p8180_hw2_jj3205

Jia Ji (jj3205)

2022-10-04

Problem 1

```
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 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(readxl)
```

Below we import and clean data from `NYC_Transit_Subway_Entrance_And_Exit_Data.csv`. The process begins with data import, updates variable names, and selects the columns that will be used in later parts fo this problem. We update `entry` from `yes` / `no` to a logical variable. As part of data import, we specify that `Route` columns 8-11 should be character for consistency with 1-7.

```
trans_ent =
  read_csv(
    "NYC_Transit_Subway_Entrance_And_Exit_Data.csv",
    col_types = cols(Route8 = "c", Route9 = "c", Route10 = "c", Route11 = "c")) %>%
  janitor::clean_names() %>%
  select(
    line, station_name, station_latitude, station_longitude,
    starts_with("route"), entry, exit_only, vending, entrance_type,
    ada) %>%
  mutate(entry = ifelse(entry == "YES", TRUE, FALSE))
```

As it stands, these data are not "tidy": route number should be a variable, as should route. That is, to obtain a tidy dataset we would need to convert `route` variables from wide to long format. This will be useful when focusing on specific routes, but may not be necessary when considering questions that focus on station-level variables.

The following code chunk selects station name and line, and then uses `distinct()` to obtain all unique combinations. As a result, the number of rows in this dataset is the number of unique stations.

```
trans_ent %>%
  select(station_name, line) %>%
  distinct
```

```
## # A tibble: 465 x 2
##    station_name             line
```

```
##    <chr>                    <chr>
##  1 25th St                  4 Avenue
##  2 36th St                  4 Avenue
##  3 45th St                  4 Avenue
##  4 53rd St                  4 Avenue
##  5 59th St                  4 Avenue
##  6 77th St                  4 Avenue
##  7 86th St                  4 Avenue
##  8 95th St                  4 Avenue
##  9 9th St                   4 Avenue
## 10 Atlantic Av-Barclays Ctr 4 Avenue
## # ... with 455 more rows
```

The next code chunk is similar, but filters according to ADA compliance as an initial step. This produces a dataframe in which the number of rows is the number of ADA compliant stations.

```
trans_ent %>%
  filter(ada == TRUE) %>%
  select(station_name, line) %>%
  distinct
```

```
## # A tibble: 84 x 2
##    station_name                  line
##    <chr>                         <chr>
##  1 Atlantic Av-Barclays Ctr      4 Avenue
##  2 DeKalb Av                     4 Avenue
##  3 Pacific St                    4 Avenue
##  4 Grand Central                 42nd St Shuttle
##  5 34th St                       6 Avenue
##  6 47-50th Sts Rockefeller Center 6 Avenue
##  7 Church Av                     6 Avenue
##  8 21st St                       63rd Street
##  9 Lexington Av                  63rd Street
## 10 Roosevelt Island              63rd Street
## # ... with 74 more rows
```

To compute the proportion of station entrances / exits without vending allow entrance, we first exclude station entrances that do not allow vending. Then, we focus on the `entry` variable – this logical, so taking the mean will produce the desired proportion (recall that R will coerce logical to numeric in cases like this).

```
trans_ent %>%
  filter(vending == "NO") %>%
  pull(entry) %>%
  mean
```

```
## [1] 0.3770492
```

Lastly, we write a code chunk to identify stations that serve the A train, and to assess how many of these are ADA compliant. As a first step, we tidy the data as alluded to previously; that is, we convert `route` from wide to long format. After this step, we can use tools from previous parts of the question (filtering to focus on the A train, and on ADA compliance; selecting and using `distinct` to obtain dataframes with the required stations in rows).

```
trans_ent %>%
  pivot_longer(
    route1:route11,
    names_to = "route_num",
    values_to = "route") %>%
```

```
  filter(route == "A") %>%
  select(station_name, line) %>%
  distinct
```

```
## # A tibble: 60 x 2
##    station_name                  line
##    <chr>                         <chr>
##  1 Times Square                  42nd St Shuttle
##  2 125th St                      8 Avenue
##  3 145th St                      8 Avenue
##  4 14th St                       8 Avenue
##  5 168th St - Washington Heights 8 Avenue
##  6 175th St                      8 Avenue
##  7 181st St                      8 Avenue
##  8 190th St                      8 Avenue
##  9 34th St                       8 Avenue
## 10 42nd St                       8 Avenue
## # ... with 50 more rows
```

```
trans_ent %>%
  pivot_longer(
    route1:route11,
    names_to = "route_num",
    values_to = "route") %>%
  filter(route == "A", ada == TRUE) %>%
  select(station_name, line) %>%
  distinct
```

```
## # A tibble: 17 x 2
##    station_name                  line
##    <chr>                         <chr>
##  1 14th St                       8 Avenue
##  2 168th St - Washington Heights 8 Avenue
##  3 175th St                      8 Avenue
##  4 34th St                       8 Avenue
##  5 42nd St                       8 Avenue
##  6 59th St                       8 Avenue
##  7 Inwood - 207th St             8 Avenue
##  8 West 4th St                   8 Avenue
##  9 World Trade Center            8 Avenue
## 10 Times Square-42nd St          Broadway
## 11 59th St-Columbus Circle       Broadway-7th Ave
## 12 Times Square                  Broadway-7th Ave
## 13 8th Av                        Canarsie
## 14 Franklin Av                   Franklin
## 15 Euclid Av                     Fulton
## 16 Franklin Av                   Fulton
## 17 Howard Beach                  Rockaway
```

Problem 2

Problem 3