Distributions and Central Tendency

EDP 613

Week 3

Prepping a New R Script

1. Open up a blank R script using the menu path **File > New File > R Script**.

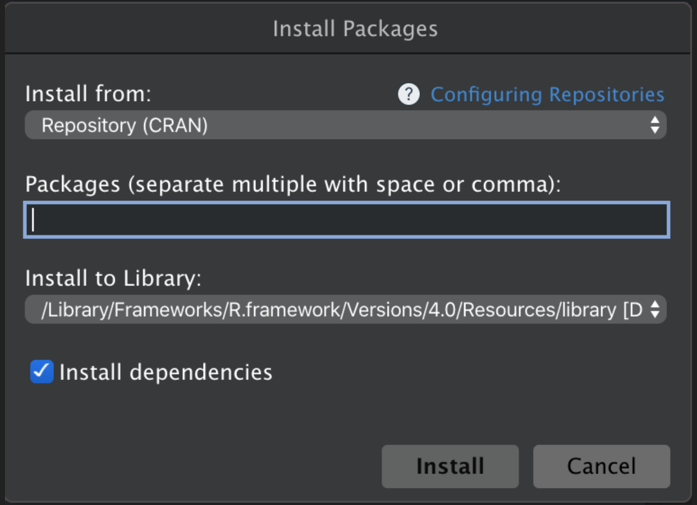
2. Save this script as `whatever.R` (replacing the term `whatever`) in your R folder. Remember to note where the file is!

3. After you have saved this file as `whatever.R`, go to the menu and select **Session > Set Working Directory > To Source File Location**.

Getting ready for this session

- Get the file `2012 Voter Fraud.csv` and save it in the same location as this script. Similarly make sure you also have `2012_Voter_Fraud_Codebook.pdf`. This document provides information about every variable in this data set. If you pursue any data driven research, a codebook like this is essential for data sets!

- Install the packages `car` and `descr`. Remember you can download it using **Tools > Install Packages** and typing in the name. Please make sure the **Install Dependencies** option has a checkmark beside of it. The install may take a minute.


Install Packages

Install from: Repository (CRAN)

Packages (separate multiple with space or comma):

Install to Library: /Library/Frameworks/R.framework/Versions/4.0/Resources/library [D

[✓] Install dependencies

- Load up the packages along with `tidyverse`.

```r
library(tidyverse)
library(car)
library(descr)
```

Frequency distributions

Often the first thing we do when we encounter a new variable is to look at the variable's frequency distribution which in a nutshell tells us how many observations there are at different values of a variable. If we did not have a computer, we could look at data sheet and count them up by hand (depends on how much you LOVE counting data points).

But by using R (or any other data analysis package), we look at frequency distributions by using a frequency table. This table provides a summary of a variable's values in to a condensed and simplified presentation. We can use frequency tables to look at any variable from any data set.

Loading up local data

To explore this, let's load the 2012 voter fraud file first and assign it to a variable. We can do this using the `read_csv()` command from the `readr` package within `tidyverse`.

```r
voter_fraud <- read_csv("2012_Voter_Fraud.csv")
```

```
## Rows: 50 Columns: 18
## ── Column specification ────────────────────────────────────────
## Delimiter: ","
## chr  (2): state, newspaper
## dbl (16): total, preelection, postelection, circulation, bat...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Side note

R itself uses `read.csv` which can be a royal pain if you don't know what you're doing. Its strongly advised that you stick with the tidy way of loading data.

Remember:

> - `read_csv` with a `_` is tidy
> - `read.csv` with a `.` is messy

Getting an idea of the data

You can always view the data set by using `View(voter_fraud)` but in this instance, let's take a look at the first six lines of the data set. Remember we can do this by using

```
head(voter_fraud)
```

```
## # A tibble: 6 × 18
##   state      newspaper total preelection postelection circulation
##   <chr>      <chr>     <dbl>       <dbl>        <dbl>       <dbl>
## 1 Alabama    Birmingh…     6           4            2      103729
## 2 Alaska     Anchorag…    11           4            7       58468
## 3 Arizona    Tucson D…     1           1            0      102063
## 4 Arkansas   Arkansas…    24          17            7       39215
## 5 California LA Times     20          13            7      657467
## 6 Colorado   Denver P…    21          20            1      340949
## # … with 12 more variables: battleground <dbl>,
## #   restrlaws2012 <dbl>, pct_black <dbl>, pct_white <dbl>,
## #   pct_hispanic <dbl>, pct_presvote_2008 <dbl>,
## #   pct_massconserv <dbl>, vep08_turnout <dbl>, pct_alec <dbl>,
## #   cases <dbl>, allcases <dbl>, gop_unified_state <dbl>
```

Alternative

... or we can do

