Lecture 03 - Friday Demo

Wesley Burr

September 14, 2018

The Example of the Day: The World Cup

```
library(tidyverse)
```

The FIFA World Cup is a global football competition contested by the various football-playing nations of the world. It is contested every four years and match data from each game is maintained on the FIFA World Cup Archive website.

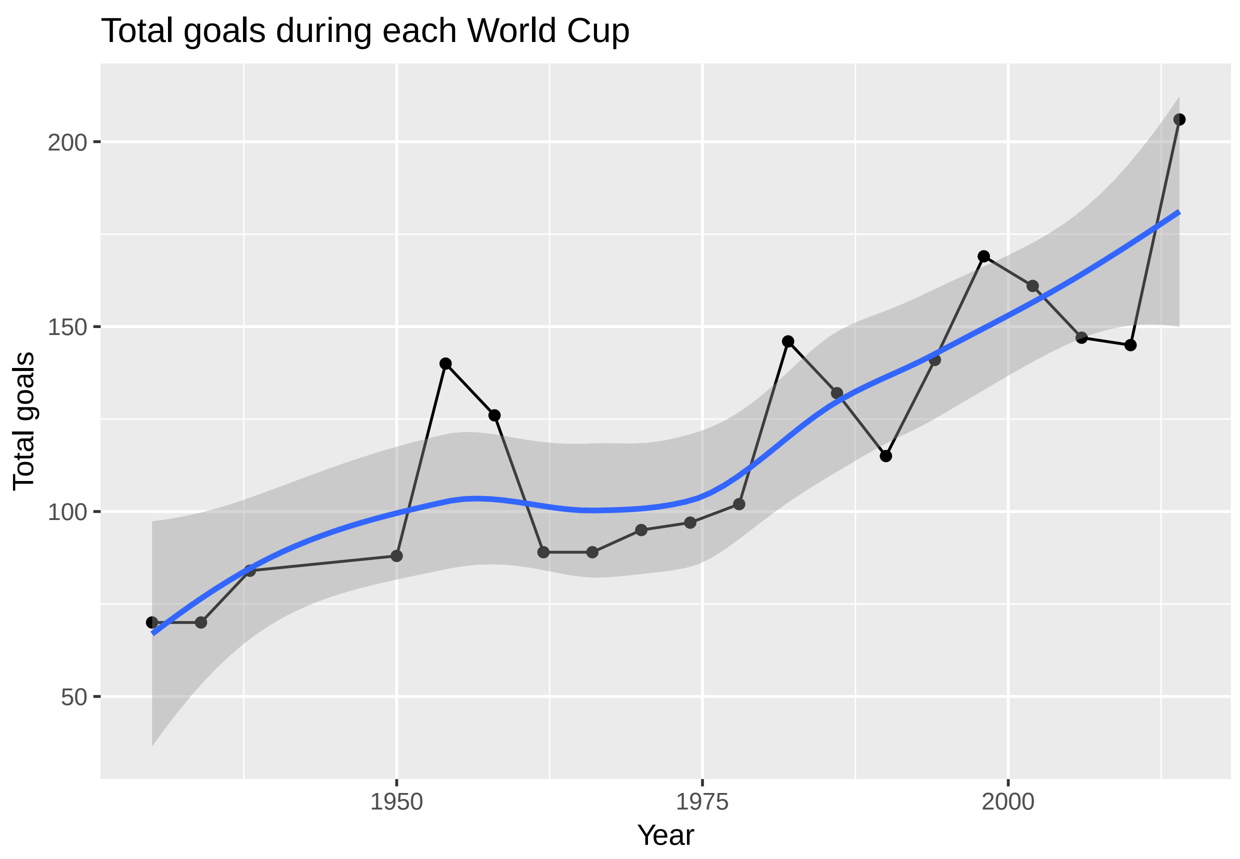
```
wc <- read_csv("http://webwork.trentu.ca/worldcup.csv")
```

The figure below shows the total number of goals scored during world cup games between 1930 and 2014. We can see that the total number of goals have been increasing over the year.

```
wc_tots <- wc %>%
  mutate(total_goals = home_team_goals + away_team_goals) %>%
  group_by(year) %>%
  summarise(total_goals_year = sum(total_goals))
```

