DATA 300 3 Homework 1 Solution

Aadarsha Gopala Reddy

September 8, 2022

Contents

1. Load Packages

In a chunk of code, load the tidyverse package and any other packages you will use in this document.

`install.packages()` Installs any packages.

`library()` Loads any packages.

```
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   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

2. Import Data

Import *protest_data.csv* as an object in R. Print the first few rows of the dataset using the `head()` function.

`read_csv()` reads a csv file into R.

`head()` prints the first few rows of a dataset.

```
protest_data <- read_csv("protest_data.csv")
```

```
## Rows: 21282 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (7): event_date, event_type, sub_event_type, admin1, source, notes, loca...
## dbl (4): year, fatalities, latitude, longitude
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(protest_data)
```

```
## # A tibble: 6 x 11
##   event_date    year event~1 sub_e~2 admin1 source notes fatal~3 locat~4 latit~5
##   <chr>        <dbl> <chr>   <chr>   <chr>  <chr>  <chr>   <dbl> <chr>     <dbl>
## 1 12 August 2~  2022 Protes~ Peacef~ Calif~ KTVU ~ On 1~       0 Martin~    38.0
## 2 12 August 2~  2022 Protes~ Peacef~ Texas  News ~ On 1~       0 Midland    32.0
## 3 12 August 2~  2022 Protes~ Peacef~ Calif~ NewsC~ On 1~       0 Palm S~    33.8
## 4 12 August 2~  2022 Protes~ Peacef~ Georg~ 11 Al~ On 1~       0 Atlanta    33.8
## 5 12 August 2~  2022 Protes~ Peacef~ New Y~ 13WHA~ On 1~       0 Batavia    43.0
## 6 12 August 2~  2022 Protes~ Peacef~ Delaw~ Fox29  On 1~       0 Wilmin~    39.7
## # ... with 1 more variable: longitude <dbl>, and abbreviated variable names
## #   1: event_type, 2: sub_event_type, 3: fatalities, 4: location, 5: latitude
```

3. Filter Data

Subset the protest dataset to only the protests that occurred in the year 2022. Make sure to save your changes to the dataset.

> `filter()` filters a dataset based on a condition. This was used to filter the dataset to include only the protests that occurred in 2022.
>
> `%>%` pipes the output of one function into the input of another function. `tally()` counts the number of rows in a dataset. Both were used together to calculate the number of protests that occurred in 2022.

```
protest_data <- filter(protest_data, year == 2022)
head(protest_data)
```

```
## # A tibble: 6 x 11
##   event_date    year event~1 sub_e~2 admin1 source notes fatal~3 locat~4 latit~5
##   <chr>        <dbl> <chr>   <chr>   <chr>  <chr>  <chr>   <dbl> <chr>     <dbl>
## 1 12 August 2~  2022 Protes~ Peacef~ Calif~ KTVU ~ On 1~       0 Martin~    38.0
## 2 12 August 2~  2022 Protes~ Peacef~ Texas  News ~ On 1~       0 Midland    32.0
## 3 12 August 2~  2022 Protes~ Peacef~ Calif~ NewsC~ On 1~       0 Palm S~    33.8
## 4 12 August 2~  2022 Protes~ Peacef~ Georg~ 11 Al~ On 1~       0 Atlanta    33.8
## 5 12 August 2~  2022 Protes~ Peacef~ New Y~ 13WHA~ On 1~       0 Batavia    43.0
## 6 12 August 2~  2022 Protes~ Peacef~ Delaw~ Fox29  On 1~       0 Wilmin~    39.7
## # ... with 1 more variable: longitude <dbl>, and abbreviated variable names
## #   1: event_type, 2: sub_event_type, 3: fatalities, 4: location, 5: latitude
```

```
protest_data %>% tally()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1  8305
```

How many protests occurred in the year 2022? **8305 protests.**

4. Create a New Variable

Create a new variable in the protest dataset called `cont_us`. This binary variable should measure whether the protest occurred in the contiguous United States. - Protests that occurred in Hawaii or Alaska should have a value of 0. - Protests that occurred in the other 48 states and Washington D.C. should have a value of 1.

> `if_else()` creates a new variable based on a condition. This was used to create the `cont_us` variable.
>
> `select()` selects a subset of columns from a dataset. This was used to select the `state` and `cont_us` column from the dataset.
>
> `filter()` and `%>%` was used to calculate the number of protests that occurred outside of the contiguous United States.

