Math camp coding self-assessment

Ari Anisfeld

6/13/2020

As part of the statistics curriculum, you will be asked to analyze data using the programming language R. R is an open source language that is widely used by data analysts.

This is a self-assessment. If you feel comfortable completing this assignment by yourself (with the help of google), than you are free to skip coding lab. In summer coding lab, we provide an introduction to R coding focused on data analysis. In fall summer coding lab, we repeat some of the data analysis material and also introduce using R for programming.

Task 1:

1. Install R and Rstudio.
2. Install the package `readxl` and `tidyverse`.
3. Adjust the following code block to read in the provided data set `incarceration_counts_and_rates_by_type_over_time`

```
library(tidyverse)
library(readxl)
setwd(<Put path to file here>)
incarceration_data <- read_xlsx("incarceration_counts_and_rates_by_type_over_time.xlsx",
               range = "A7:CO10") %>%
    rename("type" = ...1) %>%
    pivot_longer(`1925`:`2016`, names_to = "year", values_to = "counts")
```

1. What does the code `library(readxl)` do and why is it necessary?
2. Why do you need to set a working directory (`setwd()`)?

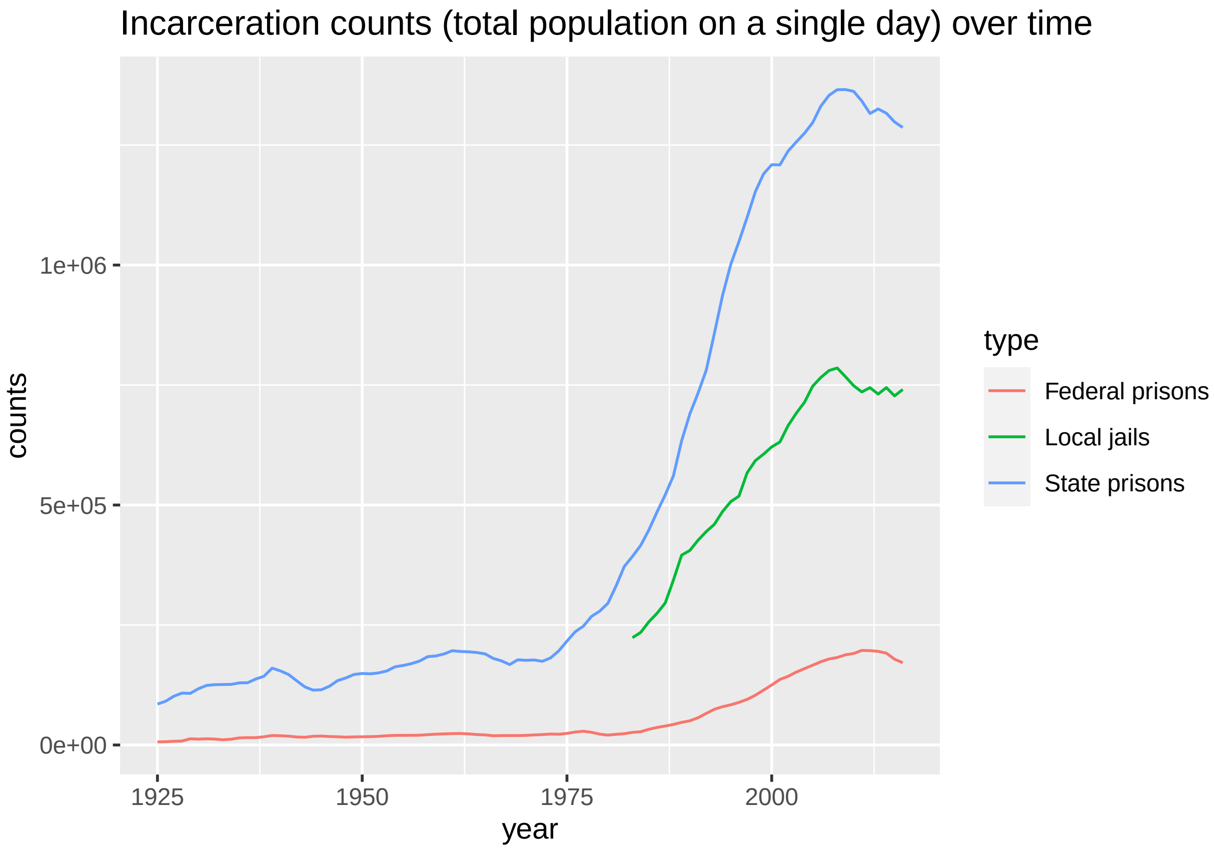
If you had trouble with readxl, we provide a csv as well. You can load the data with the following code: `incarceration_data <- read_csv("incarceration_counts_and_rates_by_type_over_time.csv")`

Task 2:

You want to make a graph visualizing the change in incarceration counts in the United States over time.

```
incarceration_data %>%
  ggplot(aes(x = year, y = counts, color = type)) +
  geom_line() +
  labs(title = "Incarceration counts (total population on a single day) over time")
```

The following code doesn't work, because `year` is stored as characters. Change `year` data type to numeric and you'll get the following graph.



Task 3:

We want to analyze state prison counts by decade. We'll prepare the data in the following ways. Store the following changes in a new tibble called `state_data`.

1. Add a column called `decade` that reflects which decade the observation comes from.
2. Filter the data so that you only have data from state prisons.
3. Use `select` to reorder the columns so that your data is organized as below:

```
## # A tibble: 4 x 4
##   type           counts decade  year
##   <chr>           <dbl>  <dbl> <dbl>
## 1 State prisons  85239   1920  1925
## 2 State prisons  91188   1920  1926
## 3 State prisons 101624   1920  1927
## 4 State prisons 108157   1920  1928
```

Task 4:

In this section, you'll use `group_by()` and `summarize()` to answer questions about state prision counts by decade.

