DataViz Assignment: R & Git

Alex C. Engler - The University of Chicago

Due: October 3rd, 2017

Install R & RStudio

Instructions are available on the course Canvas website.

Create a new Git repository

If you do not already have an account on GitHub.com, create one now. Then start a new git repository. If you have never done this before, you can follow these instructions from biostatistician Karl Broman. Create a readme file (with the file extension .md) that includes your name and explains that this is an exploratory data task using traffc fatality data. GitHub has more information on using markdown for readme files.

Download the DoT's FARS Data for 2014 and 2015

For this exploratory data analysis task, we will use The Department of Transportation's (DoT) Fatality Analysis Reporting System (FARS) data. To get started, download both of the following:

- The 2014 **SAS Data** `ftp://ftp.nhtsa.dot.gov/FARS/2014/National/`
- The 2015 **CSV Data**: `ftp://ftp.nhtsa.dot.gov/FARS/2015/National/`

The data dictionary is available if you have any questions about the data: Link.

Grab the `accident.sas7bdat` and `accident.csv` files from each downloaded folder and add only these data files to your project folder **You can delete the rest of the downloaded data.**

Create a gitignore file that specifies that this git repository should ignore all files with the file extension of sas7bdat or csv. Add the URL links to these datasets to your Git's markdown document and identify them as the data source in plain text.

Git add the readme file and git commit.

About This Data

Traffic fatalities are on the rise in the United States. 40,200 people died in 2016 due to motor vehicle accidents, according to the National Safety Council. This is a 14 percent increase in the last two years, and the state of Illinois is fairing even worse.

Various experts attribute this increases to distracted driving, more speeding, lax enforcement of seatbelt laws, and a better economy (leading to more driving), but there is little consensus. Even before the dire 2016 numbers, the Obama White House and DoT even issued a call to action to analyze this data and help mitigate the problem.

Open RStudio and Install Pertinent Packages

Open a new RStudio session and create a new R script with the file extension: .R

Save this file into your project folder. Then add to the file comments including your name and a few words suggesting this file is an exploratory analysis of the DoT FARS data.

Next, use `install.packages()` to install the R packages: `readr`, `haven`, `dplyr`, `tidyr`, `stringr`, and `ggplot2`. Load these packages in this R session using the `library()` function. There are resources for learning more about these at the end of this assignment.

You should always save the lines of R code for loading the libraries in your .R file, since you would need to run this code every time you wanted to execute this file. You do not need to do the same for the `install.packages()` lines of code, since you would not need to run those to replicate the file.

Git add the R file and then git commit.

Load the FARS Data into R.

Use the `read_csv()` function from the `readr` package and the `read_sas()` function from the `haven` package to load in the two accident datasets as **acc2014** and **acc2015**. Use `ls()` to confirm these datasets have been read into R's memory.

```
ls()
```

```
## [1] "acc2014" "acc2015"
```

Use the `class()` function to examine the objects read into R - you should see they are of several classes. Most importantly for our purposes, they are (1) of class `data.frame`, which is R's original conception of a tabular dataset and (2) of class `tbl`, pronounced tibble. Tibbles inherit the dataframe class and can be considered a modern version of the dataframe with convenient defaults. You can read more about tibbles in this vignette Link.

Git commit.

Combining the two years of FARS data

The two years of FARS data are not formatted exactly the same way, so some changes need to be made before combining them. For instance, in acc2014, missing data in the `TWAY_ID2` variable is represented by an empty string, while in acc2015 it is represented by `NA`. `NA`, for Not Available, is the representation of missing data in R.

Use the `mutate()` function and `na_if()` helper function from the `dplyr` library to convert the missing values of `TWAY_ID2` in the 2014 tibble from empty strings to `NA` values. Keep the name of the variable the same.