Total Goals Per Year

```
ggplot(data = wc_tots, aes(x = year, y = total_goals_year)) +
    geom_point() +
    geom_line() +
    geom_smooth() +
    labs(
      x = "Year", y = "Total goals",
      title = "Total goals during each World Cup"
    )
```



Variables

We talked this week about variable types, and in workshop you explored how you might check which variable type something is. Let's examine this World Cup data in a similar way.

```r
head(wc)
```

```
## # A tibble: 6 x 20
##     year datetime stage stadium city  home_team_name home_team_goals
##    <int> <chr>    <chr> <chr>   <chr> <chr>                    <int>
## 1  1930 13 Jul ~ Grou~ Pocitos Mont~ France                       4
## 2  1930 13 Jul ~ Grou~ Parque~ Mont~ USA                          3
## 3  1930 14 Jul ~ Grou~ Parque~ Mont~ Yugoslavia                   2
## 4  1930 14 Jul ~ Grou~ Pocitos Mont~ Romania                      3
## 5  1930 15 Jul ~ Grou~ Parque~ Mont~ Argentina                    1
## 6  1930 16 Jul ~ Grou~ Parque~ Mont~ Chile                        3
## # ... with 13 more variables: away_team_goals <int>, away_team_name <chr>,
## #   win_conditions <chr>, attendance <int>, half_time_home_goals <int>,
## #   half_time_away_goals <int>, referee <chr>, assistant_1 <chr>,
## #   assistant_2 <chr>, round_id <int>, match_id <int>,
## #   home_team_initials <chr>, away_team_initials <chr>
```

The **head()** command displays the "head" (top!) of a set of data. By doing this, we see the variable types immediately: **year** is an integer, **datetime** is a character, through to **home_team_goals** being an integer as well.

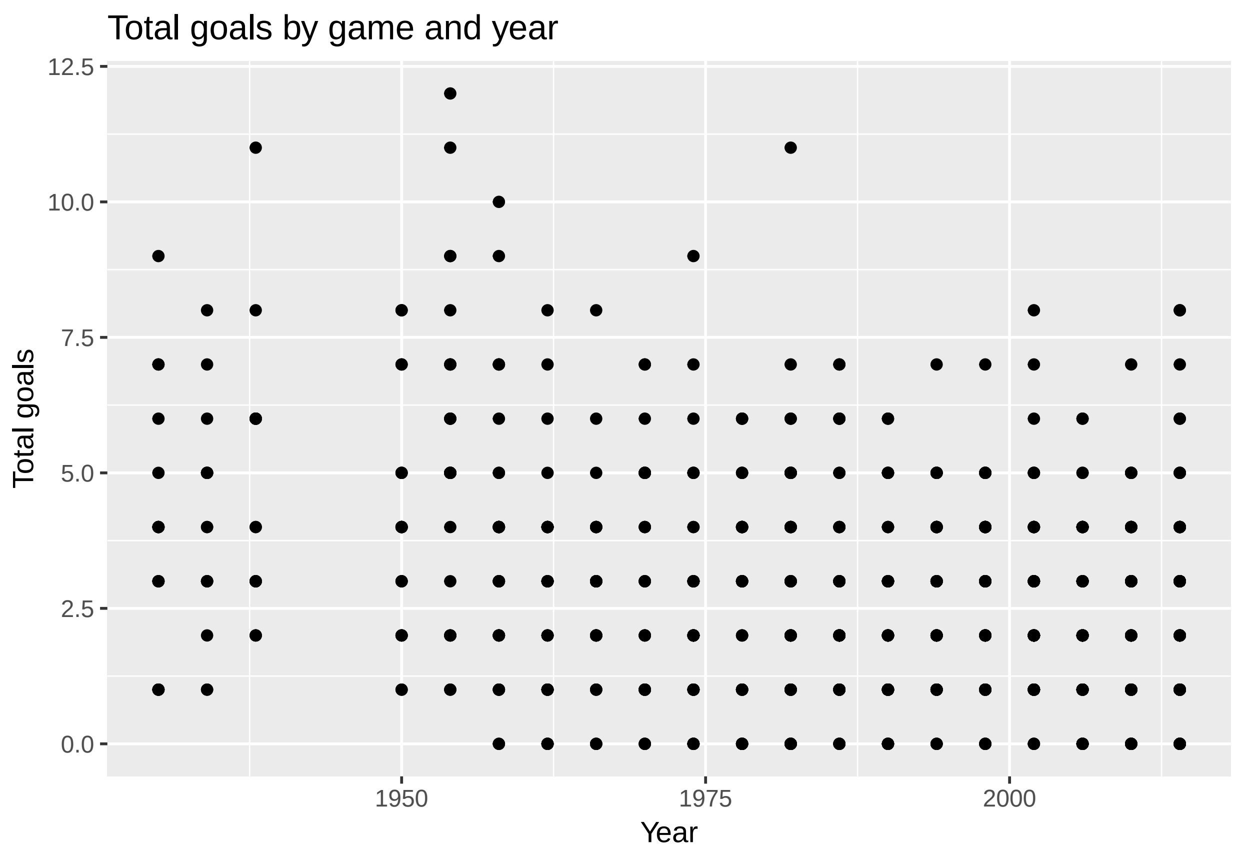
Are these variable types sensible? How might we interpret them in terms of the numerical/categorical distinction from the last lecture? Does it make sense to refer to **year** as a numerical (discrete) variable? What about **home_team_goals**?

