ggplot
ggplot

- ggplot is a visualization package built-in to R
- Fun fact: the “gg” in “ggplot” stands for “the grammar of graphics”
Source:
http:// kbroman.org/datacarpentry_R_2
016-06-01/04-ggplot2.html
Onto ggplot..

- An R package (like dplyr) used to make visualizations
- Some “baseline” kinds of plots you can make:
  - Bar charts, line charts, scatterplots
ggplot = “grammar of graphics”

- Set of rules for creation of graphics
- Each graphic made up of several independent components
- Each component can be manipulated
- Combine components in a specific way to create graphics
- Not limited to pre-defined graphics types
- Extremely flexible
The grammar’s “parts of speech”

- Data (noun/subject)
- Aesthetic mappings (adjectives)
- Geom (verb)
- Stat (adverb)
- Position (preposition)

For our purposes, we will be focusing on learning how to use the first three.
“Sentences” and “paragraphs”

- All components combine to make a layer (sentence)
- Can places layers on top of each other (paragraph)
- String it all together with + (punctuation)
Advantages of ggplot

- Works well with dplyr!
- Intuitive framework, intuitive syntax
- Flexible enough to make every kind of basic plot, can layer on features to make more complex
Key concepts

- **layers**: We build ggplot2 objects by adding one or more layers together. We do this one at a time, only plotting an object when we are ready. Each layer needs things like data, aesthetics, geometric objects, etc.

- **aesthetics**: The word aesthetics refers to the information in a plot. For example, which variables are associated with the x and y axes? We specify this using the `aes` function.

- **geometric objects**: Geometric objects ("geoms") determine how the information is displayed. For example, will it be a scatter plot or a bar plot? We can specify geoms by adding a layer via functions beginning with `geom_`.

We build up a plot by combining different functions using the `+` operator!
The general process

# 1. define a default data frame and aesthetic

ggplot.object <- ggplot(dataframe, aes(x = var1, y = var2))

# 2. add a layer with the bar geom

ggplot.object <- ggpplot.object + geom_bar()

# 3. show the plot

print(ggplot.object)
Making a bar plot for the iris data set

Using ggplot, let’s examine the relationship between species and average sepal length in the iris data set

**STEP 1:**

We use the `aes` function inside the `ggplot` function to specify which variables we plan to display. We also have to specify where the data are:

```r
bar_plot <- ggplot(iris, aes(x = Species, y = Sepal.Length))
```

So far, all this does is make a ggplot object.
Iris bar chart, cont.

**STEP 2:**

Because we want to make a bar plot, we need to use the `geom_bar` function, to specify that the geometry we want is in bars.

```r
bar_plot <- bar_plot + geom_bar()
```

Note that all we do is “add” (+) the `geom_bar` layer onto the existing plot.
Iris bar chart, cont.

**STEP 3:**

Now we have something to plot:

```python
print(bar_plot)
```
Making a scatterplot with iris data set

Now let’s do it again with a scatterplot! This time, let’s compare sepal length and sepal width.

Again, we use the `aes` function inside the `ggplot` function to create a ggplot object with the variables we plan to display:

```r
scatter_plt <- ggplot(iris, aes(x = Petal.Width, y = Petal.Length))
```

Now, since we want to make it a scatter plot, we use the `geom_point` function.

```r
scatter_plt <- scatter_plt + geom_point()
```
Iris scatter plot, cont.

Now let’s print:

```python
print(plt)
```
Iris scatter plot, cont.

Putting it together: we don't have to build a plot up in separate steps and then explicitly "print" it to the console. If we want to make the plot in one go we can do it like this:

```r
ggplot(iris, aes(x = Petal.Width, y = Petal.Length)) + geom_point()
```
Iris scatter plot, cont.

Next we’ll look at how to customize the axis labels and modify the appearance of the plot. Here we’re adding color to the points, modifying their size and transparency, and adding labels to the plot.

```r
ggplot(iris, aes(x = Petal.Width, y = Petal.Length)) +
  geom_point(colour = "blue", size = 3, alpha = 0.5) +
  labs(x = "Petal Width", y = "Petal Length")
```
Iris scatter plot, cont.

Now let’s color-code it by species!

```r
ggplot(iris, aes(x = Petal.Width, y = Petal.Length, colour = Species)) + geom_point( size = 3, alpha = 0.5) + labs(x = "Petal Width", y = "Petal Length")```
The Minard Map: "The best statistical graphic ever drawn"
The Minard Map:

Step 1: Set-up data

library(dplyr)
library(ggplot2)
library(scales)
library(ggmap)
minard <- read.csv("minard.csv",
                   fileEncoding="UTF-8-BOM", stringsAsFactors=FALSE)
The Minard Map:

Step 2: Some data cleaning (we give this code to you)

```r
# Make sure latitude and longitude are numeric
minard$LONC <- as.numeric(minard$LONC)
minard$LATC <- as.numeric(minard$LATC)

# select relevant columns, rename columns's names and remove NA values
troops <- select(minard, long = LONP, lat = LATP, survivors = SURVIV,
                 city = CITY, direction = DIREC, division = DIVISION)
cities <- select(minard, long = LONC, lat = LATC, city = CITY)
cities <- na.omit(cities)
troops <- na.omit(troops)
```
The Minard Map:

Step 3: Plot the troop path (Hint use the troops dataframe)

You want long to be the x-axis and lat to be the y-axis
The Minard Map:

Step 4: Add `geom_path` to your `ggplot` using “+”

`geom_path (aes(...), data=troops)`

Remember `long` is on the x-axis, `lat` is on the y-axis, `size` is the survivors column, `color` is the direction (to Russia, back to France) and `group` is the division.

Now you have the troops’ path! Yay!
The Minard Map:

Step 5: Now we’re going to add the city names using `geom_text`.

```r
geom_text(aes(long, lat, label = city), size = 3,
          family = "Times", fontface = "italic",
          hjust = 0, vjust = -1, color = "black",
          data = cities)
```
The Minard Map:

Step 6: Now we're going to add the points to the graph using `geom_point()`.

The dataset is `cities`. The x-axis is `long` and y-axis is `lat`. You can use `col="red"` to set your city point color.
The Minard Map:

Step 7: Add some formatting fun to make your map look like Minard’s:

```r
# Format graph
v <- c(1, 2, 3) * 10^5
plot_final <- plot_both + scale_size("Survivors", range = c(1, 10), breaks = v, labels = comma(v)) +
  scale_color_manual("Direction", values = c("#DFC17E", "#252523")) +
  xlab("Longitude") + ylab("Latitude") +
  ggtitle("Napoleon's March to Russia")
plot_final
```
The Minard Map:

Step 8: Let's use ggmap to add a map (Replace russia with the first ggplot from step 3)

```r
russia <- ggmap(
    get_map(location = c(left=22, bottom=53.5, right=39, top=56.5),
             zoom=6, maptype="toner", source="stamen"))
```

Play around with ggmap:
Different maptypes give you different effects ("terrain", "terrain-background", "satellite", "roadmap", and "hybrid" (google maps), "terrain", "watercolor", and "toner" (stamen maps).
You can also use different sources (Google Maps ("google"), OpenStreetMap ("osm"), Stamen Maps ("stamen"), or CloudMade maps ("cloudmade")).