Hacking for Justice - Introduction to R

Alex C. Engler - The University of Chicago

Saturday, September 22nd

Open RStudio

Since you installed R and RStudio during class, you should simply need to open RStudio (not R) in order to get started. RStudio is a tool to make working in R (the programming language) a bit easier and more intuitive.

Download the SAO Data

The State's Attoneys Office (SAO) has four datasets at the case level - which means each row of data describes one court case. You can find those datasets under this search on the Cook County Data Catalog. For your convenience, direct links to and descriptions of the datasets are provided here:

- Sentencing: The sentencing data presented in this report reflects the judgment imposed by the court on people that have been found guilty. Each row represents a charge that has been sentenced.
- Dispositions: The disposition data presented in this data reflects the culmination of the fact-finding process that leads to the resolution of a case. Each row represents a charge that has been disposed of.
- Initiation: The Initiation results data presented here reflects all of the arrests that came through the door of the State's Attorneys Office (SAO). An initiation is how an arrest turns into a "case" in the courts. Most cases are initiated through a process known as felony review, in which SAO attorneys make a decision whether or not to prosecute. Cases may also be indicted by a grand jury or, in narcotics cases, filed directly by law enforcement (labeled "BOND SET (Narcotics)" in this data). Included in this data set are the defendant counts by initiation and year. This data includes felony cases handled by the Criminal, Narcotics, and Special Prosecution Bureaus. It does not include information about cases processed through the Juvenile Justice and Civil Actions Bureaus.
- Intake: The intake data presented in this data reflects the cases brought in for review. Each row represents a potential defendant in a case.

Loading Data Into R:

For our introduction, we will use the tidyverse, which is a set of R packages that enable quick and (somewhat) intuitive ways to explore and manipulate date in R. If you have already install the tidyverse using `install.packages("tidyverse")`, then you should just need to run the code below.

```
library(tidyverse)
```

Next, load the sentencing data into R. The `read_csv()` function loads the data into R, and the assignment operator `<-` saves the data under the name `sentence`, which we will use to refer to it from here forward. Data loaded into R is called a `dataframe`, and we will use that terminology going forward.

```
# Note: You can write comments in your R code following a hashtag `#`.
# Anything after a hashtag will not run, so you can use comments to write
# notes to yourself, explaining what your code does (or should do!).
sentence <- read_csv("Sentencing.csv")
```

We can use simple functions, like `nrow()` and `ncol` to see how many rows and columns of data there are in this dataframe. Note, you must use all lowercase letters for `sentence` and the function names, as R is case sensitive.

```
nrow(sentence)
```

```
## [1] 189287
```

```
ncol(sentence)
```

```
## [1] 36
```

