R-intro - Session 4 (part 2 - ggplot2)

João Gonçalves

1 de Agosto de 2018

Contents

Objectives of session 4 (*ggplot2*)

- A brief tour to the `tidyverse`
- Making pretty graphics with `ggplot2` - design principles
- Introduction to main types of plots/graphics:
  - Scatter-plots
  - Regression, trend and smooth lines
  - Histograms
  - Boxplots
  - Modifying plots
  - Making plots by groups

The *ggplot2* package

---

ggplot2 is a powerful data visualization package for the statistical programming language R providing an alternative for making graphs in this software.

The creators of *ggplot2* define it as:

"a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details."

So what is the rationale behind the *grammar of graphics*?

Put simply, a statistical graphic is a mapping of data variables to aesthetic attributes of geometric objects.

So, building on this, the basic idea behind the *grammar of graphics* is to break up graphs into semantic components and thus giving you the ability to independently specify each building block of a plot.

Combining them allows to create just about any kind of graphical display you want.

These building blocks include:

- Data
- Aesthetic mapping (e.g., what to put in x or y, shape, size, colors)
- Geometric object (e.g., points, lines, boxplots)
- Statistical transformations (e.g., scale, log)
- Scales
- Coordinate system (e.g., Cartesian, polar)
- Position adjustments
- Faceting

Before proceeding you need to install the *ggplot2* package:

```
install.packages("ggplot2", dependencies = TRUE)
```

Then load the package using:

```
library(ggplot2)
```

ggplot2 provides two ways to produce plot objects:

- The `qplot()` function: or (q)uick plot which is a straightforward version to make graphics but does not make full use of the power of ggplot2 (and hence not used in this session), and
- The `ggplot()` function: (g)rammar of (g)raphics plot which has a steeper learning curve but allows much more flexibility and power when building graphs (which is the focus of this session)

Let's see the first examples for building a scatterplot.

Making scatterplots

First steps

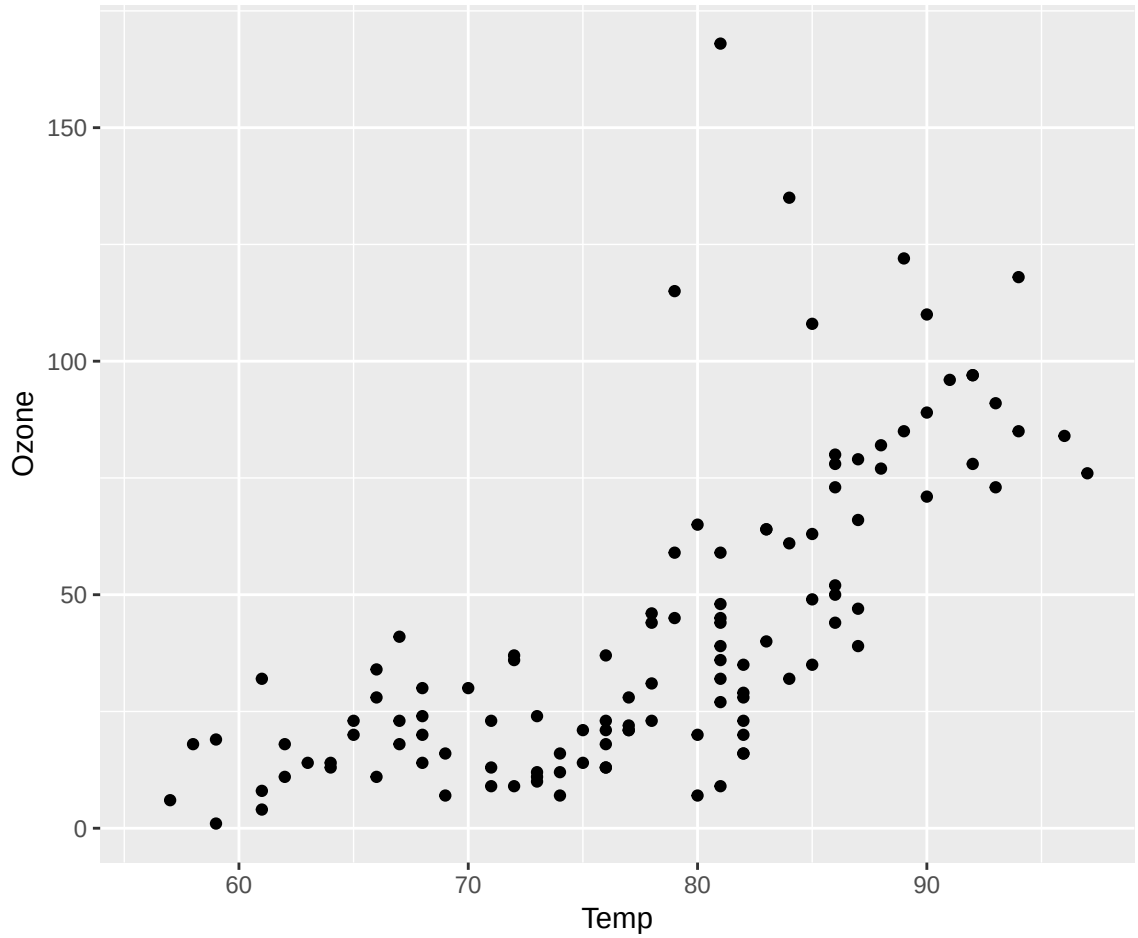
As said before, *ggplot2* works by layering different graphical components.