```
protest_data$cont_us <- if_else(protest_data$admin1 %in% c("Hawaii", "Alaska"), 0, 1)
select(protest_data, "admin1", "cont_us")
```

```
## # A tibble: 8,305 x 2
##    admin1     cont_us
##    <chr>        <dbl>
##  1 California       1
##  2 Texas            1
##  3 California       1
##  4 Georgia          1
##  5 New York         1
##  6 Delaware         1
##  7 California       1
##  8 California       1
##  9 Michigan         1
## 10 Ohio             1
## # ... with 8,295 more rows
```

```
filter(protest_data, cont_us == 0)  %>% tally()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    66
```

How many protests occurred outside of the contiguous United States? **66 protests occurred outside the contiguous USA.**

5. Transform the Dataset

Transform the protest-level dataset into a state-level dataset and save it as a new object in R. In the new dataset, each observation should be a state or territory of the United States. It should contain the following variables:

- state: the name of the state.
- fatalities: the total number of fatalities at protests in that state in 2022.
- count: a count of the number of protests in that state in 2022.

Print the first few rows of the new dataset using the `head()` function.

> `group_by()` groups a dataset by a variable. This was used to group the dataset by state.

`summarize()` summarizes a dataset by a variable. This was used to calculate the total number of fatalities and the number of protests in each state.

`%>%` was used to pipe the output of `group_by()` into the input of `summarize()`.

```
state_data <- protest_data %>% group_by(admin1) %>% summarise(fatalities = sum(fatalities), count = n())
head(state_data)
```

```
## # A tibble: 6 x 3
##   admin1     fatalities count
##   <chr>           <dbl> <int>
## 1 Alabama             0   103
## 2 Alaska              0    18
## 3 Arizona             0   117
## 4 Arkansas            0    59
## 5 California          0  1174
## 6 Colorado            0    92
```

6. Sort the State-Level Data

Sort the state-level dataset by the number of protests occurring in the state. Using a method of your choosing, display the names and number of protests for the 5 states with the most protests in 2022.

`arrange()` arranges a dataset by a variable. This was used to arrange the dataset by the number of protests in each state.

`select()` was used to select the `state` and `count` columns from the dataset.

`%>%` was again used to pipe the output of `arrange()` into the input of `select()`.

```
state_data <- state_data %>% arrange(desc(count)) %>% select(admin1, count)
head(state_data)
```

```
## # A tibble: 6 x 2
##   admin1        count
##   <chr>         <int>
## 1 California     1174
## 2 New York        869
## 3 Florida         522
## 4 Pennsylvania    434
## 5 Texas           342
## 6 Massachusetts   334
```

7. Merge Two Datasets

Import a second dataset, `state.csv`. Each observation in this dataset is a state. This dataset contains two variables:

- state: the name of the state.
- population: the number of people living in the state.

Join your state-level dataset together with state.csv such that the population variable appears in your state-level dataset. Use the `head()` function to print your state-level dataset with the population variable included.

`left_join()` joins two datasets together. This was used to join the state-level dataset with the state.csv dataset.

`select()` was used to select the `state`, `count`, and `population` columns from the dataset.

`%>%` was again used to pipe the output of `left_join()` into the input of `select()`.

```
state_population <- read_csv("state.csv")
```

```
## Rows: 50 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): state
## dbl (1): population
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
state_data <- left_join(state_data, state_population, by = c("admin1" = "state"))
head(state_data)
```

```
## # A tibble: 6 x 3
##   admin1        count population
##   <chr>         <int>      <dbl>
## 1 California     1174      21198
## 2 New York        869      18076
## 3 Florida         522       8277
## 4 Pennsylvania    434      11860
## 5 Texas           342      12237
## 6 Massachusetts   334       5814
```