Scatterplots and Data Summaries

When exploring data, creating plots is often one of the first things we want to do. Visual inspection of data can reveal structure of the data, help us explore relationships between variables, and summarize information in a compact way.

```
wc_games <- wc %>%
  mutate(total = home_team_goals + away_team_goals) %>%
  select(year, total)

ggplot(data = wc_games, aes(x = year, y = total)) +
    geom_point() +
    labs(
      x = "Year", y = "Total goals",
      title = "Total goals by game and year"
    )
```



This is an interesting plot! We've put a dot for every game in every year. Obviously some of the dots are overlapping (there aren't only 7 or 8 games per year), but we see quite a regular pattern. What happened in the 1940s? Also, what do we notice about the more recent data?

How else might we want to summarize this data? We might just want to summarize the average number of goals per game (both sides) across all time.

```r
mean(wc_games$total)
```

```
## [1] 2.830986
```

So over the entire data set, there's an average of just under 3 goals per game. As we discussed in class this week, it makes no sense to average the years: what would that even show?

If there's an average of just under 3 goals per game over time, why might the total number of goals during each World Cup be going up so obviously?

Vectors and Probability

When working in R, we can take the **mean()** of any vector of numbers available to us. Next week in workshop we'll have you create vectors, take means of them, and generally explore how we work with numbers. The **vector** is the first big thing that R provides that makes it superior to just hand calculation: they have no realistic maximum size! So we can have thousands or tens of thousands or millions of numbers, and just easily compute **statistics** on those numbers.

We'd like to make up some numbers to work with. Sometimes it's easier to just make up data than to try to find a data set to work with. Next week we're going to start talking about **probability**, so consider today a teaser for that.

You've all likely played with a die (six-sided) before. Maybe you played Yahtzee as a child (I loved Yahtzee!). Or Monopoly. Or any other board game which uses dice. Imagine we rolled a die 1000 times, and wrote down all of the numbers. It would be tedious, but doable. But tedious is what computers **love** to do! Let's "roll a die" 1000 times using the computer instead.

```r
dice_rolls <- sample(x = c(1, 2, 3, 4, 5, 6), size = 1000, replace = TRUE)
class(dice_rolls)
```

```
## [1] "numeric"
```

```r
head(dice_rolls)
```

```
## [1] 3 3 6 1 5 1
```

So now we have 1000 numbers, each of which is between 1 and 6, representing results from rolling a single die over and over. We can ask questions about these rolls: how many 1s did we get? What is the average result from the rolls? What do the rolls look like?

```r
mean(dice_rolls)
```

```
## [1] 3.532
```

```r
var(dice_rolls)
```