```r
voter_fraud[1:6,]
```

```
## # A tibble: 6 × 18
##   state      newspaper total preelection postelection circulation
##   <chr>      <chr>     <dbl>       <dbl>        <dbl>       <dbl>
## 1 Alabama    Birmingh…     6           4            2      103729
## 2 Alaska     Anchorag…    11           4            7       58468
## 3 Arizona    Tucson D…     1           1            0      102063
## 4 Arkansas   Arkansas…    24          17            7       39215
## 5 California LA Times     20          13            7      657467
## 6 Colorado   Denver P…    21          20            1      340949
## # … with 12 more variables: battleground <dbl>,
## #   restrlaws2012 <dbl>, pct_black <dbl>, pct_white <dbl>,
## #   pct_hispanic <dbl>, pct_presvote_2008 <dbl>,
## #   pct_massconserv <dbl>, vep08_turnout <dbl>, pct_alec <dbl>,
## #   cases <dbl>, allcases <dbl>, gop_unified_state <dbl>
```

Just the (column) names please

We can only take a look at the column names by doing

```r
names(voter_fraud)
```

```
##  [1] "state"             "newspaper"         "total"
##  [4] "preelection"       "postelection"      "circulation"
##  [7] "battleground"      "restrlaws2012"     "pct_black"
## [10] "pct_white"         "pct_hispanic"      "pct_presvote_2008"
## [13] "pct_massconserv"   "vep08_turnout"     "pct_alec"
## [16] "cases"             "allcases"          "gop_unified_state"
```

Let's look at a frequency table for the variable `battleground`, which is a dummy variable for whether a state was a battlegroun not during the 2012 US presidential election. Here

> - `1 = 'battleground state'` and
> - `0 = 'not battleground state'`

There a number of ways of displaying tabular information , but in this case we'll just use the built-in R function `table`

```
table(voter_fraud$battleground)
```

```
##
##  0  1
## 42  8
```

> - The first row of the output tells us the values (remember what `1` and `0` stand for - these are categorical variables even though they are represented numerically).
> - The second row gives is the number of observations for each value. So we have
>   - 42 states battleground states and
>   - 8 non battleground states.

Well that is great but what if we have a lot of values? For example, let's look at the variable `total` which gives us data for the num
articles by newspaper.

```
table(voter_fraud$total)
```

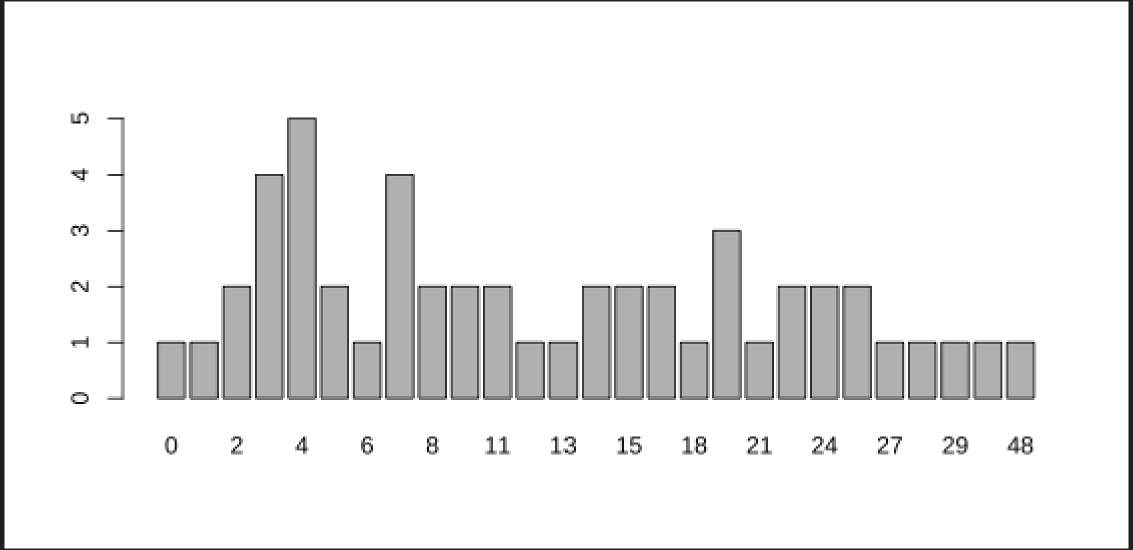
```
##
##  0  1  2  3  4  5  6  7  8  9 11 12 13 14 15 16 18 20 21 22 24
##  1  1  2  4  5  2  1  4  2  2  2  1  1  2  2  2  1  3  1  2  2
## 25 27 28 29 40 48
##  2  1  1  1  1  1
```

Using the `freq` command (1/2)

