Week 1 Practice

2022-09-23

Let's just practice around in R to get a feel for how R performs its operations.

We'll end the session by pushing our final script to the GitHub repository you made in class. (Or if you didn't make one, make one now for this class :]).

First, we'll download this data from a `tidytuesday` post. In the spirit of dealing with some data that has to do with political issues, let's take a look at this data from the Vera Institute about incarceration trends. I've chosen the `incarceration_trends.csv`, which is the full raw data provided on the post.

We'll load in the data and then also remember to go ahead and load the packages that you think you will need to do your data exploration. For now, `tidyverse` is probably sufficient.

```
incarceration <- read.csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/20

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Exploring the Data

Before you can do really anything with the data, you will want to check for a codebook or any equivalent documentation of the properties of the data.

The entire codebook is really quite long (43 pages!), which is great because that means the data is well documented. This is the best case scenario for visualization and analysis. I will save myself from just rehashing the entire codebook here, and suggest that if you want to play with this dataset beyond what I provide here, then please visit the site link.

If you'll notice, the dataset that I have chosen here is in "raw" format. What does that mean? It means that the data is in its collected format, there are no summaries that have been conducted, and the data in this case is **wide**. What is wide data? Well we can take a look at the data to find out more.

To begin with, I always suggest looking at your data in two ways, depending on your level of familiarity with the data to begin with. The first is with the `glimpse()` function from `dplyr`– a package you loaded in with the `tidyverse`. The only **argument** that goes into `glimpse()` for now is our dataset name.

In the instance that you are already acquainted with your data, that is you already know the variables in there and the general structure and just want to be reminded of the dimensions of your data, you can instead run the function `dim()`. This command also takes just the dataset name as the argument.

```r
# option 1

glimpse(incarceration)
```

```
## Rows: 147,533
## Columns: 79
## $ yfips                       <int> 197001001, 197101001, 197201001, 19730~
## $ year                        <int> 1970, 1971, 1972, 1973, 1974, 1975, 19~
## $ fips                        <int> 1001, 1001, 1001, 1001, 1001, 1001, 10~
## $ state                       <chr> "AL", "AL", "AL", "AL", "AL", "AL", "A~
## $ county_name                 <chr> "Autauga County", "Autauga County", "A~
## $ total_pop                   <int> 24661, 25503, 27156, 28453, 29261, 297~
## $ total_pop_15to64            <int> 14154, 14765, 15939, 16906, 17578, 180~
## $ female_pop_15to64           <int> 7293, 7585, 8168, 8651, 8992, 9210, 94~
## $ male_pop_15to64             <int> 6861, 7180, 7771, 8255, 8586, 8797, 90~
## $ asian_pop_15to64            <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ black_pop_15to64            <int> 3413, 3451, 3625, 3747, 3791, 3804, 38~
## $ latino_pop_15to64           <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ native_pop_15to64           <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ other_pop_15to64            <int> 15, 24, 31, 41, 49, 58, 66, 74, 81, 95~
## $ white_pop_15to64            <int> 10726, 11290, 12283, 13118, 13738, 141~
## $ urbanicity                  <chr> "small/mid", "small/mid", "small/mid",~
## $ region                      <chr> "South", "South", "South", "South", "S~
## $ division                    <chr> "East South Central", "East South Cent~
## $ commuting_zone              <int> 60, 60, 60, 60, 60, 60, 60, 60, 60, 60~
## $ metro_area                  <int> 33860, 33860, 33860, 33860, 33860, 338~
## $ land_area                   <dbl> 594.449, 594.449, 594.449, 594.449, 59~
## $ total_jail_adm              <dbl> 0.0000, 0.0000, 0.0000, 0.0000, 0.0000~
## $ total_jail_adm_dcrp         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ female_jail_adm_dcrp        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ male_jail_adm_dcrp          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ total_jail_pop              <dbl> 0.00000, 0.00000, 0.00000, 0.00000, 0.~
## $ female_jail_pop             <dbl> 0.000000, 0.000000, 0.000000, 0.000000~
## $ male_jail_pop               <dbl> 0.00000, 0.00000, 0.00000, 0.00000, 0.~
## $ asian_jail_pop              <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ black_jail_pop              <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ latino_jail_pop             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ native_jail_pop             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ white_jail_pop              <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ total_jail_pretrial         <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0~
## $ female_jail_pretrial        <dbl> 0.000000, 0.000000, 0.000000, 0.000000~
## $ male_jail_pretrial          <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0~
## $ jail_from_state_prison      <dbl> 0.000000, 0.000000, 0.000000, 0.000000~
## $ jail_from_other_state_prison <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ jail_from_state_jail        <dbl> 0.0000, 0.0000, 0.0000, 0.0000, 0.0000~
## $ jail_from_other_state_jail  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ jail_from_fed               <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ jail_from_ice               <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ total_jail_pop_dcrp         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ female_jail_pop_dcrp        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ male_jail_pop_dcrp          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ total_prison_pop            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ female_prison_pop           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

```
## $ male_prison_pop             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ asian_prison_pop            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ black_prison_pop            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ latino_prison_pop           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ native_prison_pop           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ other_prison_pop            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ white_prison_pop            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ total_prison_adm            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ female_prison_adm           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ male_prison_adm             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ asian_prison_adm            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ black_prison_adm            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ latino_prison_adm           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ native_prison_adm           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ other_prison_adm            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ white_prison_adm            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ num_facilites               <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ num_employees               <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ confined_pop                <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ capacity                    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ ucr_population              <int> NA, NA, NA, NA, NA, NA, NA, 30000, 292~
## $ index_crime                 <int> NA, NA, NA, NA, NA, NA, NA, 697, 672, ~
## $ violent_crime               <int> NA, NA, NA, NA, NA, NA, NA, 42, 21, NA~
## $ property_crime              <int> NA, NA, NA, NA, NA, NA, NA, 655, 651, ~
## $ murder_crime                <int> NA, NA, NA, NA, NA, NA, NA, 5, 2, NA, ~
## $ rape_crime                  <int> NA, NA, NA, NA, NA, NA, NA, 7, 4, NA, ~
## $ robbery_crime               <int> NA, NA, NA, NA, NA, NA, NA, 12, 7, NA,~
## $ agr_assault_crime           <int> NA, NA, NA, NA, NA, NA, NA, 18, 8, NA,~
## $ burglary_crime              <int> NA, NA, NA, NA, NA, NA, NA, 183, 146, ~
## $ larceny_crime               <int> NA, NA, NA, NA, NA, NA, NA, 422, 451, ~
## $ mv_theft_crime              <int> NA, NA, NA, NA, NA, NA, NA, 50, 54, NA~
## $ arson_crime                 <int> NA, NA, NA, NA, NA, NA, NA, 0, 0, NA, ~
```

```
#147,533 obs over 79 vairables

# option 2

dim(incarceration)
```

```
## [1] 147533     79
```