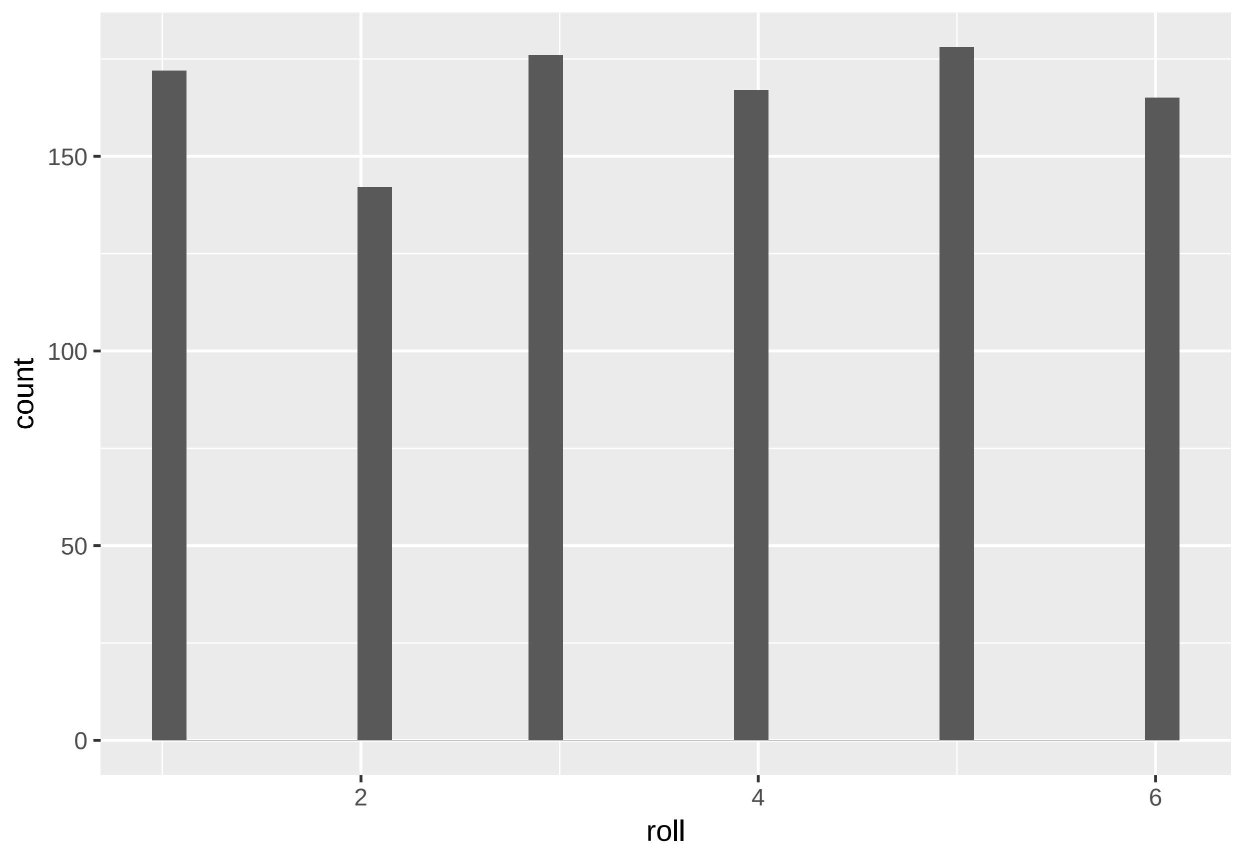
```
## [1] 2.91389
```

Next lecture we'll talk about the **mean** and **variance**: they are summaries of data which we use in statistics to easily communicate what the data looks like.

Histogram

Now, let's take a look at this data we just made **visually**. The **histogram** is a visual representation of data density, where higher bars represent where the data are relatively common. We can compute and plot this in R.

```r
dice_df <- data.frame(index = 1:1000, roll = dice_rolls)
ggplot(data = dice_df, aes(roll)) + geom_histogram()
```



Not terribly interesting, perhaps, but what does it show? The rolls are all basically at the same level: a little bit of variation up and down, but nowhere near as bad as humans might be! (Remember the example: if I ask you to pick a random number between 1 and 20, almost no-one picks 1, or 2, or 19, or 20. We're bad at randomness!)