First we will use `ggplot()` function to create a gg graphical object and define the data and the aesthetic mapping (i.e., describe how variables in the data are mapped to visual properties or aesthetics of geoms)

And, then we will add a point geometry using `geom_point()`

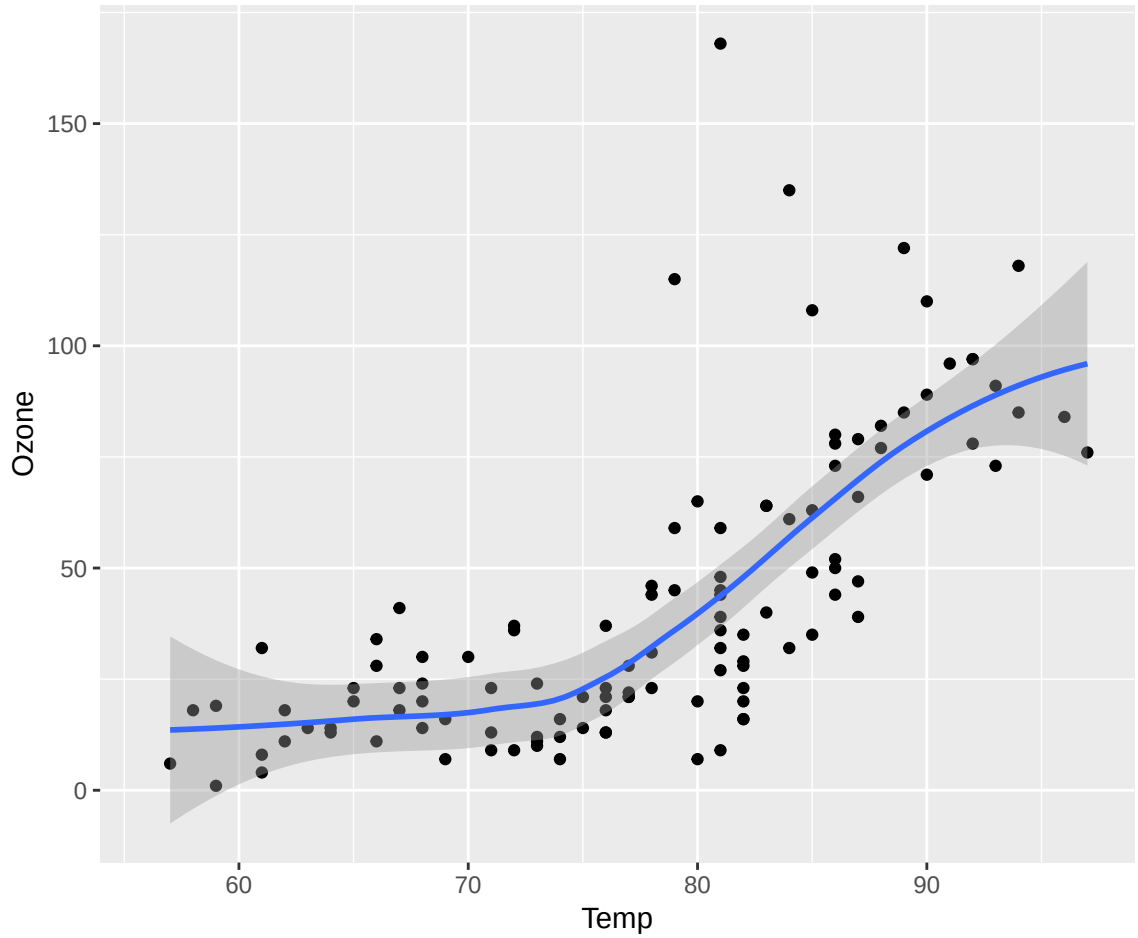
Notice how each new layer is added using the `+` operator

```
g <- ggplot(data = airquality, mapping = aes(x = Temp, y = Ozone)) +
     geom_point()

plot(g)
```



Now we can add a smooth trend line or a regression line between the two variables to have a sense of their association and linearity