If you have done this correctly, your results should match those below:

```
table(is.na(acc2014$TWAY_ID2))
```

```
##
## FALSE  TRUE
##  7792 22264
```

Next, run the `dim()` function (which prints number of rows then columns) on both tibbles and notice that the number of columns differs.

```
dim(acc2014)
```

```
## [1] 30056    50
```

```
dim(acc2015)
```

```
## [1] 32166    52
```

Use the `colnames()` function and the `%in%` operator, you can identify which columns appear in one dataset by not the other. Indexing with square brackets can be useful here as well. Write a comment identifying these columns (there are four in total) and which dataset(s) they are missing from.

Finally, use `bind_rows()` to combine these two tibbles into a single new tibble called `acc`. Use the `count()` function to create a frequency table of the `RUR_URB` variable - why are there over 30,000 NA values for this column? Explain in a comment.

Git commit.

Merging on another data source.

Read the fips.csv dataset into R (this file is attached to this assignment on Canvas) as `fips`. FIPS stands for "Federal Information Processing Standards" and is a common set of codes for United States counties provided by the US Census Bureau. You can use the `glimpse()` function to look at this data, which contains county and state names that are missing from FARS data.

```
glimpse(fips)
```

```
## Observations: 3,142
## Variables: 4
## $ StateName      <chr> "Alabama", "Alabama", "Alabama", "Alabama", "Al...
## $ CountyName     <chr> "Autauga", "Baldwin", "Barbour", "Bibb", "Bloun...
## $ StateFIPSCode  <chr> "01", "01", "01", "01", "01", "01", "01", "01",...
## $ CountyFIPSCode <chr> "001", "003", "005", "007", "009", "011", "013"...
```

The `fips` tibble contains columns named `State.FIPS.Code` and `County.FIPS.Code` whose values should almost match the codes in the `STATE` and `COUNTY` columns of the `acc` tibble. However, these values do not quite match, you must first:

- Convert the `State` and `County` variables in `acc` from integers to characters;
- Use the `str_pad()` function from the `stringr` package to add leading zeros so that all entries of the `State` variable have two characters and all values of the `County` variable have three characters.
- Rename the `State` and `County` variables in `acc` to `StateFIPSCode` and `CountyFIPSCode` - the `rename()` function may be valuable.

Merge the fips data onto the `acc` tibble using either `left_join()` or `right_join()` from the `dplyr` package. You do not want to lose any data within `acc`, so the number of rows in the `acc` tibble should not change. You will need to use both `StateFIPSCode` and `CountyFIPSCode`.

Git commit.

Exploratory Data Analysis in R's dplyr and tidyr package

- Use `group_by` and `summarize()` functions to calculate the total fatalities (using the `FATALS` variable) by state for each of the past two years. Save this new variable as `TOTAL` and the data as a new tibble called `agg`
- Use `tidyr`'s `spread()` function to change this new data format from long form to wide form. Save this new tibble as `agg_wide`. Each state should now occupy only one row of data, with one column for fatalities in 2014 and another column for fatalities in 2015.

Note: When using `spread()`, you may be default create new columns called `2014` and `2015`, which you may want to rename. Alternatively, you can still refer to them as such by surrounding those column names using grave accents: "`"

- Use `mutate()` to calculate the *percent difference* between 2014 and 2015 fatalities for each state.
- Use `arrange()` to sort the tibble from largest increase to biggest percent decline.
- Use `filter()` to subset the data down to only the states with more than a 15% increase. Also remove the `NA` state value.
- Rewrite the prior six steps using the same functions, but now only using one assignment `<-` with `dplyr`'s chain operator: `%>%`

Your results should look like this:

```
glimpse(agg)
```

```
## Observations: 12
## Variables: 4
## $ StateName    <chr> "Arizona", "Maryland", "Idaho", "Montana", "Flori...
## $ Year2014     <dbl> 773, 442, 186, 192, 2494, 823, 131, 95, 1164, 462...
## $ Year2015     <dbl> 891, 513, 216, 224, 2939, 977, 156, 114, 1430, 56...
## $ Diff_Percent <dbl> 0.1526520, 0.1606335, 0.1612903, 0.1666667, 0.178...
```

Git commit & push.

