Visualization Lab 1: Introduction to ggplot2

Data Science Team

Instructions: Don't worry about understanding everything perfectly but do ask a member of the teaching team if something is preventing from moving forward. You may work alone or with a partner.

The main goal for this lab is just to make a few graphics using the `ggplot2` package

Loading data

1. First we'll load the dataset that we'll use in this lab:

```
movies <- read.csv("https://web.stanford.edu/class/stats101/data/movies.csv", stringsAsFactors = FALSE)
```

2. `ggplot2` expects data to be in a `data.frame`; confirm that `movies` is one by checking its `class`.

```
class(movies)
```

Loading Packages (a.k.a. Libraries)

Often, we wish to enhance `R` by loading additional packages. You've probably seen quite a bit of this happening in the markdowns already. You can think of packages as collections of functions. Packages usually have specific purposes. For example, in this lab you will use the package `ggplot2` to make plots. While base `R` does have graphics capability, many people find the `ggplot2` syntax more intuitive. `ggplot2` also has different defaults and uses a different "grammar"/syntax from base R graphics, which is designed to encourage good practices.

Another useful packages/library is `dplyr`. It offers a useful way to compactly display the contents of a `data.frame`, but its main function is to help you with data "wrangling" (aka "munging") – transforming data and merging datasets to prepare them for analysis. Libraries need to be loaded in each `RNotebook` to be used (but only need to be once per document).

1. Load the `dplyr` package using the `library` function

```
library('dplyr')
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Did this give you some warning/informational output? Did it concern "masking"? See if you can guess what it means for an object to be "masked" after a package is loaded. Is this something you should worry about when loading a lot of packages?

Now that you've done this, all of the functions of `dplyr` are in R's "namespace" – you can call them just like you call base R functions (such as `library`!)

2. Call the `glimpse` function from dplyr on `movies`