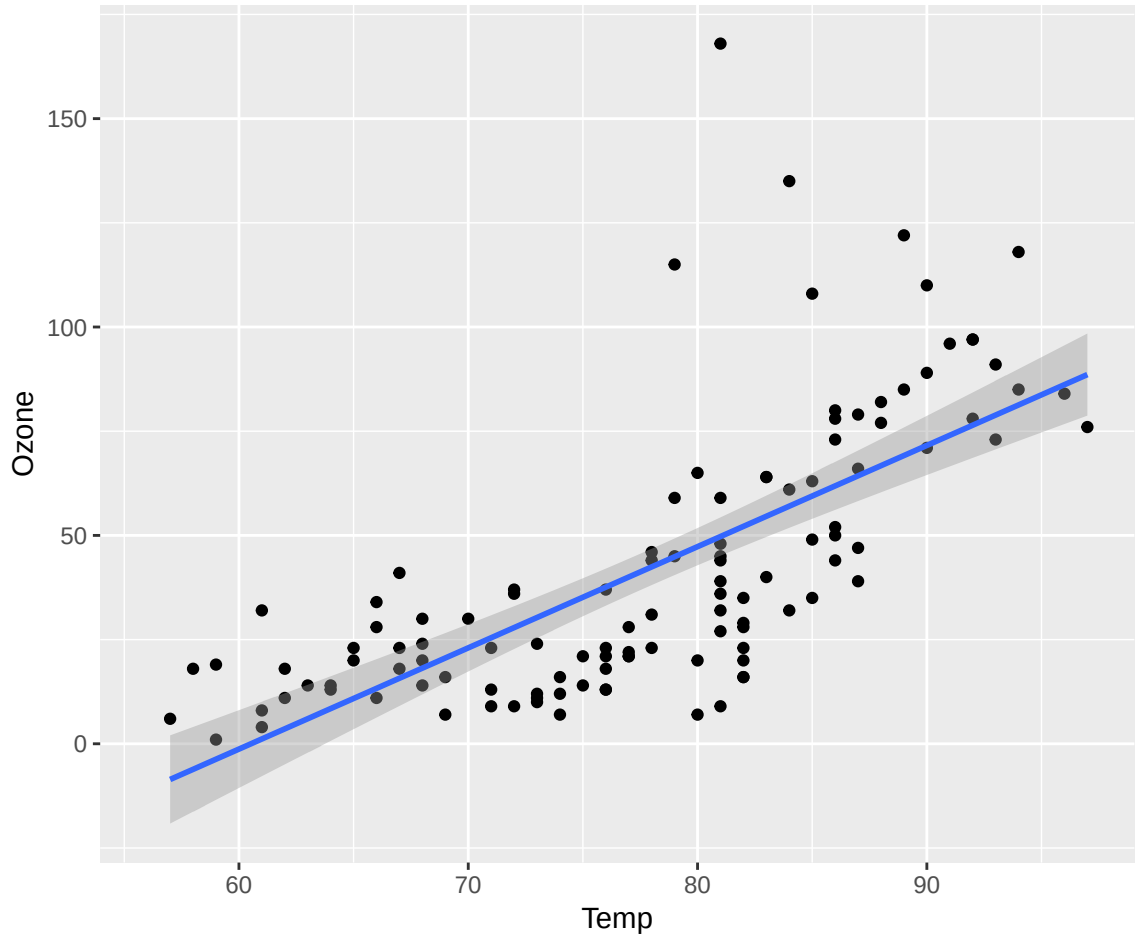
```
g <- ggplot(data = airquality, mapping = aes(x = Temp, y = Ozone)) +
     geom_point() +
     geom_smooth()

plot(g)
```



By default, the `geom_smooth()` function adds a line based on the `loess` smoother which uses locally-weighted polynomial regression.

However, we can modify this to add a 'straight' regression line by modifying the `geom_smooth()` arguments

```
g <- ggplot(data = airquality, mapping = aes(x = Temp, y = Ozone)) +
     geom_point() +
     geom_smooth(method = "lm") # Modify method to use linear regression

plot(g)
```



Exercise 1

a) Using the `airquality` dataset create a scatterplot with a regression line between variables `Wind` speed (in the x-axis) and `Ozone` concentration (in the y-axis).