We can visually inspect our loaded dataframe with the `glimpse()` function. This prints each column, or variable, in the dataset on each row. It also tells us what type of variable is contained in the column (e.g. a `dbl` means a continuous number, whereas a `chr` means characters, which could be letters, numbers or a sentence). Finally, this function shows us some of the first values in each column, listed from left to right.

```
glimpse(sentence)
```

```
## Observations: 189,287
## Variables: 36
## $ CASE_ID                    <dbl> 26783584167, 26651437018, 2676852092...
## $ CASE_PARTICIPANT_ID        <dbl> 46480038575, 46118600972, 4643960910...
## $ CHARGE_ID                  <dbl> 33613165290, 33297012068, 3361328199...
## $ CHARGE_VERSION_ID          <dbl> 200413732973, 200414312333, 20041445...
## $ PRIMARY_CHARGE             <chr> "true", "true", "true", "false", "tr...
## $ OFFENSE_TITLE              <chr> "UNLAWFUL USE OR POSSESSION OF A WEA...
## $ CHAPTER                    <chr> "720", "720", "720", "720", "720", "...
## $ ACT                        <int> 5, 570, 5, 550, 570, 5, 5, 5, 570, 5...
## $ SECTION                    <chr> "24-1.1(a)", "402(c)", "19-1(a)", "4...
## $ CLASS                      <chr> "2", "4", "2", "4", "4", "2", "2", "...
## $ AOIC                       <chr> "0012309", "5101110", "1110000", "50...
## $ DISPO_DATE                 <chr> "10/11/2012 12:00:00 AM", "07/29/201...
## $ SENTENCE_PHASE             <chr> "Original Sentencing", "Original Sen...
## $ SENTENCE_DATE              <chr> "10/03/2012 12:00:00 AM", "07/28/201...
## $ SENTENCE_JUDGE             <chr> "Stanley  Sacks", "Thaddeus L Wilson...
## $ SENTENCE_TYPE              <chr> "Prison", "Probation", "Prison", "Pr...
## $ COMMITMENT_TYPE            <chr> "Illinois Department of Corrections"...
## $ COMMITMENT_TERM            <dbl> 6, 2, 6, 2, 2, 7, 3, 16, 3, 2, 2, 1,...
## $ COMMITMENT_UNIT            <chr> "Year(s)", "Year(s)", "Year(s)", "Ye...
## $ CHARGE_DISPOSITION         <chr> "Plea Of Guilty", "Plea Of Guilty", ...
## $ CHARGE_DISPOSITION_REASON <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ COURT_NAME                 <chr> "District 1 - Chicago", "District 1 ...
## $ COURT_FACILITY             <chr> "26TH Street", "26TH Street", "26TH ...
## $ LENGTH_OF_CASE_in_Days     <int> 414, 86, 339, 86, 44, 31, NA, 709, 1...
## $ AGE_AT_INCIDENT            <int> 29, 45, 50, 45, 41, 25, 30, 19, 57, ...
## $ GENDER                     <chr> "Male", "Male", "Male", "Male", "Mal...
## $ RACE                       <chr> "Black", "HISPANIC", "Black", "HISPA...
## $ OFFENSE_TYPE               <chr> "UUW - Unlawful Use of Weapon", "Nar...
## $ INCIDENT_BEGIN_DATE        <chr> "07/06/2011 12:00:00 AM", "03/26/201...
## $ INCIDENT_END_DATE          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ ARREST_DATE                <chr> "07/06/2011 11:35:00 PM", "03/26/201...
## $ LAW_ENFORCEMENT_AGENCY     <chr> "CHICAGO PD", "CHICAGO PD", "CHICAGO...
## $ UNIT                       <chr> "District 10 - Ogden", "District 8 -...
## $ INCIDENT_CITY              <chr> "Chicago", "Chicago", "Chicago", "Ch...
## $ RECEIVED_DATE              <chr> "07/07/2011 12:00:00 AM", "03/29/201...
## $ ARRAIGNMENT_DATE           <chr> "08/16/2011 12:00:00 AM", "05/03/201...
```

Some of the data is self-explanatory, such as the `SENTENCE_TYPE` column, which contains the type of the sentence that resulted in this judgement. Below, we use the `table()` function to create a frequency table - this tells us every value of the `SENTENCE_TYPE` column and how many times that value appears in the dataset. We use the `$` operator to refer to the `SENTENCE_TYPE` column within the `sentence` dataframe.

```
table(sentence$SENTENCE_TYPE)
```

```
##
##                  2nd Chance Probation
##                                  1080
##                 Conditional Discharge
##                                  2696
##                   Conditional Release
##                                    70
##                             Conversion
##                                     5
##                 Cook County Boot Camp
##                                  1806
##                                 Death
##                                    59
##      Inpatient Mental Health Services
##                                   137
##                                  Jail
##                                  5563
##                                Prison
##                                106093
##                             Probation
##                                 69499
##        Probation Terminated Instanter
##                                    74
##   Probation Terminated Satisfactorily
##                                    43
## Probation Terminated Unsatisfactorily
##                                   569
##                            Supervision
##                                  1593
```

```
# or count(sentences, SENTENCE_TYPE)
```

Remember to Use the Data Documentation

However, other columns may not be so easily interpreted, and guessing can lead to mistakes. For instance, the `PRIMARY_CHARGE` column has values of "true" and "false", which is not clearly self-explanatory.

It's important to consistently use the data documentation, sometimes called a codebook, to help learn about a dataset. Scrolling down on the same page we found this data, we can see there are short descriptions of what each coulmn contains.

```
table(sentence$PRIMARY_CHARGE)
```

```
##
##  false   true
##  50411 138876
```

```
# or count(sentences, PRIMARY_CHARGE)
```

Level Of The Data

Reading the documentation has revealed something important about the data (as it often does!). It is easy to open this dataset and assume that each row of data was the sentencing for a distinct and unique case. However, this is not correct! We can see there are many cases that appear in the data more than once.

