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:

First, we need to 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 `head()` function. This print the first six rows of the dataset, with columns print left to right and wrapping down when the R console runs out of space. Since this would take a lot of room in this document, the output is not printed below.

```
head(sentence)
```

By default, `head()` prints out six row of data, but note that you can affect that by changing the `n` argument, as seen below. `head()` is a function (code that does something), and `n` is what is called an 'argument', which you can think of as an option that alters how a function works.

```
head(sentence, n=10)
```

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
```

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
```

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, use indexing (this is the square brackets `[]`) to select only some of the variables and put them in a new dataframe.

Indexing a Dataframe

We can use the square brackets to only select some small piece of our entire dataframe. We specify which rows and columns we want in the order of `[rows, columns]`. Here are some quick examples.

```
## Get the first five rows of data:
sentence[1:5,]

## Get the first ten rows of the second column of the data:
```

```
sentence[1:10,2]

## Get the first five rows of the column named "ARREST_DATE" of the data:
sentence[1:5,"ARREST_DATE"]

## Run the line below - can you understand why it returns this result?
sentence[100:115,c("LAW_ENFORCEMENT_AGENCY","ARREST_DATE")]
```

Filtering The Data

We can use our new understanding of indexing and the `which()` function to filter the dataset based on the values in the columns, which is far more useful than doing this based on row numbers.

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 square brackets

```
sentence_female <- sentence[which(sentence$GENDER == "Female"),] ## == means exactly equal to
sentence_under21 <- sentence[which(sentence$AGE_AT_INCIDENT <= 21),] ## <= means less than or equal to
sentence_probation <- sentence[which(sentence$SENTENCE_TYPE  %in% c("Probation", "2nd Chance Probation"),
```

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.

Columnar Selection

```
cases <- sentence[,c("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 calculations.

```
cases <- cases[!duplicated(cases),]
dim(cases)
```

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

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

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

```
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.

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

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

```
##
##     FALSE      TRUE      <NA>
## 0.3520254 0.6479746 0.0000000
```

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]", "
# table(sentence$RACE, sentence$HISPANIC)
```

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.

Handling Missing Data

Many of the values for the `LENGTH_OF_CASE_in_Days` are already missing, as we can see below. The `is.na()` function is checking if each individual row of data contains an `NA` value (R's value for missing data).

```
table(is.na(sentence$LENGTH_OF_CASE_in_Days))
```

```
##
##  FALSE   TRUE
## 181244   8043
```

So there are `8043` missing values for this column. However, there appear to be additional rows with infeasible values, like having negative case lengths.

```
table(sentence$LENGTH_OF_CASE_in_Days < 0)
```

```
##
##  FALSE   TRUE
## 180906    338
```

I think we can reasonably assume that cases are not lasting negative time, so we should remove these values before running any analysis (like calculating the average length of case).

> Can you use mutate() to replace negative values of `LENGTH_OF_CASE_in_Days` with missing values (NA)? You might make a new column called `LENGTH_OF_CASE_ALT` to do this.

```
# sentence <- mutate(sentence, LENGTH_OF_CASE_ALT = ifelse(LENGTH_OF_CASE_in_Days < 0, NA, LENGTH_OF_CA
```

Run the line with the combination of `table()` and `is.na()` again, did the number of missing values increase? It should have - always make sure to check your work!

Alternatively, we can use hist to make a histogram (more on this in the next section). You should see an image like the one below if it ran correctly.

```
# hist(sentence$LENGTH_OF_CASE_ALT)
```

Merging On Another Dataset

Very frequently, the data you are more interested or curious in will not be made available in just one dataset. So acommon task is to combine two datasets from different sources. Let's load in a different dataset from the SAO and merge this onto our sentencing data.

```
initiation <- read.csv("Dispositions.csv")
```

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

Once you have done this, we can merge on the dispositions data with the `merge()` function.

Loading New Packages

For our introduction, used the normal functionality available in R. However, one of the great advantages of working in R is tha ability to load new packages (pre-defined functionality) that others have written. The `tidyverse` is a great example of this advantage of open source languages. The `tidyverse` which is a set of R packages that enable quick and (somewhat) intuitive ways to explore and manipulate date in R. To install and then load these packages, run the code below.

```
# install.packages("tidyverse")
library(tidyverse)
```

We now have new functions available that we could not use before, like `glimpse()`, for cleanly displaying our data.