Modifying and improving plots

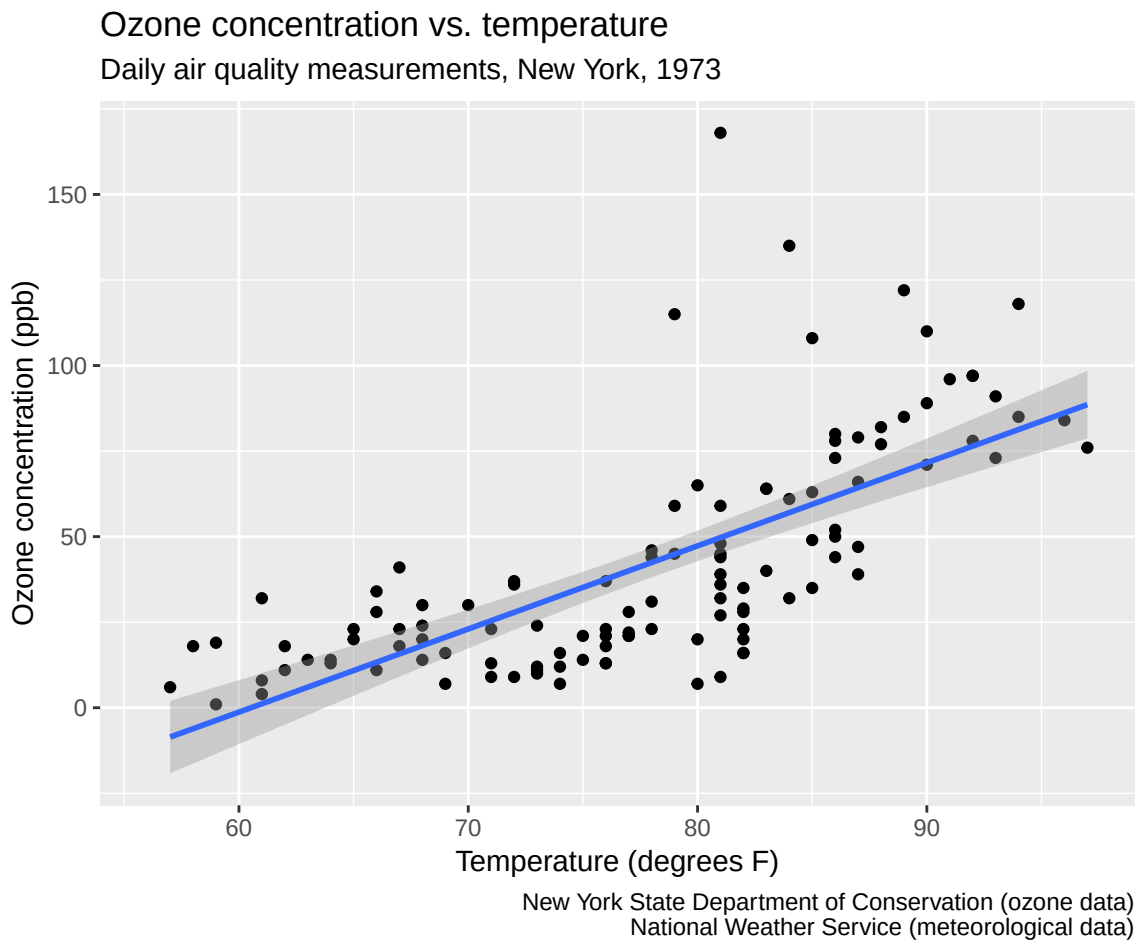
Among other aspects we can define custom labels for each axis and the main title.

Keep in mind that good labels are critical for making your plots accessible and readable to a wider audience! ;-)

For this, we will use the `labs()` function to modify axis, legend, and plot labels.

```
g <- ggplot(data = airquality, mapping = aes(x = Temp, y = Ozone)) +
     geom_point() +
     geom_smooth(method = "lm") +
     labs(x="Temperature (degrees F)",
          y="Ozone concentration (ppb)",
          title="Ozone concentration vs. temperature",
          subtitle="Daily air quality measurements, New York, 1973",
          caption="New York State Department of Conservation (ozone data)\nNational Weather Service (me
```

```
plot(g)
```



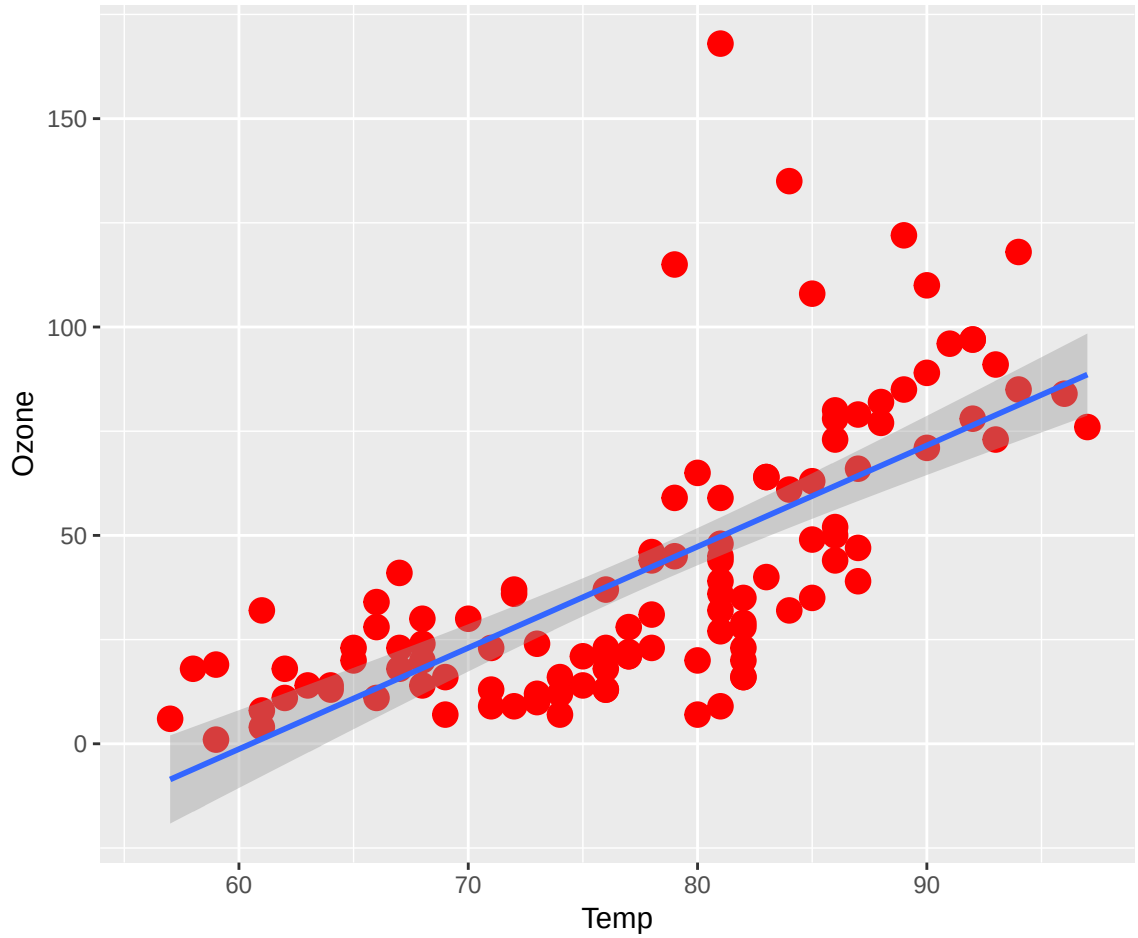
We can also change the color of points (or other geoms) and their size. This can be done in two ways:

