Introduction to ggplot2
SCCR Workshop

Joslin Goh

Statistical Consulting and Collaborative Research Unit
University of Waterloo

2019-01-18
Instructors’ assumptions

Attendees have

- basic understanding of RStudio, or have taken the workshop “Introduction to R” in Fall 2018,
- installed RStudio on their laptops, and
- installed the ggplot2 library.
1.0 About ggplot2

- It is a data visualization package for R and RStudio.
- It is implemented based on Leland Wilkinson’s book, “Grammar of Graphics”.

1.1 Grammar of graphics

Figure 1: The layers of a graph
Throughout this workshop, we will show you how to build a plot layer by layer.

The examples shown in the slides are from the data set diamond from the ggplot2 library.
1.2 The data sets

The diamond data set consists of the price, quality information and physical measurements of different diamonds:

- carat
- cut
- colour
- clarity
- depth
- table
- x
- y
- z
library(ggplot2)
str(diamonds)

Classes 'tbl_df', 'tbl' and 'data.frame': 53940 obs. of
$ carat : num 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...  
$ cut : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...  
$ color : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...  
$ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 2 2 4 2 6 7 3 4 5 ...  
$ depth : num 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...  
$ table : num 55 61 65 58 58 57 57 55 61 61 ...  
$ price : int 326 326 327 334 335 336 336 337 337 338 ...  
$ x : num 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.8 ...  
$ y : num 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.8 ...  
$ z : num 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.5 ...
The diamond data set consists of many data points. To simplify illustration of the different components in ggplot2, we will only use a subset of the data. To select a subset:

```r
set.seed(2019)
my.diamonds <- diamonds[sample(nrow(diamonds), 100),]
```
"What do I want to graph?"

The first step to graphing is to specify the data set and decide what goes on the axes.

Example: Suppose we want to investigate how carat affects the price of a diamond. The two variables (or columns) involved are price and carat. The x-axis is usually the explanatory variable and the y-axis is the dependent (or response variable). In this scenario, price should be on the y-axis and carat on the x-axis.

To initiate this graph in ggplot2, we have

```r
ggplot(my.diamonds, aes(x=carat, y=price))
```
carat

price

0
5000
10000
0.5 1.0 1.5
carat
A blank plot was produced with no points or lines in it. The function does not assume the type of graphs it needs to produce unless it was told. Since this is the first (base) layer which will be used over and over again, it is wiser to save it as an object:

```
p <- ggplot(my.diamonds, aes(x=carat, y=price))
```
The first layer to be added on to the blank plot is a layer of the data points.

In our case, we are interested to make a scatterplot. To add the layer of data points, we add `geom_point` to the initial layer:

```r
p + geom_point()
```
Each layer has its own component. For this layer, the common components include:

- **col**: colour of the points specified using names, rgb specification or NA for transparent colour
- **size**: size of the points specified in milimetres
- **shape**: shape of the points
3.1.1 The colour component

A common way to specify the colour of the points is through the name of the colours. For example, red, darkblue, magenta, chocolate etc.

A complete list of colours can be found at

p + geom_point(col='darkblue')
When col=NA, the points become transparent
Practice 3.1

Setting carat as the x-axis and table as the y-axis, produce the following:
3.1.2 Setting vs mapping

So far, we set the colour of the points to a specific colour of our choice. In some cases, we prefer the colour to change based on the information from another column (usually categorical) in the data set. For example, suppose we want the colour of the points on the graph to change based on cut, which has 5 categories: Fair, Good, Very Good, Premium and Ideal.
This is called mapping.
3.1.3 Changing colour palette

The choice of colour palette is set through the choice of colour palette. When the choice is not mentioned, the default option is used. There are many online packages with pre-set palettes that you can use. We will show you the most common one: Colorbrewer.

Figure 2: Set 1 from ColorBrewer
p1 <- p + geom_point(aes(col=cut))
p1 + scale_colour_brewer(palette = "Set1")
Other ColorBrewer palettes

The ColorBrewer palettes come in three sub-groups:

- Qualitative
- Sequential
- Diverging

All of them can be found at https://jiffyclub.github.io/palettable/colorbrewer/
Colour blind friendly palette

Our preference for a good general purpose solution for palette is to use a colour blind friendly palette such as:

Figure 3: Colour blind friendly palette (grey)

Figure 4: Colour blind friendly palette (black)
Both palettes are not part of ColorBrewer and are extracted from http://jfly.iam.u-tokyo.ac.jp/color/.