Below, we first create a list of all distinct values of the `CASE_ID` column using the `unique()` function. Then, in the same line of code, we count how many there are using `length()`.

```
length(unique(sentence$CASE_ID))
```

```
## [1] 155443
```

This results in 155443 unique values, far fewer than the 189287 rows of data.

Breaking Apart Confusing Code

If the code above was tough for you to follow, try break it apart into its component pieces. For instance, what happens if you just run the following line:

```
unique(c("cat", "dog", "fish, "cat))
```

Does this help better illustrate how `unique()` is working? Now trying running `length()` and `unique()` together, like below.

```
length(unique(c("cat", "dog", "fish, "cat)))
```

Interpreting the Level Of the Data

In this data, one row is actually defined by the `CHARGE_ID` variable. This is to say that each row of data, or observation, is one unique charge resulting in sentencing, with potentially several or many charges per case.

This is important, since if we were to simply look at the average age across this dataset, we might substantially misinterpret the resulting number. Instead, lets select a group of columns that will be consistent across each case. Below, we use the `select()` function to grab only a few of the columns.

Columnar Selection

```
cases <- select(sentence, CASE_ID, CASE_PARTICIPANT_ID, AGE_AT_INCIDENT,
                GENDER, RACE, LENGTH_OF_CASE_in_Days, INCIDENT_CITY)
ncol(cases)
```

```
## [1] 7
```

Now, we can grab only the rows that are unique across these values, which should be one participant per case. From that dataframe, we can use various functions for descriptive statistics, like `mean()`, `median()`, and `fivenum()`. We use the addition argument `na.rm = TRUE` to tell R to ignore missing values in these calculaions.

```
cases <- distinct(cases)
dim(cases)
```

```
## [1] 167397      7
```

```
mean(cases$AGE_AT_INCIDENT, na.rm=TRUE)
```

```
## [1] 32.10246
```

```r
median(cases$AGE_AT_INCIDENT, na.rm=TRUE)
```

```
## [1] 29
```

So, for any case, we can expect a participant to be aged 29, with an average age of a participant in a case being 29.

The `fivenum()` function returns the minimum, 25th percentile, median, 75th percentile, and maximum.

```r
fivenum(cases$AGE_AT_INCIDENT, na.rm=TRUE)
```

```
## [1] 17 22 29 41 85
```

Take a minute to break apart the second line of code below. Can you intuit what the numbers being returned mean?

```r
prop.table(table(cases$INCIDENT_CITY == "Chicago", useNA="always"))
```

```
##
##      FALSE       TRUE       <NA>
## 0.28211975 0.64797458 0.06990567
```

Filtering The Data

We can use the `filter()` function to look at only some rows of the data. Below, I have created several smaller datasets from the original. Each new dataset, on the left of the assignment operator `<-` is composed of the rows from the original dataset that meet the criteria specified in the `filter() function`.

```
sentence_female <- filter(sentence, GENDER == "Female") ## == means exactly equal to
sentence_under21 <- filter(sentence, AGE_AT_INCIDENT <= 21) ## <= means less than or equal to
sentence_probation <- filter(sentence, SENTENCE_TYPE %in% c("Probation", "2nd Chance Probation"))
## %in% means 'is one of'
```

How can you be sure that these filters worked as you expected? Use the `table()` function and the `hist()` function on the newly created datasets (sentence_female, sentence_under21, and sentence_probation) to ensure you understand what the filters accomplished.

> If you were able to confirm what was happening above, try this on your own. Write a filter that only looks at cases longer than one year (using `LENGTH_OF_CASE_in_Days`) and/or sentencing imposed on Hispanic persons.