i) *"Statically"* - simply by changing the size, color or any other attribute outside the aesthetic mapping function `aes()`;

ii) *"Dynamically"* - if we define a variable to change the color, size, etc. within the aesthetic mapping function `aes()`:

Let's test this with the previous example and using *"static"* color and size changes for points:

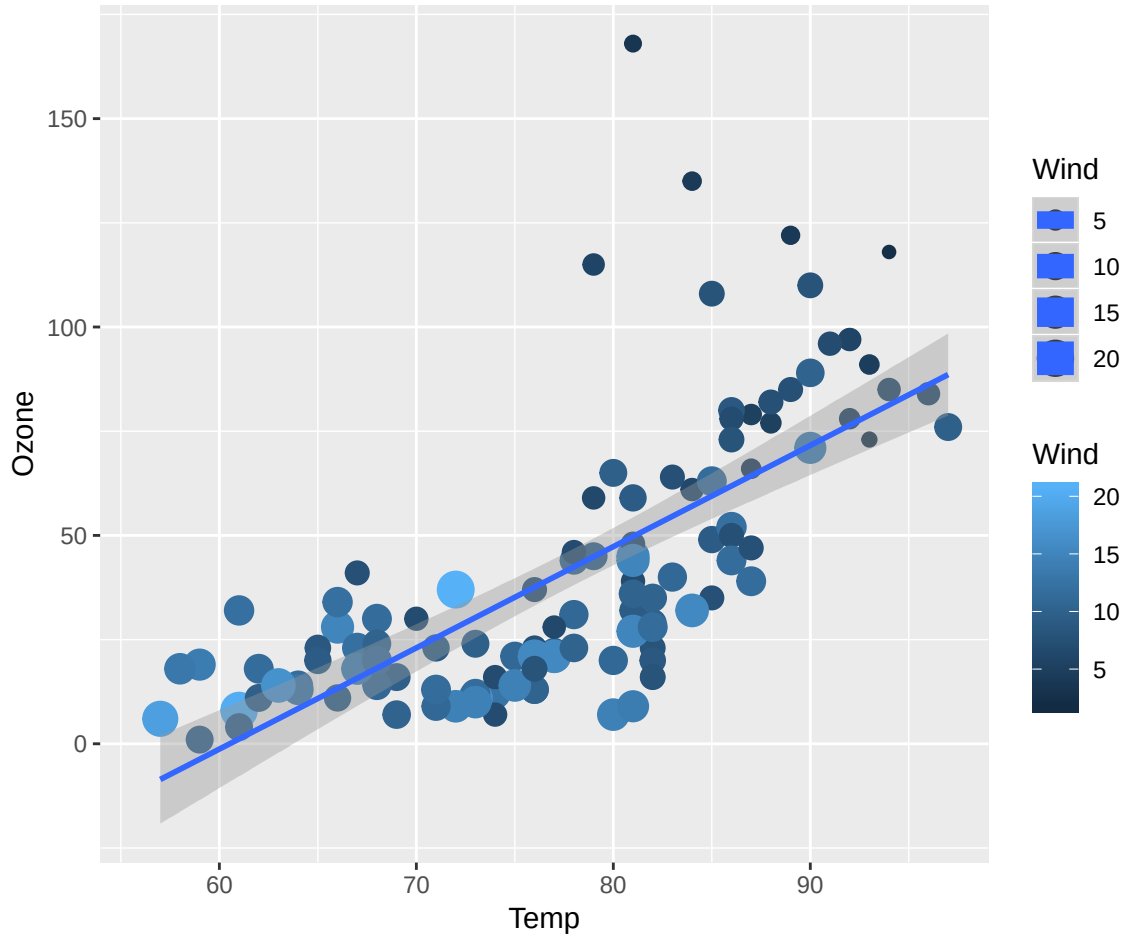
```
g <- ggplot(data = airquality, mapping = aes(x = Temp, y = Ozone)) +
     geom_point(color = "red", size = 4) + # Changed color of points here
     geom_smooth(method = "lm")

plot(g)
```



Notice how all points changed in terms of color and size.