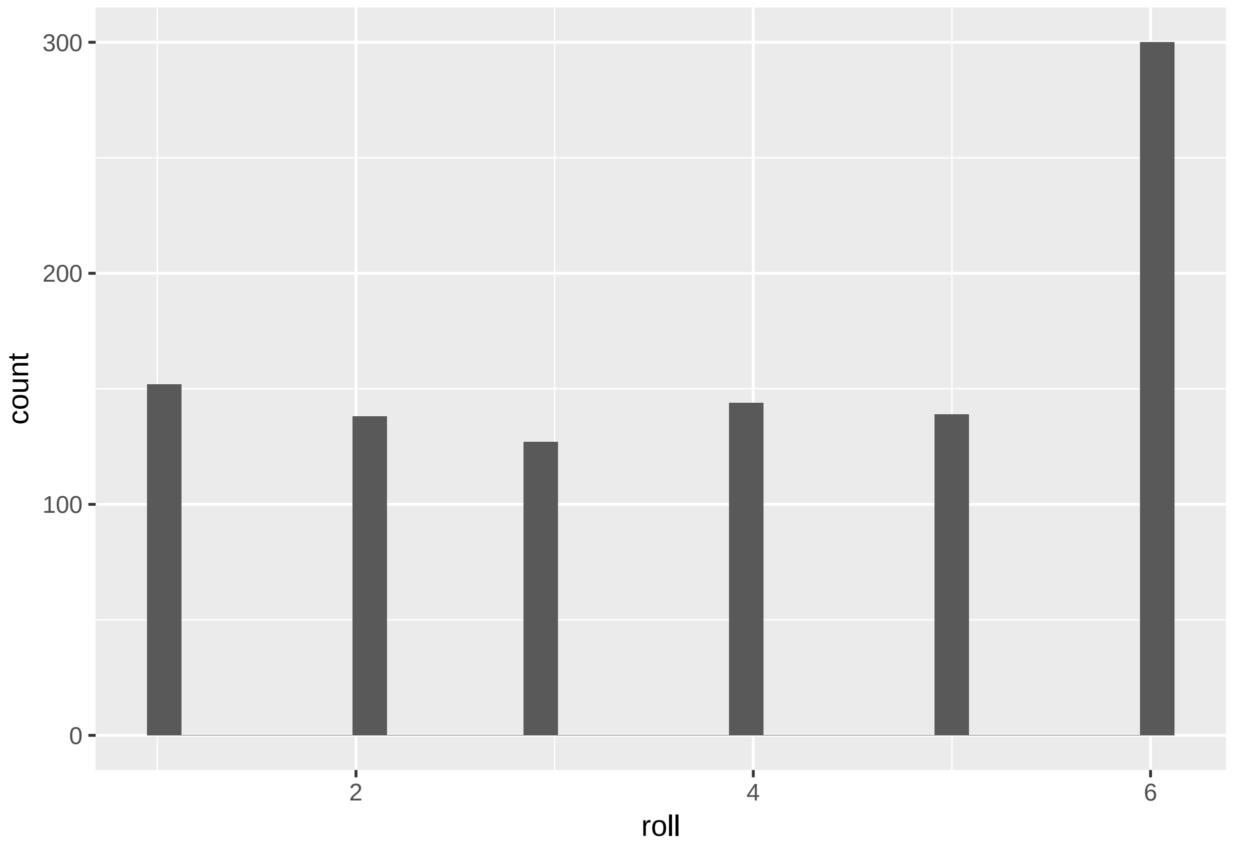
Bias and Histograms

When we "rolled a die" above, there was an argument we didn't set which defaulted to a sensible value. Let's rerwrite that line:

```r
dice_rolls <- sample(x = c(1, 2, 3, 4, 5, 6), size = 1000, replace = TRUE,
                     prob = c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6))
```

The argument **prob** sets the **probability** of getting particular elements from **x**. In this case, we've said "the probability of getting 1 is the same as 2, . . . , 6" - in other words, this is a "fair die". What if we suspect our friend is cheating at Monopoly by using a special die which lands on 6s more often than it should. What might data from repeated rolls of a die like **that** look like?

```r
bad_dice_rolls <- sample(x = c(1, 2, 3, 4, 5, 6), size = 1000, replace = TRUE,
                         prob = c(1/7, 1/7, 1/7, 1/7, 1/7, 2/7))
bad_dice_df <- data.frame(index = 1:1000, roll = bad_dice_rolls)
ggplot(data = bad_dice_df, aes(roll)) + geom_histogram()
```



What does this show? Wow, there's a lot of 6s! It looks like 300 of the 1000 rolls were 6s. Our friend is definitely cheating if this was their die.

Challenge

To see if you understand how this works, take the above code, copy and paste it, and see if you can setup a little "simulation" (that's what these are) to simulate flipping a normal two-sided coin 500 times. Try it with an "unbiased coin" (equal probability of heads and tails) and a "biased coin" (make heads have probability 0.6). Plot both of them.