```
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             <fct> true, true, true, false, true, true,...
## $ OFFENSE_TITLE              <fct> UNLAWFUL USE OR POSSESSION OF A WEAP...
## $ CHAPTER                    <fct> 720, 720, 720, 720, 720, 720, 720, 7...
## $ ACT                        <fct> 5, 570, 5, 550, 570, 5, 5, 5, 570, 5...
## $ SECTION                    <fct> 24-1.1(a), 402(c), 19-1(a), 4(d), 40...
## $ CLASS                      <fct> 2, 4, 2, 4, 4, 2, 2, X, 1, 1, 4, 4, ...
## $ AOIC                       <fct> 0012309, 5101110, 1110000, 5015400, ...
## $ DISPO_DATE                 <fct> 10/11/2012 12:00:00 AM, 07/29/2011 1...
## $ SENTENCE_PHASE             <fct> Original Sentencing, Original Senten...
## $ SENTENCE_DATE              <fct> 10/03/2012 12:00:00 AM, 07/28/2011 1...
## $ SENTENCE_JUDGE             <fct> Stanley  Sacks, Thaddeus L Wilson, L...
## $ SENTENCE_TYPE              <fct> Prison, Probation, Prison, Probation...
## $ COMMITMENT_TYPE            <fct> Illinois Department of Corrections, ...
## $ COMMITMENT_TERM            <dbl> 6, 2, 6, 2, 2, 7, 3, 16, 3, 2, 2, 1,...
## $ COMMITMENT_UNIT            <fct> Year(s), Year(s), Year(s), Year(s), ...
## $ CHARGE_DISPOSITION         <fct> Plea Of Guilty, Plea Of Guilty, Plea...
## $ CHARGE_DISPOSITION_REASON <fct> , , , , , , , , , , , , , , , , , , ...
## $ COURT_NAME                 <fct> District 1 - Chicago, District 1 - C...
## $ COURT_FACILITY             <fct> 26TH Street, 26TH Street, 26TH Stree...
## $ 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                     <fct> Male, Male, Male, Male, Male, Male, ...
## $ RACE                       <fct> Black, HISPANIC, Black, HISPANIC, Wh...
## $ OFFENSE_TYPE               <fct> UUW - Unlawful Use of Weapon, Narcot...
## $ INCIDENT_BEGIN_DATE        <fct> 07/06/2011 12:00:00 AM, 03/26/2011 1...
## $ INCIDENT_END_DATE          <fct> , , , , , , , , , , , , , , , , , , ...
## $ ARREST_DATE                <fct> 07/06/2011 11:35:00 PM, 03/26/2011 0...
## $ LAW_ENFORCEMENT_AGENCY     <fct> CHICAGO PD, CHICAGO PD, CHICAGO PD, ...
## $ UNIT                       <fct> District 10 - Ogden, District 8 - Ch...
## $ INCIDENT_CITY              <fct> Chicago, Chicago, Chicago, Chicago, ...
## $ RECEIVED_DATE              <fct> 07/07/2011 12:00:00 AM, 03/29/2011 1...
## $ ARRAIGNMENT_DATE           <fct> 08/16/2011 12:00:00 AM, 05/03/2011 1...
```

If you want to further your understanding of R for analyzing data, my first suggestion is to read (and practice)

R for Data Science, which teaches how to use the packages in the `tidyverse`.

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.
- Package : An R package is a collection of R code that adds new functionality and functions.
- 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
```

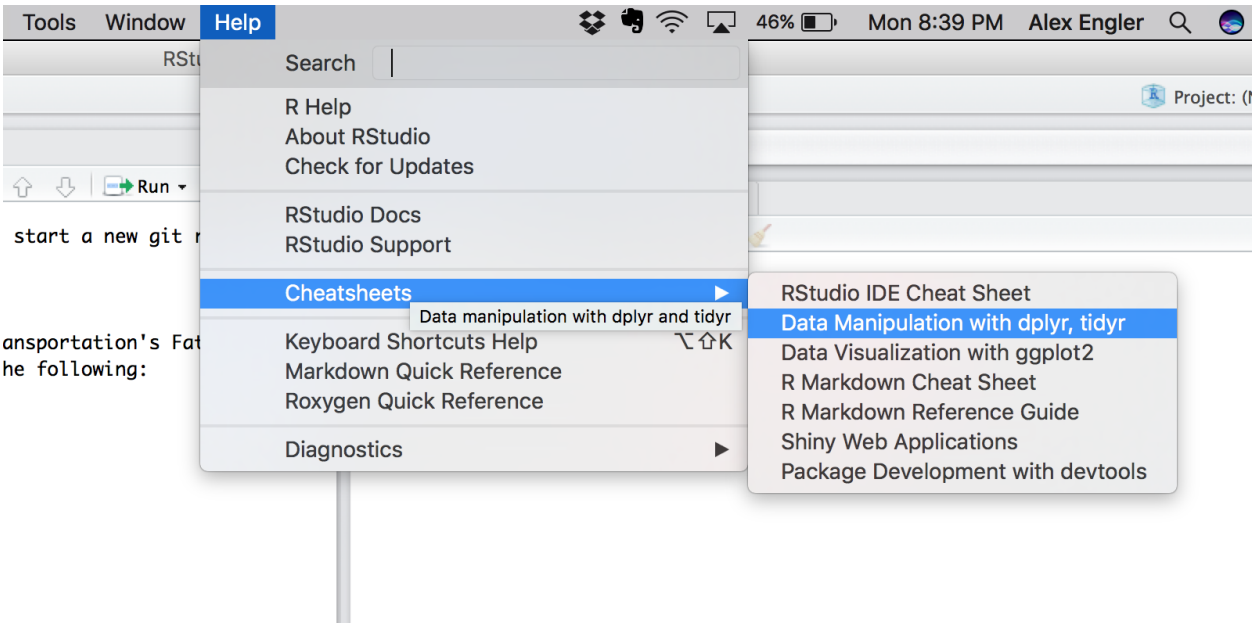

Figure 1: Cheat Sheets

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.