Now let's test what happens when we define a *"dynamic"* color and size change in the `aes` function:

```
g <- ggplot(data = airquality,
            mapping = aes(x = Temp,
                          y = Ozone,
                          color = Wind,    # Changed color here
                          size = Wind)) + # Changed size here
     geom_point() +
     geom_smooth(method = "lm")

plot(g)
```



WOW! :-0 Quite cool, right?

Notice how both the colors (darker is higher wind speed) and the size (bigger is higher speed too) show this third variable in the plot. Almost immediately we can see that lower wind speeds are generally associated to higher ozone concentrations.

Exercise 2

Now that you know how to modify plots let's use it to visualize the `trees` dataset (containing measurements of the *Girth* in inches, *Height* in ft and *Volume* of timber in cubic ft for 31 black cherry trees).

a) Using this data, make a scatterplot to visualize the association between `Girth` (x) and `Height` (y) and including a loess regression line;

b) Using the previous plot, modify the labels to include the correct info for the x and y axes as well as for the main title;

c) Using the previous plot, modify the point size with the `Volume` variable.

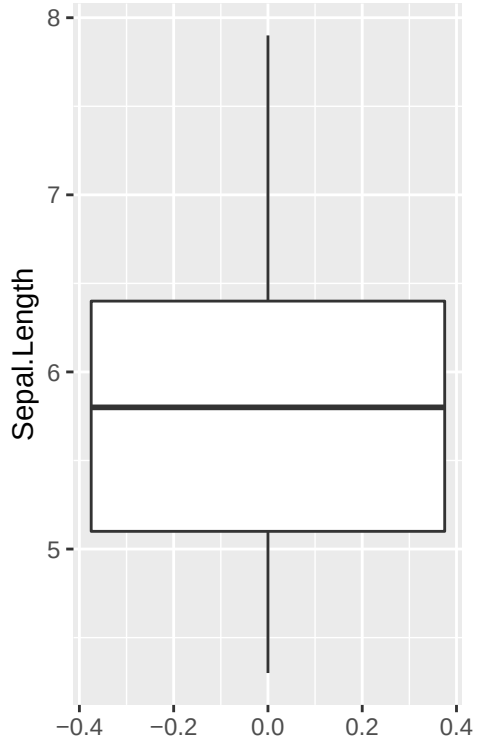
Visualizing distributions

Boxplots, histograms and density plots

In a nutshell, histograms provide an accurate representation of the distribution of numerical data which is often useful for exploratory data analysis (EDA).

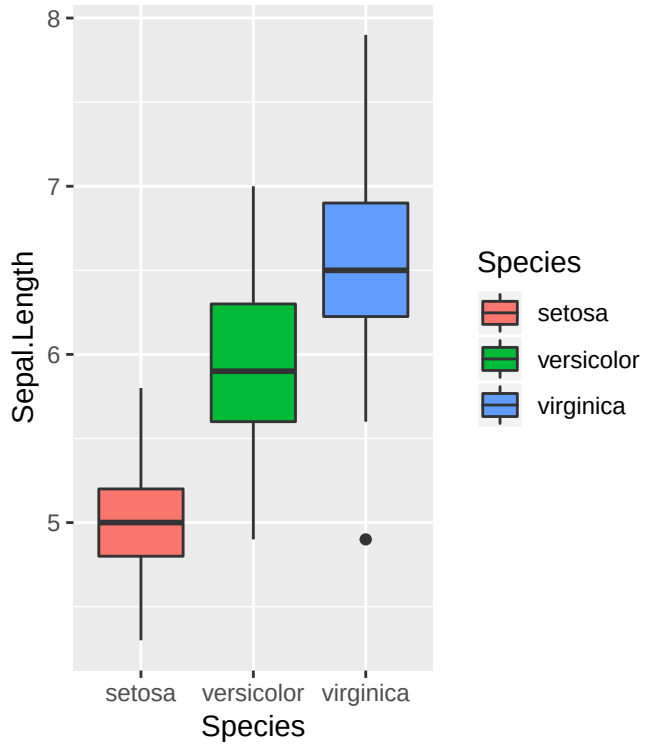
In *ggplot2* making histograms is easy and involves the `geom_boxplot()` function. Let's explore it with the ('good old') iris dataset:

```
# A boxplot for the Sepal length distribution
g <- ggplot(data = iris, mapping = aes(y=Sepal.Length)) +
     geom_boxplot()

plot(g)
```



Hummm... that's nice but what about seeing the distribution of values by different iris species:

```
# A boxplot for the Sepal length distribution
g <- ggplot(data = iris, mapping = aes(
          y=Sepal.Length,
          x=Species,
          fill=Species)) +
     geom_boxplot()

plot(g)
```