New Column Creation

We can use the mutate function to create a new column of data. Below, I create a column called `HISPANIC` that is `TRUE` for observations with a hispanic participant.

```
sentence <- mutate(sentence, HISPANIC = ifelse(RACE %in% c("HISPANIC", "White [Hispanic or Latino]", "W
table(sentence$RACE, sentence$HISPANIC)
```

```
##
##                                    FALSE   TRUE
##   American Indian                     97      0
##   Asian                             1085      0
##   ASIAN                               63      0
##   Biracial                            31      0
##   Black                           125047      0
##   HISPANIC                             0   5332
##   Unknown                            257      0
##   White                            27326      0
##   White [Hispanic or Latino]           0  28102
##   White/Black [Hispanic or Latino]     0    892
```

Remember you can give the `ifelse()` function a try on its own if you want to see what it might do. Try playing around with different versions of: `ifelse(5 > 2, "Yes!", "No!")`

> Can you create a new variable for sentences resulting in over a two year commitment to prison? You will need to look at the `SENTENCE_TYPE`, `COMMITMENT_TERM`, and `COMMITMENT_UNIT` variables to do so.

TO DO NEXT:

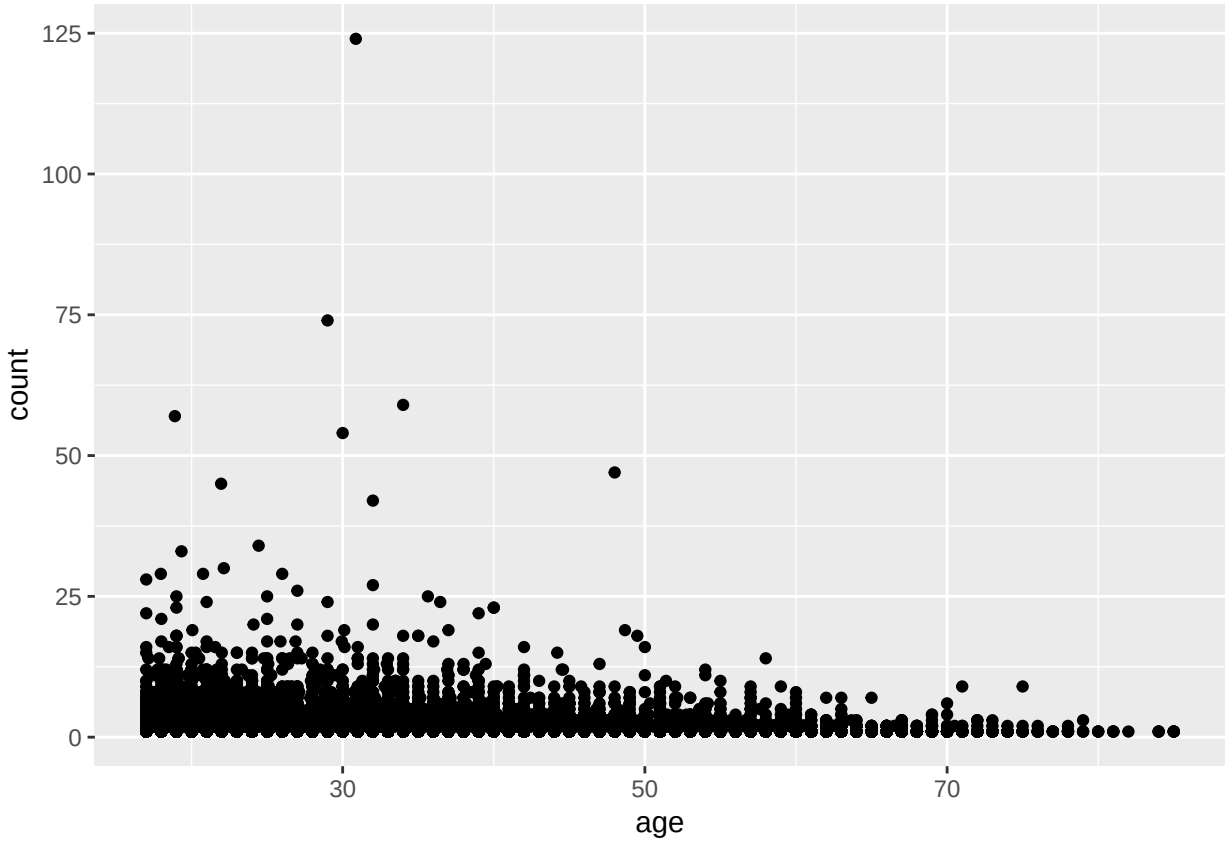
Probably the next parts of this to include are:

- Arrange (easy)
- Histogram? Basic Plots? (Only if no one else is)
- Grouped Aggregation
- Merging on a different SAO dataset (maybe too much?) Guidance on which dataset would be good. My instinct is initiation and hope CASE_ID or CHARGE_ID works there.