```
## Rows: 2,961
## Columns: 11
## $ title               <chr> "Over the Hill to the Poorhouse", "The Broadway Me~
## $ genre               <chr> "Crime", "Musical", "Comedy", "Comedy", "Comedy", ~
## $ director            <chr> "Harry F. Millarde", "Harry Beaumont", "Lloyd Baco~
## $ year                <int> 1920, 1929, 1933, 1935, 1936, 1937, 1939, 1939, 19~
## $ duration            <int> 110, 100, 89, 81, 87, 83, 102, 226, 88, 144, 172, ~
## $ gross               <int> 3000000, 2808000, 2300000, 3000000, 163245, 184925~
## $ budget              <int> 100000, 379000, 439000, 609000, 1500000, 2000000, ~
## $ cast_facebook_likes <int> 4, 109, 995, 824, 352, 229, 2509, 1862, 1178, 2037~
## $ votes               <int> 5, 4546, 7921, 13269, 143086, 133348, 291875, 2153~
## $ reviews             <int> 2, 107, 162, 164, 331, 349, 746, 863, 252, 119, 33~
## $ rating              <dbl> 4.8, 6.3, 7.7, 7.8, 8.6, 7.7, 8.1, 8.2, 7.5, 6.9, ~
```

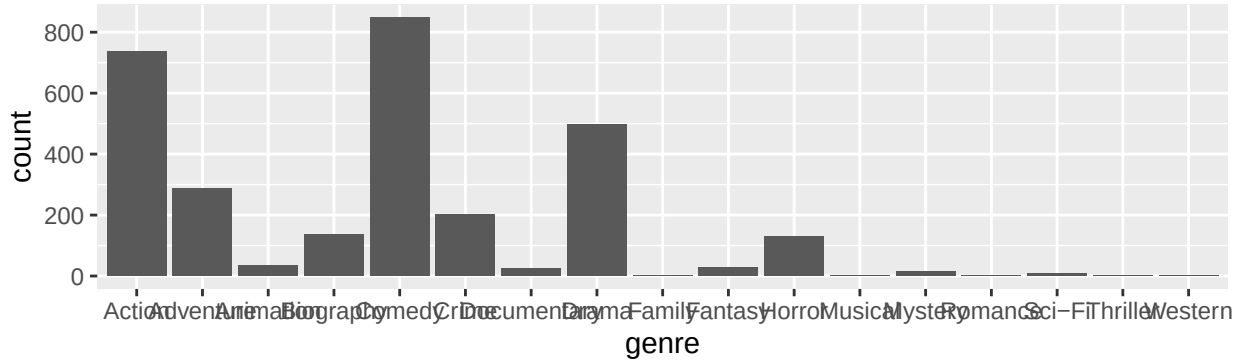
What did this do? Take a minute or two to see if you can determine what the output from `glimpse` means. You may want to click the "play" button next to `movies` in the Environment tab (top right), which will pop open a viewer tab that allows you to browse through the data as though it were in a spreadsheet. You can compare that to the output of `glimpse` to get a sense of what information `glimpse` gives you.

ggplot basics

A few things to know about `ggplot`

- The main worker function of `ggplot2` is `ggplot`
- `ggplot` is basically a function that takes a `data.frame` and other inputs (known as "arguments") and creates a graphic display of the data
- The arguments you pass it determine which variables (columns of the data frame) are being plotted and what type of plot is made
- The function `ggplot` can make many different plot types, and you control the plot type by the way you call the function
- One important argument is `aes` (as in 'aesthestics'), which is used to specify which variables are plotted
- Additional features are added (literally using `+`) to layer on additional plot features. A basic plot feature that all plots need is called the "geometry" – basically the plot type. For example, `geom_scatter()` is used to create scatter plots.
- Here is an example that makes a bar plot

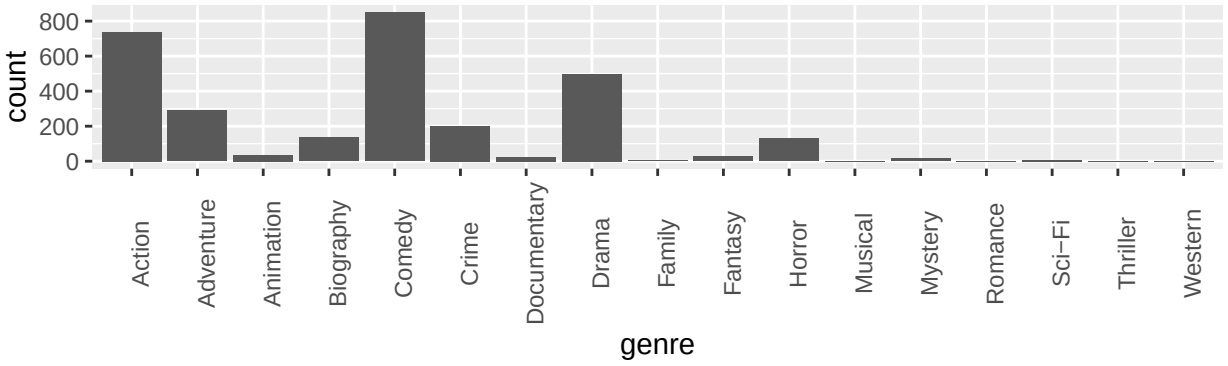
```
library(ggplot2)
ggplot(movies, aes(x=genre)) + geom_bar()
```



That's the form of a basic call to `ggplot`.

Notice something about the horizontal axis labels? They're all overlapping and can't be read. Let's fix this by adjusting the plot's `theme`