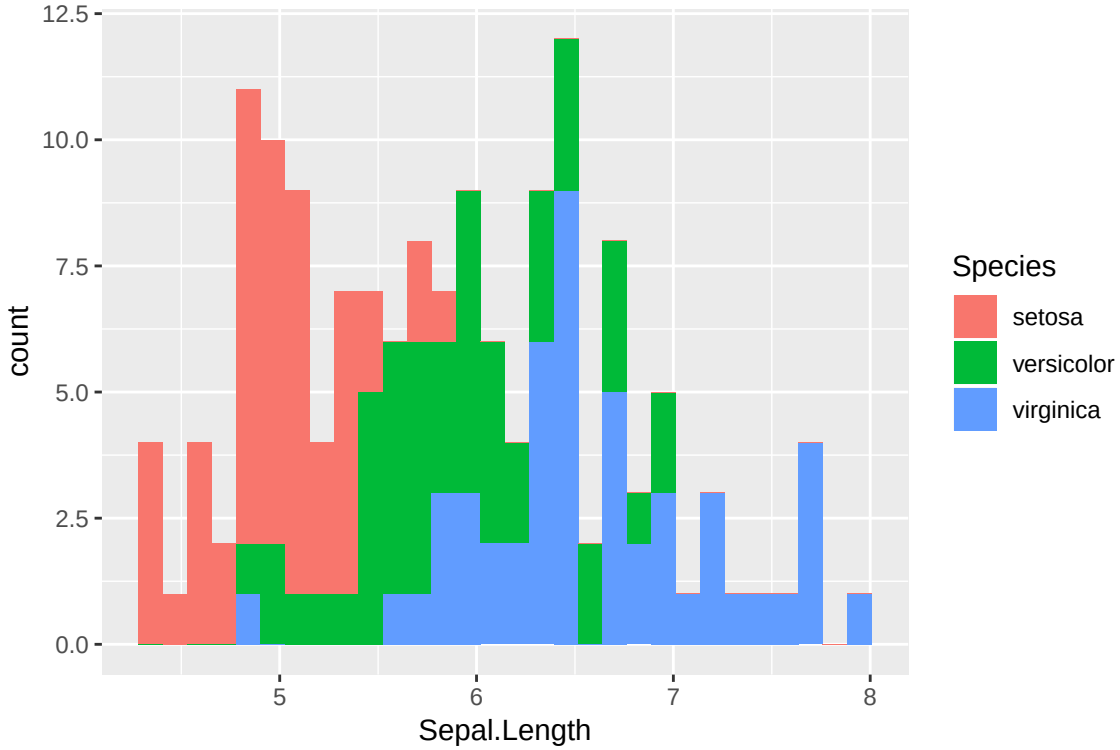


Wow! Much better ;-) Now we can see differences between species in terms of sepal length (or any other variable, it's just a matter of making a few changes).

This happens because we passed a *factor* variable to the `x` and the `fill` parameters which tells ggplot to make groups of geometries.

The same idea can be applied for histograms and density plots, we just have to change the geometry type. See below:

```
# A histogram for the Sepal length distribution
# by species
g <- ggplot(data = iris, mapping = aes(
          x=Sepal.Length,
          fill=Species)) +
     geom_histogram()

plot(g)
```

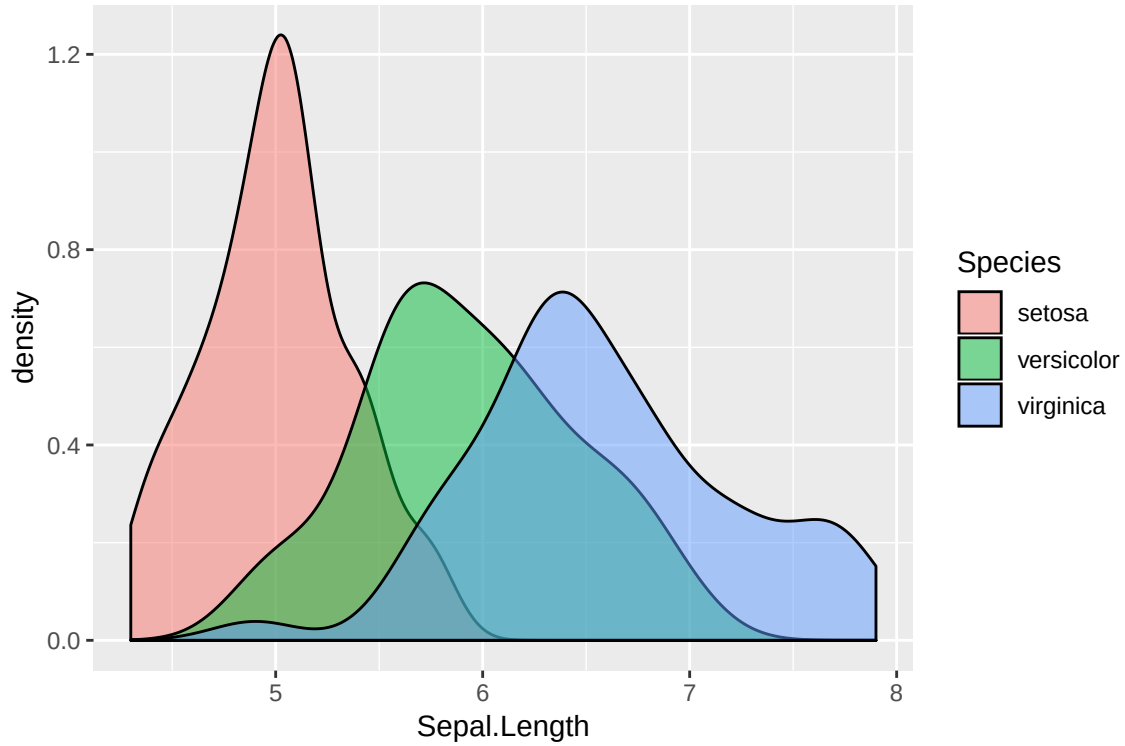


Histograms are nice but when overlapped become a little harder to read...