We can see more charges per person at younger ages, so our original histogram was skewed to the left. `echo=FALSE` is on here.

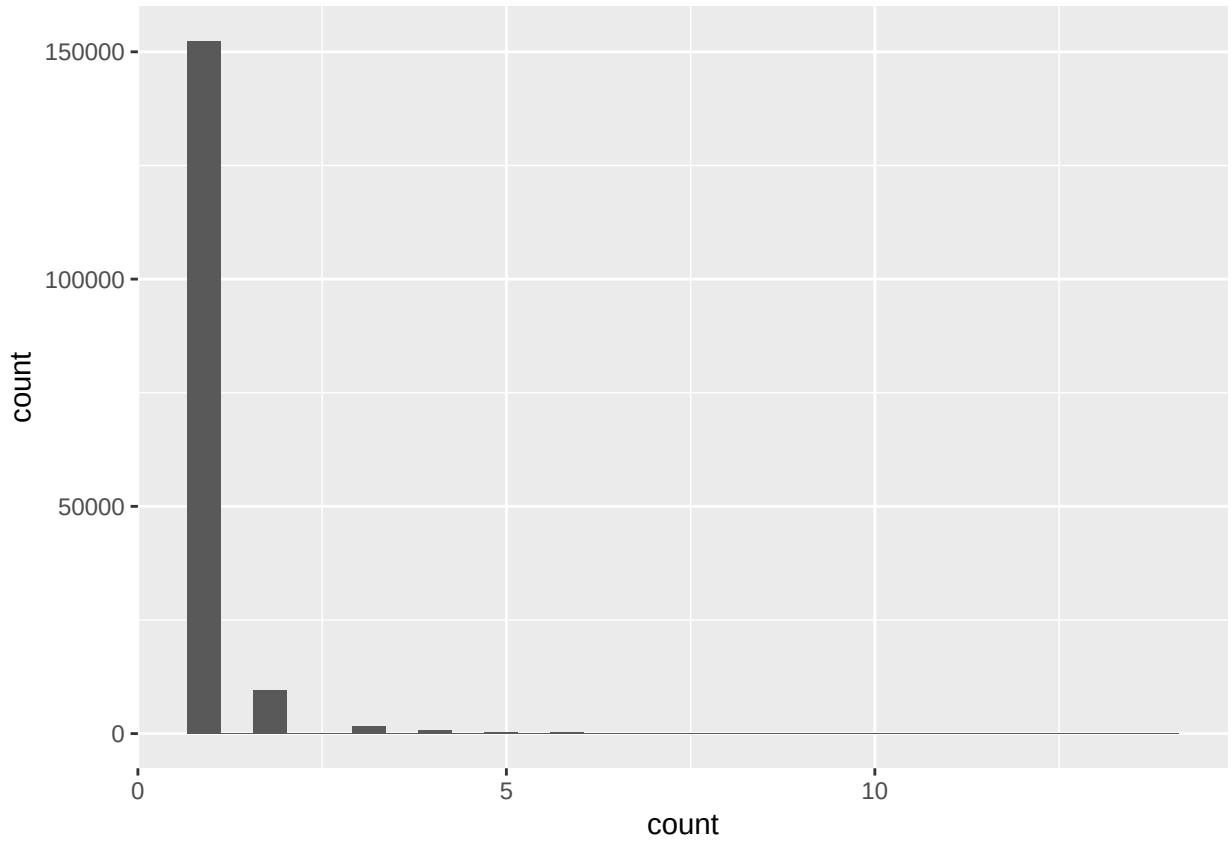
```
sentence %>% group_by(CASE_ID) %>%
  summarize(age = mean(AGE_AT_INCIDENT), count=n()) %>%
  ggplot(aes(age, count)) + geom_point()
```

```
## Warning: Removed 2097 rows containing missing values (geom_point).
```