They are coded as follow:

```
# Colour blind friendly palette with grey
cbgPalette<-c("#999999", "#E69F00", "#56B4E9", "#009E73",
             "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

# Colour blind friendly palette with black
ccbPalette<-c("#000000", "#E69F00", "#56B4E9", "#009E73",
              "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
```

If you chose to create your own palette, we recommend to have them included at the very top of your R script to so you can use them throughout your project.
With the colour blind friendly palette with grey

\[ p1 + \text{scale\_colour\_manual(values=cbgPalette)} \]
With the colour blind friendly palette with black

```r
p1 + scale_colour_manual(values=cbbPalette)
```
Practice 3.2

Extending from Pratice 3.1, i.e carat on the x-axis and table on the y-axis, map the points according to clarity with a colour palette of your choice.
We used the Accent from ColorBrewer in this example.
3.1.3 The size component

Another component of `geom_point` is the size of the points. They can be changed by either setting or mapping.

The size of the points is specified in millimetres.
Setting the size

```r
p1 + geom_point(size=5)
```
1. All the points are larger in size.
2. The colour of the points are the same. Recall a few slides earlier, we map the colour of the points to cut and saved it.

```r
p1 <- p + geom_point(aes(col=cut))
```

When `geom_point` was called the second time, it overwrites the command to map the colour of the points.

How do we have change the colour and size at the same time?
p + geom_point(aes(col=cut), size=5)
Mapping the size

Similar to mapping the colour component, the size of the points can be mapped to a variable.
Notice in the previous graph that the points are in black colour but the legend still includes cut. This is because we wrote:

```
p1 + geom_point(aes(size=cut))
```

However, due to our previously stored graph done:

```
p1 <- p + geom_point(aes(col=cut))
```

the plot appears “incorrect” and there will be a lot of warnings.
To map both colour and size, we do this:

```r
p + geom_point(aes(col=cut, size = cut))
```
Previously, both the size and colour of the points are mapped to the cut of the diamonds. Suppose we are interested to reflect the change in price due to carat by both cut and color of the diamonds, what should the code look like?
Practice 3.3: Option 2

![Graph showing the relationship between carat and price with different color and cut categories.]

- **Color**: D, E, F, G, H, I, J
- **Cut**: Fair, Good, Very Good, Premium, Ideal
3.1.4 The shape component

Another component to consider is the shape of the points, which are identified using numbers.
The previous graph has its points “set” to an inverse triangles.

What do you think the code should be? *Hint: shape is a component of the geom_point.*

Suppose we want to map the points to the cut of the diamonds, what do you think the code looks like?
p + geom_point(aes(shape=cut), size=5)
You may have received a warning that the shape component is not recommended for ordinal variable such as cut. This is a recommendation. Usually, shape is for nominal variable, i.e. categorical variables that do not have a natural ordering.
A brief summary

- There are three basic components in `geom_point`: colour, size and shape.
- For each component, we can choose to set the points to our preference or map them to a variable.
- It is recommended to include how you want the points to look in one call of `geom_point`. For example,

```
p + geom_point(aes(col= cut, size = cut, shape=cut))
```
4.0 Geometrics

Geometric objects perform the actual rendering of the layer and control the type of plot that you created. For example,

- `geom_point` produces scatterplots
- `geom_line` produces line graphs
- `geom_bar` produces bar plots
4.1 Line graphs

Recall that

\[
p \leftarrow \text{ggplot}(\text{my.diamonds, aes}(x=\text{carat}, y=\text{price}))
\]

Previously we have been drawing scatterplots to draw relationship between carat and price. We used geom_point. What happens if we used geom_line
p + geom_line()
4.1.1 Setting colour

```r
p + geom_line(col = "red")
```
4.1.2 Setting thickness of the line

\texttt{p + geom_line(size=1.5)}
4.1.3 Setting the type of line

\[ p + \text{geom\_line(linetype = "dashed")} \]
4.1.4 Lines with points

\[ p + \text{geom\_line}() + \text{geom\_point}() \]
Practice 4.1

Using \( p \) as a basis, try to reproduce the following graph.
To draw multiple lines, the points must be grouped by a variable. Otherwise, all the points will be connected by a single line.

