Data Visualization with R - Ggplot2 tutorial

Ariane Ducellier

July 20, 2023

Something about ggplot2 and why we are learning it

Main concepts of Ggplot2

Ggplot2 is made of geometric objects (e.g, lines, bars, points) that are used to visualize data.
Examples of one-dimensional objects for one-dimensional data:

- histogram
- bar-chart

Two-dimensional objects for the relationship between two variables:

Main concepts of Ggplot2

To the geometric objects, we are going to add aesthetics:

- Coordinates scale,
- Fonts,
- Colors...

The aesthetics are described using the grammar of graphics.

Example: Histograms

Built-in R graphics package:

```
hist(airquality$Temp)
```

Quick plot using ggplot2:

```
qplot(airquality$Temp)
```

Ggplot2 command structure

```
ggplot(airquality, aes(x=Temp))
```

dataset, aes = describe the variables from the dataset that we want to visualize and their qualities)
This command does not plot anything.

Ggplot2 command structure

We need to add a command to explain the kind of object that we want to plot:

```
ggplot(airquality, aes(x=Temp)) + geom_histogram()
```

Bar plots

We can use bar plots to visualize one categorical variable:

```
df_desc <- read.csv("../data/
    historical-hourly-weather-data/
    weather_description.csv")
ggplot(df_desc, aes(x=Vancouver)) + geom_bar()
```

The height of the bar is proportional to the number of cases in each group.

Bar plots

Or a combination of a categorical variable and a continuous variable:

```
    ggplot(RetailSales, aes(x=Month, y=Sales)) +
    geom_bar(stat="identity")
```

Using `stat = "identity"` tells ggplot2 to sum the values for each group (Month) and plot bars proportional to the sums.

Box plots

```
ggplot(df_hum, aes(x=month, y=Vancouver)) +
geom_boxplot()
```

Scatter plots and line plots

```
ggplot(df, aes(x=time, y=distance)) + geom_point()
```

```
ggplot(df, aes(x=time, y=distance)) + geom_line()
```

Changing histogram defaults

Modify the number of bins:

```
ggplot(df_hum, aes(x=Vancouver)) +
geom_histogram(bins=15)
```

Modify the filling and the color:

```
ggplot(df_hum, aes(x=Vancouver)) +
geom_histogram(bins=15, fill="white", color=1)
```

Adding aesthetics to the plot

Add title and axis labels:

```
ggplot(df_hum, aes(x=Vancouver)) +
geom_histogram(bins=15, fill="white", color=1) +
ggtitle("Humidity for Vancouver city") +
xlab("Humidity") +
theme(axis.text.x=element_text(size=12),
axis.text.y=element_text(size=12))
```

Adding aesthetics to the boxplot

```
ggplot(df_hum, aes(x=month, y=Vancouver)) +
geom_boxplot(color=1, fill=3) +
ylab("Humidity") +
theme(axis.text.x=element_text(size=15),
axis.text.y=element_text(size=15),
axis.title.x=element_text(size=15, color=2),
axis.title.y=element_text(size=15, color=2))
```

Layers

Each plot can be thought as a separate variable, and the sum of the variables will make the final plot. You can define:

```
p1 <- ggplot(df,
      aes(x=Electricity_consumption_per_capita))
p2 <- p1 + geom_histogram()
p3 <- p1 + geom_histogram(bins=15)
p4 <- p3 + xlab("Electricity consumption per capita")
```

and you can choose to plot p2, p3, or p4.

Scales

Scales `scale_x_continuous` or `scale_x_discrete` can be used to specify the axes. `name`, `limits`, `breaks`, and `labels` are the main parameters that can be adjusted.

```
p1 <- ggplot(df, aes(x=gdp_per_capita))
p2 <- p1 + geom_histogram()
p3 <- p2 + scale_x_continuous(
           name"GDP per capita),
           limits=c(0, 50000),
           breaks=seq(0, 40000, 4000),
           labels=c("0K", "4K", "8K", "12K", "16K",
           "20K",  "24K", "28K", "32K", "36K", "40K"))
```

Polar coordinates

```
t <- seq(0, 360, by=15)
r <- 2
qplot(r, t) +
coord_polar(theta="y") +
scale_y_continuous(breaks=seq(0, 360, 30))
```

Facets

A Trellis display allows creating a plot for each group of a categorical variable:

```
    p <- ggplot(df,
         aes(x=gdp_per_capita,
         y=Electricity_consumption_per_capita)) +
         geom_point()
    p + facet_grid(Country ~ .)
    p + facet_grid(. ~ Country)
    p + facet_wrap(~Country)
```

You can group subplots horizontally, vertically or wrapped together.

Shapes and colors

You can change shape and color for the entire plot:

```
ggplot(df, aes_string(x=var1, y=var2)) +
geom_point(color=2, shape=2)
```

Or assign a different shape and color for each group of a categorical variable:

```
ggplot(df, aes_string(x=var1, y=var2)) +
geom_point(aes(color=Country, shape=Country))
```

Themes

Theme is used to change the non-data elements of the plot:

Theme	Type	Arguments
axis.title.x	element_text	size, color, family, angle
axis.title.y	element_text	size, color, family, angle
plot.background	element_rect	fill, color, linewidth
panel.background	element_rect	fill, fill, color, line width
panel.grid.major	element_line	color, linetype, linewidth

Type `?theme` to show all possible types of themes, their types and their arguments.

Themes

You can add themes to the plot to customize the non-data elements:

```
p1 <- ggplot(dfn, aes(Genre, WorldGross))
p2 <- p1+ geom_bar(aes(fill=LeadStudio),
                   stat="Identity", position="dodge")
p3 <- p2 + theme(axis.title.x=element_text(size=15),
                 axis.title.y=element_text(size=15),
plot.background=element_rect(fill="gray87"),
panel.background = element_rect(fill="beige"),
panel.grid.major = element_line(color="Gray",
                                 linetype=1))
```

Themes

You can also use predefined themes:

```
p2 + theme_bw() + ggtitle("theme_bw()")
p2 + theme_classic() + ggtitle("theme_classic()")
p2 + theme_classic() + ggtitle("theme_gray()")
p2 + theme_minimal() + ggtitle("theme_minimal()")
```

Themes

You can also use define your own theme:

```
mytheme <- theme(legend.title=element_blank(),
legend.position="bottom",
text=element_text(colour="Blue"),
axis.text=element_text(size=12, color="Red"),
axis.title=element_text(size=rel(1.5)))
```

and use it for a single plot:

```
p2 + mytheme + ggtitle("Changed Plot with my theme")
```

or for all the plots by placing it at the beginning of your code:

```
theme_set(my_theme)
```

Themes

You can also change the color palette.
Type ?scale_fill_brewer to see all the color palettes available.

```
p4 <- p2 + theme_bw() + ggtitle("theme_bw()")
p4 + scale_fill_brewer(palette="Spectral")
```