Better perhaps to explore density plots which try to represent and visualize the distribution of data over a continuous interval by using kernel density estimation/smoothing.

Check out below:

```
# A histogram for the Sepal length distribution
# by species
g <- ggplot(data = iris, mapping = aes(
          x=Sepal.Length,
          fill=Species)) +
     geom_density(alpha=0.5)

plot(g)
```

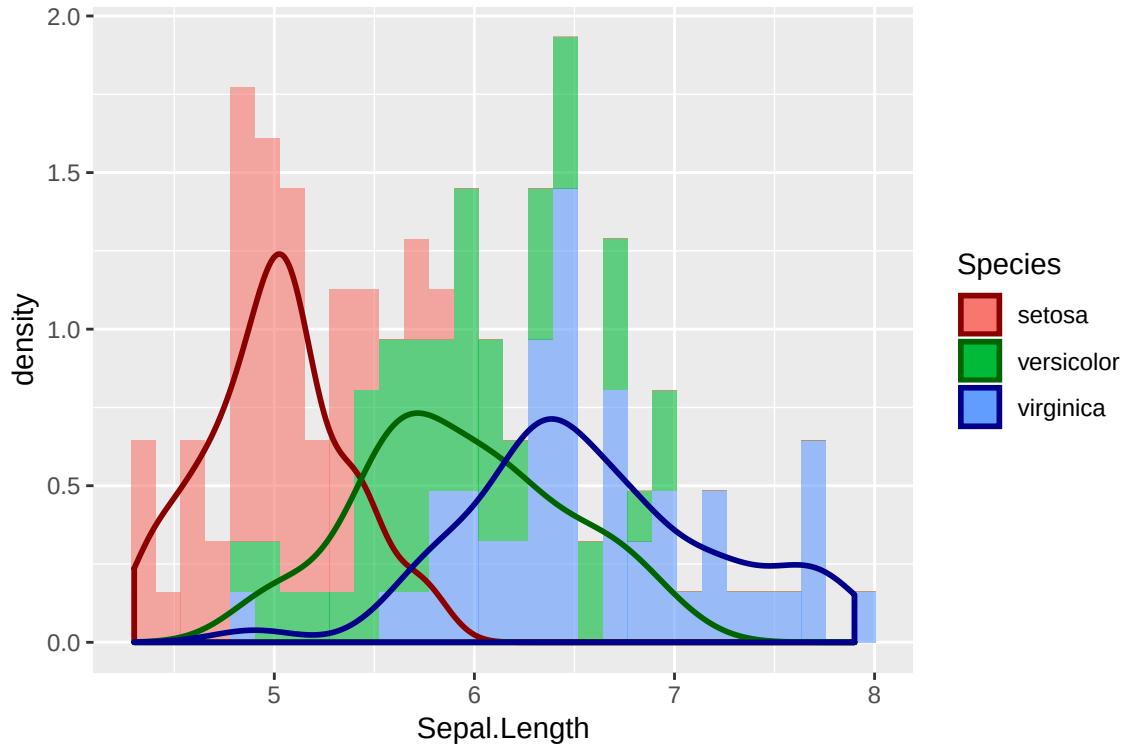


Note how we can see the distribution of sepal length for different species and also its overlap.

The `fill` attribute is used to set a different color for each species dynamically (we used a variable inside `aes`) and the transparency is set using `alpha`.

What about if we combined the two plot/geometry types??:

```
# A histogram for the Sepal length distribution
# by species
g <- ggplot(data = iris) +
     geom_histogram(mapping = aes(x = Sepal.Length, y = ..density.., fill=Species),
                    alpha = 0.6) +
     geom_density(aes(x = Sepal.Length, color=Species),size=1) +
     scale_color_manual(values=c("dark red","dark green","dark blue")) # Change line colors

plot(g)
```



Notice how we used the ggplot term `..density..` inside the histogram that tells this function to calculate not the frequency (which is the default) but the density which can be used to overlap with `geom_density()`.

This plot includes also a customization on the (density) line colors using `scale_color_manual` which allow you to specify your own set of mappings from levels in the data to aesthetic values. This includes colors, labels, scale name, etc. (check `?scale_color_manual` for more info).

Exercise 3

Using the `PlantGrowth` dataset (with results from an experiment used to compare yields, measured by dried weight of plants, obtained under a control and two different treatment conditions) solve the following exercises:

a) Make a histogram with weight distribution;

b) Make a histogram with weight distribution by experimental group;

c) Make a boxplot with weight distribution by experimental group (with different colors for each one);