Suppose we are interested to draw the lines according to cut. Hint: The base layer needs to incorporate the group of the points.
p2 <- ggplot(my.diamonds, aes(x=carat, y=price, group = cut))
p2 + geom_line()
To map the colours of the lines to cut, there are two options:

Option 1:

```r
ggplot(my.diamonds, aes(x=carat, y=price, group = cut)) + geom_line(aes(col=cut))
```

Option 2:

```r
ggplot(my.diamonds, aes(x=carat, y=price, group = cut, col = cut)) + geom_line()
```
To demonstrate the benefit of using the second option, we will use a simulated data set that contains nominal variables.

```
dat1 <- data.frame(
    sex = factor(c("Female","Female","Male","Male")),
    time = factor(c("Lunch","Dinner","Lunch","Dinner"),
                  levels=c("Lunch","Dinner")),
    total_bill = c(12.53, 17.8, 15.24, 17.2)
)

dat1
```

```markdown
<table>
<thead>
<tr>
<th>sex</th>
<th>time</th>
<th>total_bill</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>Lunch</td>
<td>12.53</td>
</tr>
<tr>
<td>Female</td>
<td>Dinner</td>
<td>17.80</td>
</tr>
<tr>
<td>Male</td>
<td>Lunch</td>
<td>15.24</td>
</tr>
<tr>
<td>Male</td>
<td>Dinner</td>
<td>17.20</td>
</tr>
</tbody>
</table>
```
Suppose we are interested to visualize the relationship between time (nominal) and total_bill (continuous) while considering the effect of sex (nominal) using both lines and points.

How do you graph them?
Option 1

```r
ggplot(dat1, aes(x=time, y=total_bill, group=sex)) + geom_line(aes(col=sex)) + geom_point(aes(col=sex))
```
Option 2

```r
ggplot(dat1, aes(x=time, y=total_bill, group=sex, col=sex)) +
  geom_line() + geom_point()
```
Option 2 is more efficient than Option 1.
You can manipulate the components of the line and points as shown earlier.
4.2 Bar graphs

Bar graphs are commonly used to graph categorical variables. For example, the total price of diamonds by cut or color.
The code for producing the earlier bar chart is:

```r
ggplot(my.diamonds, aes(x=color, y= price))+ geom_bar(stat="identity")
```

Notice that the x- and y-axes are similar to that of the scatterplots. The only difference is the use of `geom_bar`.
Suppose we want to add fill colours

\[
\text{ggplot}(\text{my.diamonds, aes}(x=\text{color, y= price, fill=\text{color}})) + \\
\text{geom_bar(stat=\text{"identity"})}
\]
Suppose we want to colour the bars by cut after totalling the price for the diamonds by color, i.e.
ggplot(my.diamonds, aes(x=color, y= price, fill=color)) + geom_bar(stat="identity") + scale_fill_manual(values=cbbPalette)
5.0 Others

Some items of interest to all:

- x- and y-axes labels
- title of the graph
- legends
5.1 Axes labels

Similar to the base package, we can change the labels of the axes by adding the components as follow:

- x-axis: xlab("name")
- y-axis: ylab("name")
p + geom_line(col = "red") + xlab("Price") + ylab("Carat")
To add a title to the graph, we can use `ggtitle`

```r
p + geom_line(col = "red") +
  xlab("Price") + ylab("Carat") +
ggtitle("Relationship between price and carat")
```
Relationship between price and carat
The title are left-centred and can be adjusted through the “theme” layer which we will not covered here.

Personally, the title is not useful since we usually will add caption to the graphs when writing up the results.
5.3 Legends

There are two ways for changing the legends title and labels:

1. Modify the data frame directly.
2. `scale_`xxx_yyy. Refer to
   http://www.cookbook-r.com/Graphs/Legends_(ggplot2)/ for
   the different combinations of xxx and yyy.
Since the legend is related to the colour of the lines, xxx is colour and the variable is categorical, we set yyy to discrete:

```r
p + geom_line(aes(col=cut)) +
  scale_colour_discrete(name="Cut of diamonds",
  breaks=c("Fair", "Good",
            "Very Good", "Premium", "Ideal"),
  labels= c("A", "B", "C", "D", "E"))
```
Thank you!

uwaterloo.ca/scr

The Statistical Consulting and Collaborative Research Unit (SCCR) is the unit through which the Department of Statistics and Actuarial Science provides statistical advice to those working on research problems.