```
##
##   0   1   2   3   4   5   6   7   8   9  11  12  13  14  15  16  18  20  21  22  24
##   1   1   2   4   5   2   1   4   2   2   2   1   1   2   2   2   1   3   1   2   2
##  25  27  28  29  40  48
##   2   1   1   1   1   1
```

That seems like a lot. Let's plot them on a bar chart using the `freq` command from the `descr` package.

```r
freq(voter_fraud$total, plot = TRUE)
```



```
## voter_fraud$total
```

Using the `freq` command (2/2)

... and you get something like this

```
## voter_fraud$total
##        Frequency Percent
## 0              1       2
## 1              1       2
## 2              2       4
## 3              4       8
## 4              5      10
## 5              2       4
## 6              1       2
## 7              4       8
## 8              2       4
## 9              2       4
## 11             2       4
## 12             1       2
## 13             1       2
## 14             2       4
## 15             2       4
## 16             2       4
## 18             1       2
## 20             3       6
## 21             1       2
## 22             2       4
## 24             2       4
## 25             2       4
## 27             1       2
## 28             1       2
## 29             1       2
## 40             1       2
## 48             1       2
## Total         50     100
```

While you could present the results by going through each newspaper by stating 1 newspaper had zero articles, 1 newspaper had o article, 2 newspapers had two articles and so on.

This is time consuming and doesn’t really help in summarizing the information. Do we really need to display each individual newsp and their counts? Seems a bit messy. One way to make the data easier to understand is to use ranges.

We need to add a column to our data set and give it values for the ranges. We can simply add a column using the `$`. Because we w
add ranges, let's just call the new column by that term.

> `voter_fraud$ranges`

Then to add the actual ranges , we can use the `recode` command from the `descr` package.

```
voter_fraud$ranges <- recode(voter_fraud$total,
                                "0:9 = '0 - 9 articles';
                                10:19 ='10 - 19 articles';
                                20:29 = '20 - 29 articles';
                                30:39 = '30 - 39 articles';
                                40:49 = '40 - 49 articles'")
```

What do you think is happening here? Think about it for a minute before moving on.

The thing about R as opposed to the click and run nature of most softwares (e.g. SAS, SPSS, MiniTab, etc.) is that you know exactly what command is being run and what its doing with your data. Because of this, you are more likely to understand what is going on and less likely to go through the motions to get a solution.

Even if you can't figure it out...that's OK! Let's run the `table` command again. Maybe that will help:

```
table(voter_fraud$ranges)
```

```
##
##   0 - 9 articles 10 - 19 articles 20 - 29 articles
##               24               11               13
## 40 - 49 articles
##                2
```

That is much better! Now did you figure it out? Here let's look at the chunk again:

```
voter_fraud$ranges <- recode(voter_fraud$total,
                                "0:9 = '0 - 9 articles';
                                10:19 ='10 - 19 articles';
                                20:29 = '20 - 29 articles';
                                30:39 = '30 - 39 articles';
                                40:49 = '40 - 49 articles'")
```

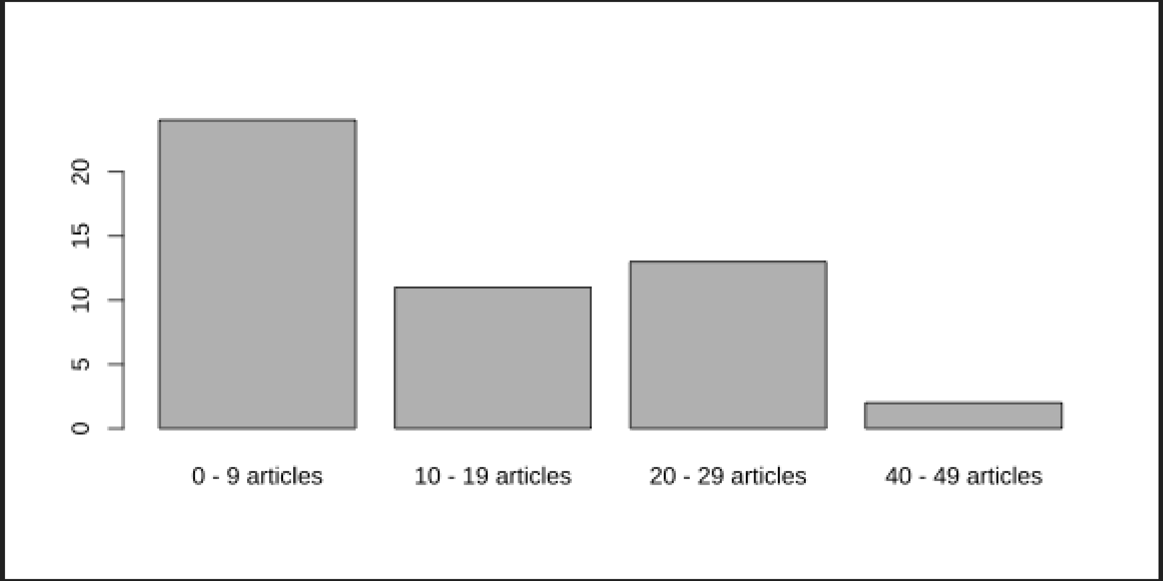
We actually did two things:

> - Inside the parenthesis: We made up our own ranges using all of the counts (e.g `0:9`) and told R to call them something else (e.g `0 - 9 articles`).
>
> - Outside the parenthesis: We used the `recode` command to tell R to count everything in each of those ranges and to give us the total.

Try playing with the range to see if you can group the articles more efficiently.

Now let's plot the bar chart again but in this case, we'll be using our new column `ranges`.

```
freq(voter_fraud$ranges, plot = TRUE)
```



```
## voter_fraud$ranges
##                   Frequency Percent
## 0 - 9 articles           24      48
## 10 - 19 articles         11      22
## 20 - 29 articles         13      26
## 40 - 49 articles          2       4
## Total                    50     100
```

That's much better!

There's actually an even easier way to do this totally in the `tidyverse` family but we'll get to that later in the term.

Right now your job is just to get familiar with how logic is used in R.

Measures of Central Tendency

To take a look at how we assess the mean, median, and mode, let's use our original data set and first look at the `total` column which has the raw data counts.

```r
head(voter_fraud$total)
```

```
## [1]  6 11  1 24 20 21
```

For the mean, we use

```r
mean(voter_fraud$total)
```

```
## [1] 13.3
```

For the median, we use

```
median(voter_fraud$total)
```

```
## [1] 11
```

For the mode, we use

```r
mode(voter_fraud$total)
```

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

`mode` still doesn't work!

For various programming reasons that are not worth discussing at this time, R does not have a standard in-built function to calculat
mode. Try running

```
?mode
```

The `?` symbol prior to any command will give you information about that function provided you have that package loaded. Yes its not clearly not written in plain English, but it does tell you that the `mode` command means something else in R.

OK let's get back to the mode we need. To get the mode, we actually have to tell R how to go about calculating it. Below is a function that will help. In order for R to recognize it, you have to run the following sequence before using the command.

```
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Notice that 'Mode' is capitalized so that R won't confuse it
# with its internal command 'mode'.
```

Keep that chunk in a file somewhere. You don't have to know what it means or how to construct it or any of that. This is not a programming class. As long as you run that chunk before trying to find a mode, you'll be fine.

```
Mode(voter_fraud$total)
```

```
## [1] 4
```

On Your Own

This is your chance to get some practice in and to ask questions. You won't get the opportunity to get help during quizzes and exams so take advantage now!

Open up a new script and load up the `Box Office.csv` data set in R. This set was scraped from Rotten Tomatoes prior to Avengers: Endgame becoming the highest grossing movie of all time.

Now try answering the following questions using R:

1. What is the average number of positive reviews for the top five movies?
2. What are the average number of negative reviews for the bottom five movies?
3. How were movies released over the years? Provide counts and a visualization.
4. Which measure of central tendency is the best to describe the average number of movies over the years?
5. Which year has the most number of ranked movies?

I'll post the solutions next week!

That's it for today!