In the case of the `glimpse()` function, you will see that this gives us so much more information that it's almost worthwhile to do it regardless of whether we already know the data or not. This is also useful and will be useful for the visualization perspective in that we see *how* certain variables are stored. Our options for visualizing the `year` variable look much different when R treats this as a integer value as opposed to a numeric value. Equivalently, seeing the types of data on their own will lead to intuitive choices for the *appropriate* type of graph in certain instances. For example, the variable named `urbanicity` in this dataset is coded as a character variable with the values `small/mid`, `rural`, `suburban`, or `urban`. Our choices to visualize the data for this variable look much different than if the urbanicity were scored on a numeric scale from 0 to 100.

Let's continue to explore.

Remember that when we are dealing in base R, if we want R to index a certain variable in a dataset that we

use the dollar sign ($) operator appended to the dataset name. For example, if I wanted to see the unique values of the variable `urbanicity` (as we did above, I would run the following command).

```
unique(incarceration$urbanicity)

# the returned result are the unique values of the urbanicity variable

# maybe we just want to see all of the names of the variables

names(incarceration)
```

One important facet of data to always explore is the presence of NA values. These are missing observations. In case of survey data, this could mean non-response. In other cases, it might mean the value can't be found for that particular case. In other cases, it might indicate that a value really is not applicable for that case at that particular observation point. We can work throughout the quarter to discern when misisng values are a problem.

For now we will see what methods we have for checking for NAs. Let's check for this on the variable `total_jail_adm`, whcih is the total jail admissions count.

```
is.na(incarceration$total_jail_adm)
#gives us a bunch of logical values as to whether the observation is an NA or not.... not so useful

#instead we can see how many NA values are in this variable

sum(is.na(incarceration$total_jail_adm))
#6,300 missing values

# we can also perform arithmetic on these functions
# for example what percentage of our total observations are NA on this variable?
# we can do this two ways, 1) divide by the raw number of total observations (147,533)
# 2) index the number of observations given the dim() function

# option 1
# missing values in the variable / total number of observations in data
sum(is.na(incarceration$total_jail_adm))/ 147533 #0.043

# option 2
# missing values in the variable / dim(incarceration)[1], this just indexes the same number as option 1
sum(is.na(incarceration$total_jail_adm))/ dim(incarceration)[1] #0.043

# result is the same!
```

Saving things as objects

We talked in class how R is an object-oriented coding language. We also created a couple of different objects.

There are a couple of different objects that we can create, atomic vectors, lists, matrices, dataframes. We will really only focus on the vectors and dataframes.