1. Which decade saw the largest percentage growth in State prisons? Measure percent growth as $\frac{C_{d_e} - C_{d_s}}{C_{d_s}}$ where C_{d_e} is the count at the end of decade and C_{d_s} is the start of the decade). You can use the $first()$ and $last()$ functions.

```
## # A tibble: 10 x 2
##    decade percentage_growth
##     <dbl>             <dbl>
##  1   1920            0.262
##  2   1930            0.365
##  3   1940           -0.0490
##  4   1950            0.245
##  5   1960           -0.0644
##  6   1970            0.581
##  7   1980            1.15
##  8   1990            0.725
##  9   2000            0.129
## 10   2010           -0.0553
```

Task 5:

Miscellanous tasks: We leave the data behind and test skills.

1. Take `numbers <- rep(seq(-9, 10, 1), 10)`. Show that the mean of the vector is .5 and that the sum of the components is 100.
2. Adjust the call to median, so that we ignore the NA value and return 3.

   ```
   toy_data <- c(1, 2, 3, NA, 4, 5)

   median(toy_data)
   ```

   ```
   ## [1] NA
   ```
3. Use brackets to extract the number 4 from `toy_data`.
4. Combine the strings assigned to `left` and `right` into a single string using an an R function.

   ```
   left <- "Harris"
   right <- "School of Public Policy"
   ```

Task 6:

These questions reflect skills covered in the fall coding lab and not during summer coding camp.

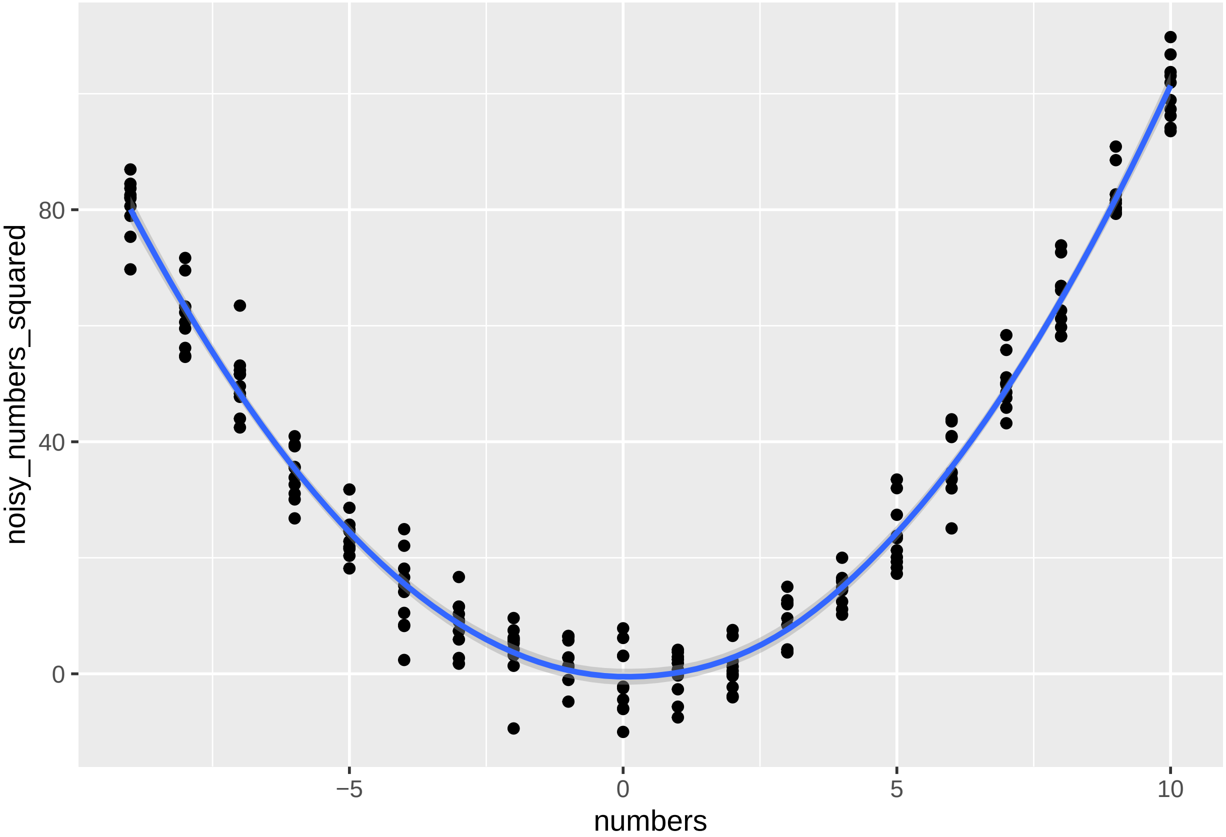
1. For loops: Take `numbers <- rep(seq(-9, 10, 1), 10)`. Using a for-loop, save the square of each number in a new vector called `numbers_squared`.
2. For loops: Take `numbers`. Using a for-loop, save the square of each number and add random noise using a call to `rnorm(1, sd = 5)` in a new vector called `noisy_numbers_squared`.

You should be able to reproduce this graph:

```
numbers_data <- tibble(numbers = numbers,
                       noisy_numbers_squared = noisy_numbers_squared)

numbers_data %>%
  ggplot(aes(x = numbers, y = noisy_numbers_squared)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



1. Functions: Write a function that takes a name as an input and adds “is a boss” to the name like so:

```
add_is_a_boss("Kate Shannon Biddle")
```

```
## [1] "Kate Shannon Biddle is a boss"
```