```
ggplot(movies, aes(x=genre)) + geom_bar() +
  theme(axis.text.x = element_text(angle=90))
```

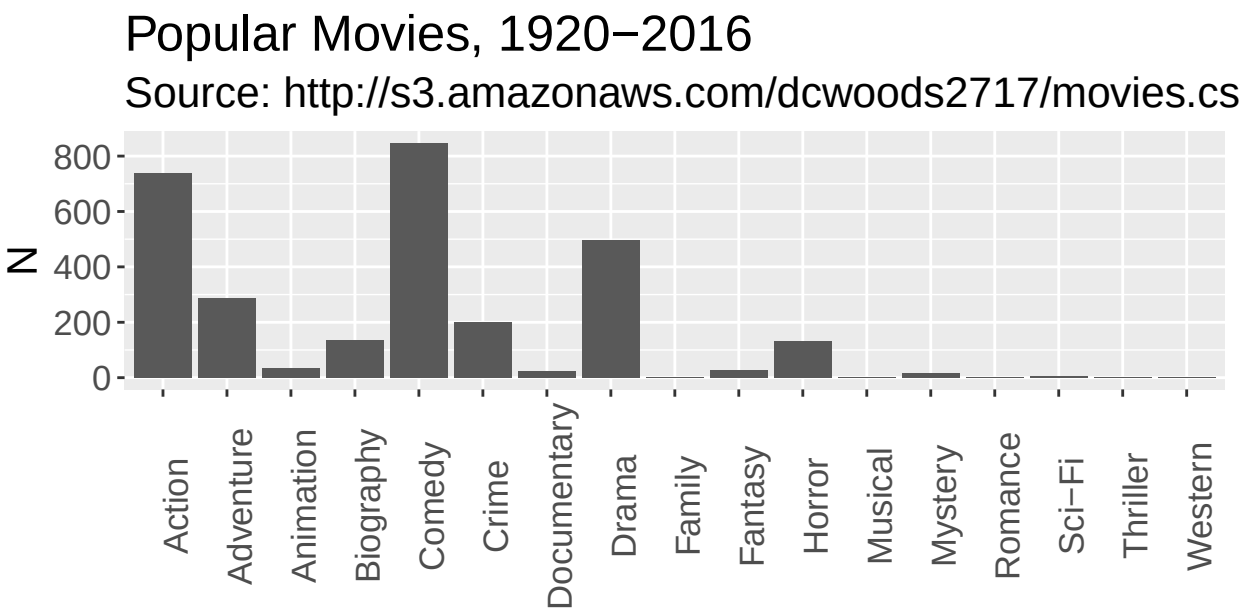


That's better, no?

gg coding style

- ggplots are often saved as objects to which additional features are added one or two at a time
- you do this by assigning (there we go again with assignment) the output of ggplot to a variable, then "adding" some additional features to that variable
- Basically, the operator `+` is interpreted differently by R when it is called on a ggplot object. Rather than interpreting it as "add these two numbers", R interprets it as "add these features to this plot"
- Here is an example. You can step through one line at a time, then run `print(g)` in the console to see what the intermediate plots look like

```
g <- ggplot(movies, aes(x=genre)) + geom_bar()
g <- g + theme(text = element_text(size=16),
               axis.text.x = element_text(angle=90))
g <- g + labs(x="", y="N", title="Popular Movies, 1920-2016",
              subtitle="Source: http://s3.amazonaws.com/dcwoods2717/movies.csv")
print(g)
```



Tasks

1. Make a histogram with ggplot of any continuous variable found in `movies` using `geom_histogram()`. (If you don't know what `continuous` means in this context, just ask!) You should use the fig.height and fig.width arguments in the `{r}` tag to prevent your figures from being too huge, as in the empty block below.

```
g <- ggplot(movies, aes(x = ))
```

If you haven't seen a histogram before, can you reason a bit about what this is showing you? Hint: the height of the bars is proportional to frequency. We'll soon explain histograms in detail.

2. Make a few different versions of the plot by altering `bins` or `binwidth` (parameters of `geom_histogram`) and/or by placing your `x` variable on a log scale

3. Pick the version of the graph you feel best represents the data and format the labels, titles, and so on such that the context is clear and everything is readable. Now put your final plot here

4. Now create a plot of some continuous variable over time, i.e., using `geom_point()`, make `year` the `x` variable and choose another continuous variable to be `y`. Also, add a "trendline" with `geom_smooth()` (the term "trend line" may have originated with financial "technical analysis" and made its way into excel. I don't find the name very appropriate, but it appears we are stuck with it)

Format the graph and add a sentence explaining whether or not there appears to be a trend over time.

Do you have any guesses as to what `geom_smooth()` is doing? We don't at all expect you to know, but it's nice to be able to reason a little about what a "trendline" means in this case. The name of the function gives you some kind of clue...

What do you think the gray bands around `geom_smooth()` are supposed to represent?

5. Choose a categorical variable to be `x` and a continuous variable to be `y`. Make two plots, one with `geom_boxplot()` and the other with `geom_violin()`. You may again want to use a log scale.

What do you think these plots are showing you? What are the horizontal lines, boxes, and "whiskers" about in the boxplot? Which do you prefer and why? Again, we don't expect you to know this, but if you think about it a bit now then it may make more sense when we formally introduce these plot types.

Adding ggplot features

First, let's load the movies data from the last lab and some libraries that we will use. By convention, libraries are listed at or near the top of the file.