We can also easily create a bunch of one type of object and combine then into another object type. To do so, we'll just come up with some toy data.

```r
# the "c" is for concatenate or combine!
# it just means we want all of the stuff from that parentheses to be included in the defined object

# let's just throw some countries together
country <- c("USA", "Mexico", "Afghanistan", "Thailand")

# and their respective continents
continent <- c("North America", "North America", "Asia", "Asia")

# and some information on their geographical location
equator <- c(0, 1, 0, 1) # whether the country is above the equator (0), at the equator/between the tro

# combine together with the command data.frame() and each of the variable names
country_df <- data.frame(country, continent, equator)

country_df
```

```
##       country     continent equator
## 1         USA North America       0
## 2      Mexico North America       1
## 3 Afghanistan          Asia       0
## 4    Thailand          Asia       1
```

Note that the order of the information in the variables is important. If you want to merge together data, you need to make sure that the data is ordered in a way that makes sense *or* that you have a key variable to merge on. We will explore this more next week.

We might also want to create objects to store output information from certain summary operations. And later on– plots! You can save anything as an object in R, as long as you use that `<-` assignment operator.

As an example, we'll return to the incarceration data variable `incarceration$total_jail_adm` from above:

```r
# can either just run the command
mean(incarceration$total_jail_adm, na.rm = T)
```

```
## [1] 2704.248
```

```r
# or can store the output as an object
mean_jail_adm <- mean(incarceration$total_jail_adm, na.rm = T)
mean_jail_adm
```

```
## [1] 2704.248
```

Pushing the script and the incarceration file to R

We can also save stuff from R onto our computer. Let's save the dataset that you loaded in. Remember that it came from the internet, so we don't actually have a copy on our computer.

First, set your working directory to the local version of your GitHub file. Then, save the incarceration data to the working directory. Last, we'll push the uploaded file to the GitHub remote repository using GitHub Desktop (you'll need to make sure this file is being watched by GitHub Desktop, if you have a question about this email me!).

```r
# 1) check working directory

getwd()

# 2) set the working directory

setwd("/Users/sarahmoore/Desktop/data_viz_390/")

# 3) save the incarceration data as a csv to the file

write.csv(incarceration, "incarceration_trends.csv")

# 4) check the data is in the file specified

# 5) push to the associated GitHub
```

