relationships.

ggplot2: Introduction

We will use examples and exercises from this book (3rd edition available online):



Credit: ggplot2

ggplot2: Introduction

A challenging thing about `ggplot2` isn't its codes. Rather, it is the understanding that graphics can be stitched from independent components insofar as grammar is concerned.

If this part sounds confusing to you, think about what painters need to do to make one painting. . .

ggplot2: Introduction

Girl with a Pearl Earring (oil on canvas)

Credit: Wikipedia

`ggplot2`: Introduction

Under the hoods, paintings have several underlying layers. This is true of your graphics, and `ggplot2` utilizes this notion of layers to create complex visualizations.

Now that the notion of layers is revealed, you might wonder how we can create graphics with `ggplot2`?

Basic steps

Step 1: You begin by mapping data to aesthetic attributes. This involves stating (1) what data you want to visualize and (2) which variables in your data frame you want to describe.

Basic steps

Step 1: You begin by mapping data to aesthetic attributes. This involves stating (1) what data you want to visualize and (2) which variables in your data frame you want to describe.

```
ggplot(data = mpg, ...
       )
```

data = mpg maps the data frame `mpg` to data.

EXERCISE: At this stage, what do we get if we run this code?

Basic steps

Step 1: You begin by mapping data to aesthetic attributes. This involves stating (1) what data you want to visualize and (2) which variables in your data frame you want to describe.

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = cty)
       )
```

aes(x = displ, y = cty) maps "engine size" to x position and "city driving" to y position.

Basic steps

Step 1: You begin by mapping data to aesthetic attributes. This involves stating (1) what data you want to visualize and (2) which variables in your data frame you want to describe.

```r
ggplot(data = mpg,
       mapping = aes(x = displ, y = cty)
       )
```

or:

```r
mpg %>%
  ggplot(mapping = aes(x = displ, y = cty)
         )
```

Basic steps

Step 1: You begin by mapping data to aesthetic attributes. This involves stating (1) what data you want to visualize and (2) which variables in your data frame you want to describe.

```
mpg %>%
  ggplot(mapping = aes(x = displ, y = cty)
         )
```

or:

```
mpg %>%
  ggplot(aes(displ, cty)
         )
```

Basic steps

Step 2: After you have set up a “frame,” you add geometric objects (points, bars, lines, etc.) onto it layer by layer. You’ll need at least one layer to complete a plot.

Basic steps

Step 2: After you have set up a "frame," you add geometric objects (points, bars, lines, etc.) onto it layer by layer. You'll need at least one layer to complete a plot.

```
mpg %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point()
```

geom_point() adds "points" as geometric objects to the frame.

Basic steps

Step 3: Now that you have your plot, you can change its scales, break it up into subsets, or alter its look and feel. For example,

```
mpg %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point(aes(color = class) ) +
  labs(x = "Engine size", y = "Miles per gallon") +
  facet_wrap(~ manufacturer) +
  theme_bw()
```

Basic steps

Step 4: Finally, you can save your graphic to your working directory with:

```r
ggsave("scatterplot.png", dpi = 300)
```

Basic steps

To plot with `ggplot2`, you must have data, aesthetic mappings, and geoms.

```r
ggplot(data    = ...,
       mapping = aes(x = ..., y = ...) ) +
  geom_point()
```

Exercise

Now that you know a little bit about a ggplot2 call, describe the data, aesthetics mappings, and layers used for each of the following plots:

```
ggplot(mpg, aes(x = cty, y = hwy) ) +
  geom_point()

ggplot(diamonds, aes(x = carat, y = price) ) +
  geom_point()

ggplot(economics, aes(x = date, y = unemploy) ) +
  geom_line()

ggplot(mpg, aes(x = cty) ) + geom_histogram()
```

Let's pause here...

`ggplot2` contains over 30+ geoms for different kinds of plots (see this site), not to mention a few hundred other functions!

We will focus on a fraction of these geoms to make sure we can transfer our knowledge to other things we can't cover in this lecture!

geom_point()

Previously, we saw this:

```r
mpg %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point()
```

With `geom_point()`, **individual observations** are plotted:

geom_point()

We can add additional information to aesthetic attributes inside aes():

```r
mpg %>%
  ggplot(aes(x = displ, y = cty, color = drv) ) +
  geom_point()
```

geom_point() understands color (or colour), size, and shape aesthetics. So, let's try this one:

```r
mpg %>%
  ggplot(aes(x = displ, y = cty, size = drv) ) +
  geom_point()
```

geom_point()

You can specify aes() at the top level of the plot (with ggplot()) or for each specific geom:

```
mpg %>%
  ggplot(aes(x = displ, y = cty, color = drv) ) +
  geom_point()
```

```
mpg %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point(aes(color = drv) )
```

You'll see why this matters in a few slides!

geom_point()

EXERCISE: What's wrong with the following code?