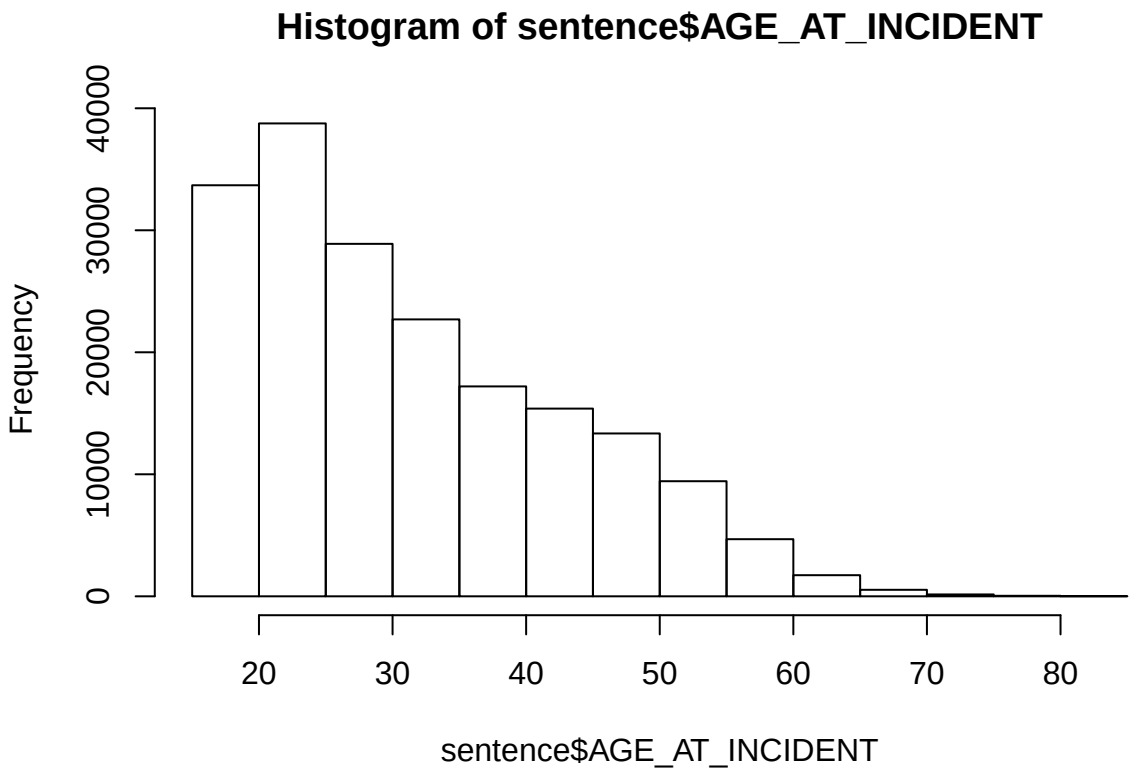



```
sentence %>% group_by(CASE_ID, AGE_AT_INCIDENT) %>%
  summarize(count=n()) %>%
  filter(count < 15) %>%
  ggplot(aes(count)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
hist(sentence$AGE_AT_INCIDENT)
```



```
# Or
# ggplot(sentence, aes(AGE_AT_INCIDENT)) + geom_histogram()
```

Merging On Another Dataset

Let's load in a different dataset from the SAO:

```
initiation <- read_csv("Initiation.csv")
```

Always make sure to carefully examine new datasets. This should at least including using functions like `glimpse()` or `View()` to look at the data, as well as explorations we have used today like `table()`, `hist()`, and `unique()`.