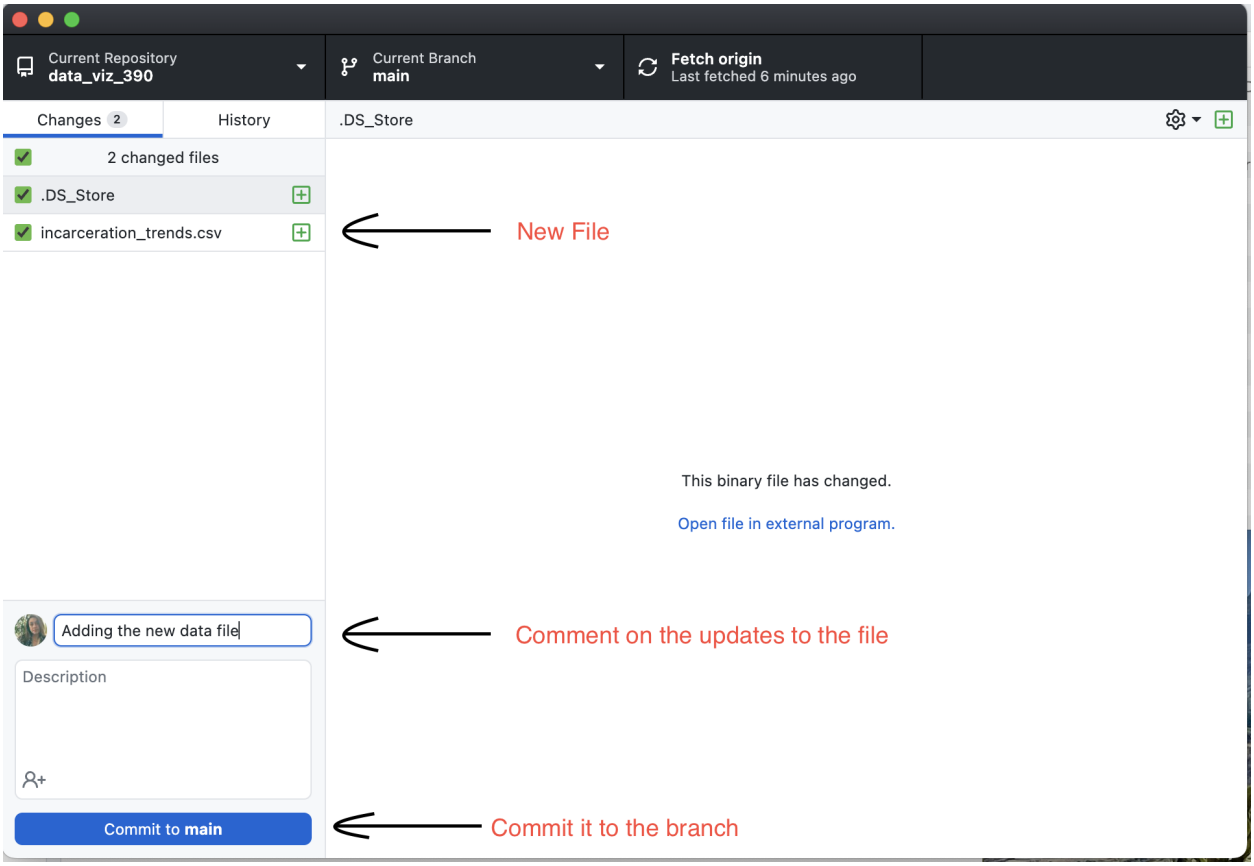

Figure 1: Follow these steps

But committing the file does not actually send the file to the remote repository. So then we have to push the commit.

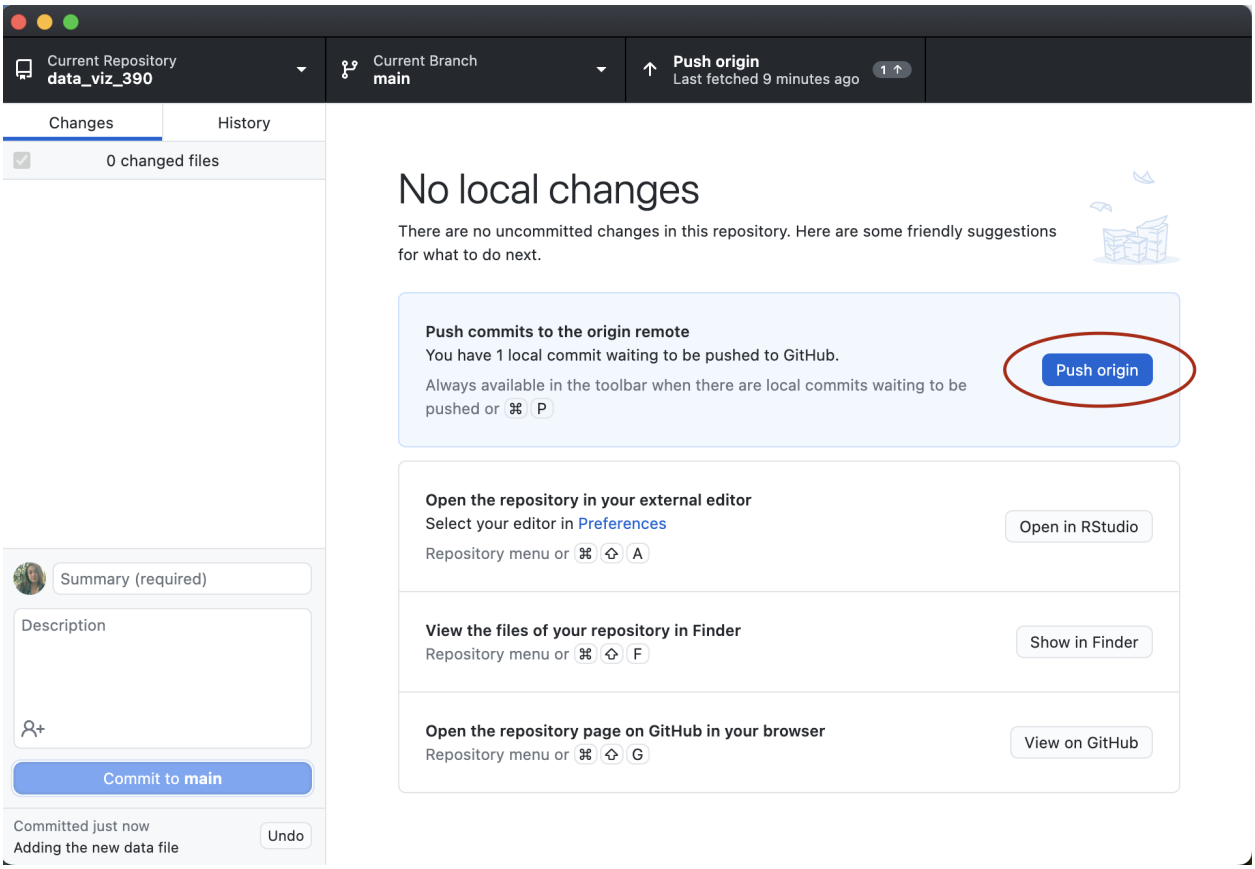

Figure 2: Push!