```
mpg %>%
  ggplot(aes(x = displ, y = cty), color = drv) +
  geom_point()
```

```
mpg %>%
  ggplot(aes(x = displ, y = cty, color = "drv") ) +
  geom_point()
```

geom_point()

For scatterplots with overlapping points, you can add `alpha` to make points more or less transparent:

```
mpg %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point(aes(color = drv), alpha = 0.4)
```

Notice where `alpha = 0.4` lands inside `geom_point()`. Contrast the previous code with this one, where `alpha` is inside `aes()`:

```
mpg %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point(aes(color = drv,
                 alpha = 0.4) )
```

Another layer: `geom_text()`

Recall that in `ggplot2`, different components can be combined layer by layer. This means we can stack one layer on top of another to create complex graphics!

Let's say we want to focus only on cars made by Ford.

Another layer: `geom_text()`

We can do that with `filter()`:

```r
mpg %>%
  filter(manufacturer == "ford")
```

EXERCISE: Now that we have only rows of cars made by Ford, what should we do next?

Another layer: geom_text()

We pipe this data set to ggplot():

```r
mpg %>%
  filter(manufacturer == "ford") %>%
  ggplot(aes(x = displ, y = cty)
         ) +
  geom_point()
```

Another layer: geom_text()

And if we want to label each point with text, we use geom_text():

```
mpg %>%
  filter(manufacturer == "ford") %>%
  ggplot(aes(x = displ, y = cty)
         ) +
  geom_point() +
  geom_text(aes(label = model) ) #label required
```

Check this site for more information about text placement (nudge_x and nudge_y).

Another layer: geom_text()

Previously, we discussed where aesthetic attributes can be placed:

```r
mpg %>%
  filter(manufacturer == "ford") %>%
  ggplot(aes(x = displ, y = cty, color = drv) ) +
  geom_point()
```

```r
mpg %>%
  filter(manufacturer == "ford") %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point(aes(color = drv) )
```

However, we couldn't see any difference with these two code chunks.

Another layer: `geom_text()`

Now with another layer added, where attributes are does matter:

```
mpg %>%
  filter(manufacturer == "ford") %>%
  ggplot(aes(x = displ, y = cty, color = drv) ) +
  geom_point() +
  geom_text(aes(label = model) )
```

```
mpg %>%
  filter(manufacturer == "ford") %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point(aes(color = drv) ) +
  geom_text(aes(label = model) )
```

Aesthetic attributes inside `ggplot()` are passed down to all geoms.

Another layer: geom_smooth()

In many cases, we may want to fit a line that shows a trend in the data. That can be achieved with geom_smooth()

```r
mpg %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point(aes(color = drv) ) +
  geom_smooth()
```

Another layer: geom_smooth()

we can specify a function to use with method and formula:

```
mpg %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point(aes(color = drv) ) +
  geom_smooth(method = lm, formula = y ~ x,
              se = FALSE
              )
```

You can try:

```
geom_smooth(method = lm, formula = y ~ exp(x),
            se = FALSE )
```

Another layer: geom_smooth()

Again, where aesthetic attributes are matters:

```
mpg %>%
  ggplot(aes(x = displ, y = cty) ) +
  geom_point(aes(color = drv) ) +
  geom_smooth(method = lm, se = FALSE)
```

```
mpg %>%
  ggplot(aes(x = displ, y = cty, color = drv) ) +
  geom_point() +
  geom_smooth(method = lm, se = FALSE)
```

Wrap-up

With ggplot2, we can explore data in myriad ways. So far, we have looked at:

```
ggplot(data    = ...,
       mapping = aes(x = .... y = ...)
       ) +
  geom_point(mapping = aes(color = ...,
                           size  = ...,
                           shape = ... )
             ) +
  geom_text(mapping  = aes(label  = ... )
            ) + #or
  geom_smooth(method   = ...,
              formula  = ...,
              se       = FALSE
              )
```