```
movies <- read.csv("https://web.stanford.edu/class/stats101/data/movies.csv", stringsAsFactors = FALSE)
library(ggplot2)   # by convention, libraries are listed at or near the top of RNotebooks
library(dplyr)
glimpse(movies)
```

```
## Rows: 2,961
## Columns: 11
## $ title               <chr> "Over the Hill to the Poorhouse", "The Broadway Me~
## $ genre               <chr> "Crime", "Musical", "Comedy", "Comedy", "Comedy", ~
## $ director            <chr> "Harry F. Millarde", "Harry Beaumont", "Lloyd Baco~
## $ year                <int> 1920, 1929, 1933, 1935, 1936, 1937, 1939, 1939, 19~
## $ duration            <int> 110, 100, 89, 81, 87, 83, 102, 226, 88, 144, 172, ~
## $ gross               <int> 3000000, 2808000, 2300000, 3000000, 163245, 184925~
## $ budget              <int> 100000, 379000, 439000, 609000, 1500000, 2000000, ~
## $ cast_facebook_likes <int> 4, 109, 995, 824, 352, 229, 2509, 1862, 1178, 2037~
## $ votes               <int> 5, 4546, 7921, 13269, 143086, 133348, 291875, 2153~
## $ reviews             <int> 2, 107, 162, 164, 331, 349, 746, 863, 252, 119, 33~
## $ rating              <dbl> 4.8, 6.3, 7.7, 7.8, 8.6, 7.7, 8.1, 8.2, 7.5, 6.9, ~
```

Context

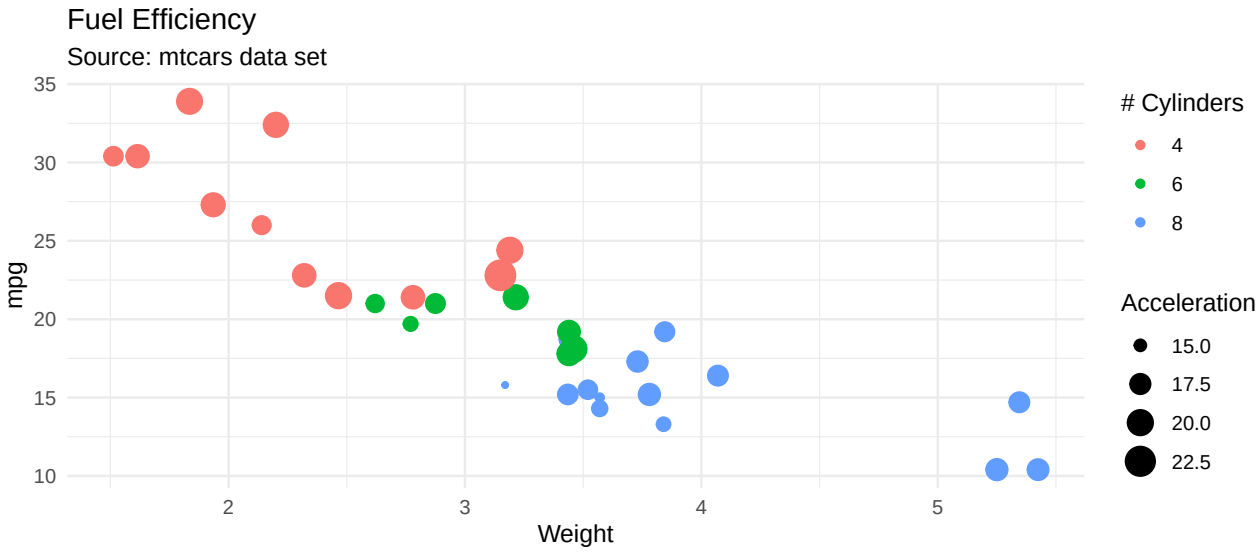
As discussed in the lecture, graphics should be kept free of clutter and visual distractions. However, for more complex data it is sometimes best to make use of additional features like color and shape and size.

Example: Scatter Plot with Four Variables

A single graphic displaying fuel economy (mpg) by cylinders (cyl), weight (wt), and acceleration (qsec). We will put weight on the horizontal axis and mpg on the vertical axis. We still need to visualize the number of cylinders and acceleration. Let's visualize the number of cylinders by color and the acceleration by the size of the dot. ggplot makes it easy to do so:

```
mtcars$cyl <- as.factor(mtcars$cyl)  # turns cyl into a categorical variable

ggplot(mtcars, aes(x=wt, y=mpg, size=qsec, col=cyl)) + geom_point() +
  labs(x="Weight", col="# Cylinders", size="Acceleration",
       title="Fuel Efficiency", subtitle="Source: mtcars data set") + theme_minimal()
```



Task

Create a similar plot where `x` and `y` are continuous variables found in `movies` and color and size show additional information. Be sure to label your plot appropriately. You may wish to create a new variable or subset. Especially for more than one criteria, `filter` (found in `library(dplyr)`) is recommended. For example, suppose we are interested in movies that are either action movies or comedies produced after the year 2000, and one variable that we want to visualize is the profit that each movie made. We can extract this subset of movies as follows:

Task

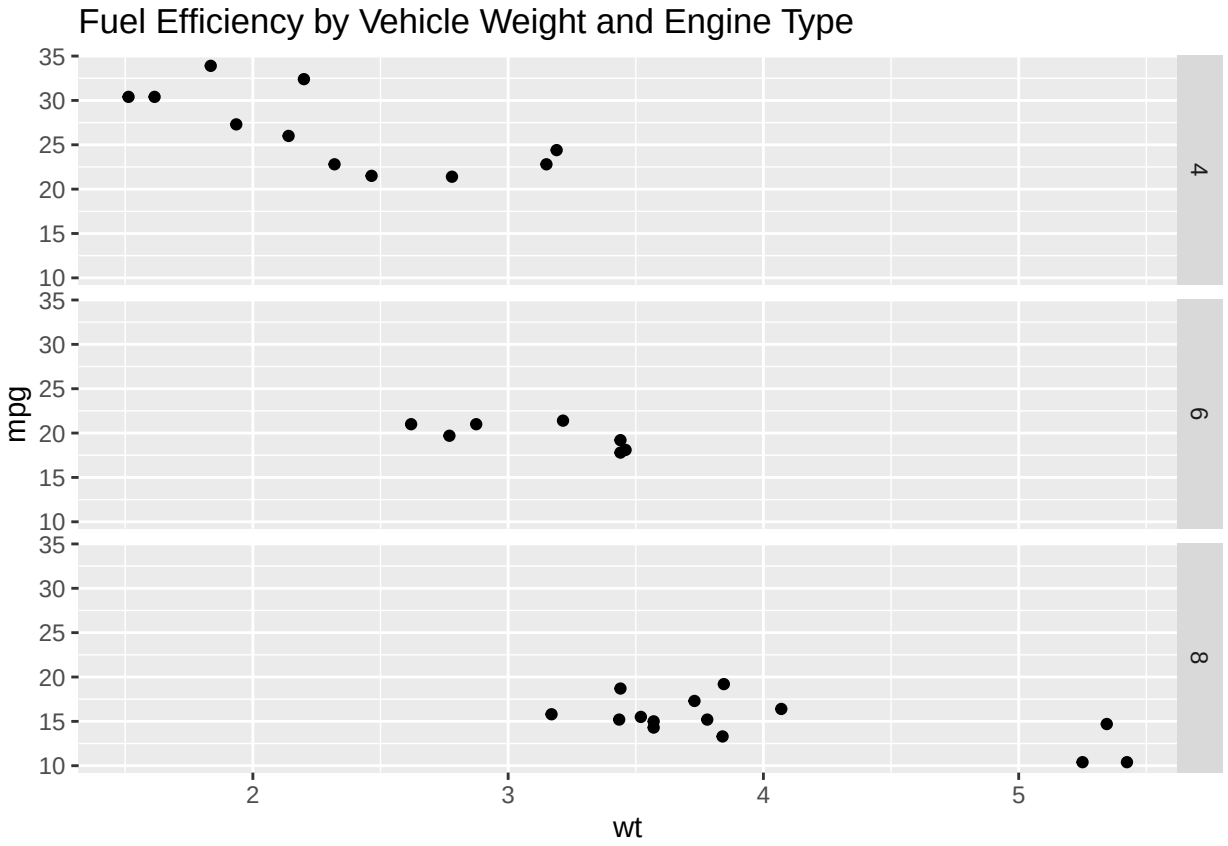
Create a density plot with `geom_density` (i.e., using a single continous `x` variable found in your version of `movies`).

Next, set `col` (color inside of `aes`) to be equal to a categorical variable. (You may need to use `as.factor` and you may wish to subset or simplify the categorical variable first.) How do the plots compare? What additional information does the latter plot contain?

Example: Faceting

Sometimes we wish to be create a set of similar plots.

```
ggplot(mtcars, aes(wt, mpg)) + geom_point() +
  facet_grid(as.factor(cyl) ~ .) +
  labs(title="Fuel Efficiency by Vehicle Weight and Engine Type")
```



Task

Using your version of `movies`, create three `facet_grid` plots using different statistics such as `geom_point`, `geom_density`, `geom_density2d`, and `geom_bar`.