Submit the URL for this GitHub repository to the Canvas assignment.

Pertinent Resources

Introduction to haven package Link

Introduction to readr package Link

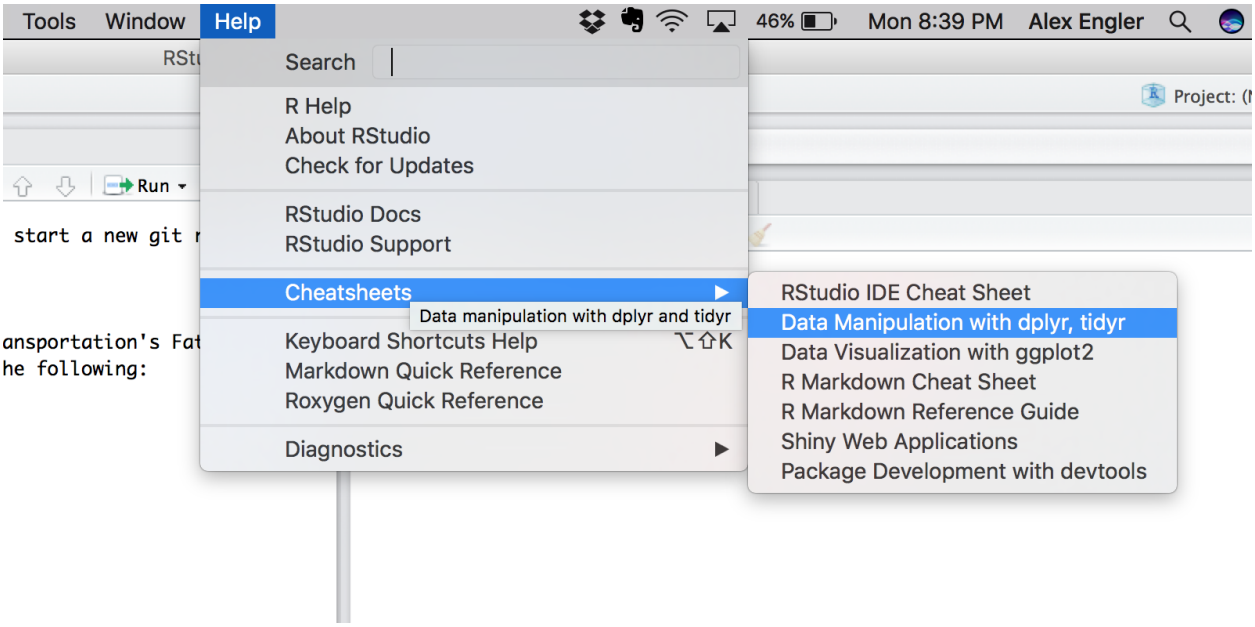
Introduction to readxl package Link

Vignette on dplyr package for Data Manipulation Link

Data Processing with dplyr & tidyr Link

String Manipulation with stringr Link

In the image below, 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 free ebook R for Data Science.



Next Up... ggplot2

This is just a preview of what we will start next week, nothing that you need to do.

```
agg %>%
  ggplot() +
    geom_segment(aes(x="2014", xend="2015", y=Year2014, yend=Year2015, color=factor(StateName))) +
    geom_text_repel(aes(x="2014", y=Year2014, label=StateName, color=StateName),
                    size=3, nudge_x=-0.1) +
    scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                  labels = trans_format("log10", math_format(10^.x))) +
    annotation_logticks(sides="l") +
    labs(title = "Traffic Fatalities Rise in Many States",
       subtitle = "12 States Saw a 15% or Greater Rise in Traffic Fatalities",
       caption = "DoT FARS Data ",
       x = "Year",
       y = "Traffic Fatalities (Log Base 10)") +
    theme(legend.position="none",
       panel.background = element_blank(),
       plot.margin = margin(0.25, 0.25, 0.25, 0.25, "cm"))
```