Appendix 1: R Terminology

- Comments: Everything after a `#` (a hashtag) in your code will have no effect if you run it in R. Thus, you can use `#` hashtags to write notes to yourself and others, making your code more readable.
- Working Directory : The folder on your computer that R is currently working in. It will only check this folder for files to load, and will write any new files to this folder.
- Dataframe : the R equivalent of an excel file. It holds relational data in rows and columns that can contain numbers or strings.
- Assignment Operator `<-` : Gives the value on the right to the object on the left.
- Function : Anything that completes a task or set of tasks in R is a function. Most functions have a name, and take one or more arguments within parentheses. Examples include 'head()', 'colnames()', 'hist()', 'mean()', 'and plot()'

- Argument : An input or an option that affects the result of a function. This often includes the data that the function runs on, AND specifications/options as to what the function should do. For example:

```
hist(dataframe$column, main = "A Histogram")
```

The function above (hist is a function for making a histogram) above is given two arguments, separated by a comma. The first is 'data$column', telling the histogram to use the data in this column to make a histogram. The second arguments is 'main = "A Histogram" ', which is activating an option, and giving the histogram a main title.

Mathematical Operators in R:

```
2+2 # Addition with the plus sign `+`
```

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

```
6-3 # Subtraction with the - sign
```

```
## [1] 3
```

```
4*2 # The asterisk (*) indicates multiplication
```

```
## [1] 8
```

```
12/3 # Division usses the backslash
```

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

```
3^3 ## This caret `^` means exponentiation, so this is 3 to the third power.
```

```
## [1] 27
```

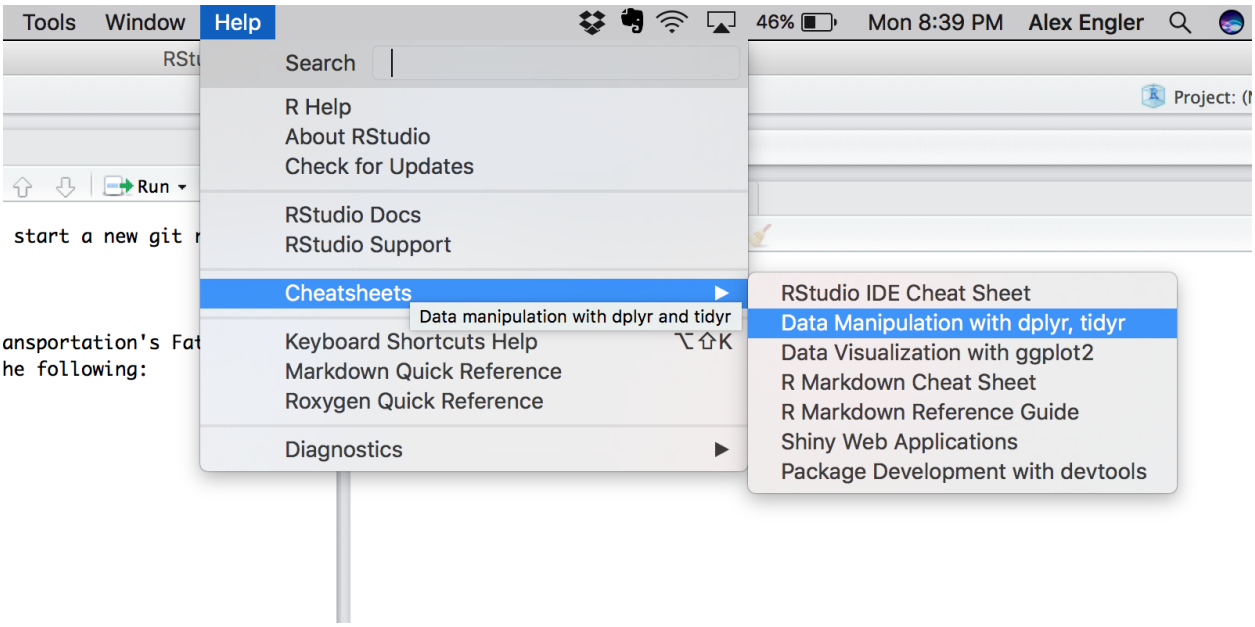


Figure 1: Cheat Sheets

Appendix 2: Pertinent Resources

Introduction to haven package Link

Introduction to readr package Link

Introduction to readxl package Link

Vignette on dplyr package for Data Manipulation Link

Data Processing with dplyr & tidyr Link

String Manipulation with stringr Link

In the image above [Figure 1], you can see how to navigate to the RStudio Cheat Sheets for R's very useful data manipulation packages, `dplyr` and `tidyr`. These packages, as well as `stringr`, are also covered in detail in the excellent free ebook R for Data Science.