p8105_hw2_WL3011

Weiqi Liang

2024-10-02

Setup File

```
library(tidyverse)
library(dplyr)
library(readxl)
library(haven)
library(knitr)
library(kableExtra)
```

I. Problem 1

1.1 Load the NYC Subway Dataset

- Retained columns: `Line`, `Station Name`, `Station Latitude`, `Station Longitude`, `Route1`:`Route11`, `Entry`, `Vending`, `Entrance Type`, `ADA`;
- Converted the `Entry` variable from "YES" and "NO" to logical TRUE and FALSE.

```
# Load the NYC Subway csv
subway_df =
  read.csv("./NYC_Transit_Subway_Entrance_And_Exit_Data.csv",
           na = c("NA", ".", "")) |>
  janitor::clean_names() |>
  select(line, station_name, station_latitude, station_longitude,
         route1:route11, entry, vending, entrance_type, ada) |>
  mutate(
    entry_logical = case_match(
      entry,
      "YES" ~ TRUE,
      "NO"  ~ FALSE
      )
    )
```

The dataset now is tidy, with each row representing a station entrance and all columns having consistent data types. It has 1868 rows and 20 columns, showing New York City subway station entrances/exits. The columns include:

Station details: Line, Station Name

Routes: Route1 to Route11, showing the subway lines served at the station

Facilities: Entrance Type, Entry, Vending, ADA (compliance), and ADA Notes

Geographical details: Entrance Latitude/Longitude, North/South and East/West Streets

1.2 Answering the Following Question

1. How many distinct stations are there?

```
# calculate the number of distinct stations (identified by name and line)
distinct_stations = subway_df |>
  distinct(station_name, line) |>
  nrow()
```

There are 465 distinct stations, identified by their name and line.

2. How many stations are ADA compliant?

```
# the number of ADA compliant stations
ada_compliant_stations = subway_df |>
  filter(ada == TRUE) |>
  distinct(station_name, line) |>
  nrow()
```

There are 84 ADA compliant stations.

3. What proportion of station entrances / exits without vending allow entrance?

```
no_vending_entry = subway_df |>
  filter(vending == "NO") |>
  filter(entry == "YES" ) |>
  nrow()

no_vending = subway_df |>
  filter(vending == "NO") |>
  nrow()

proportion = no_vending_entry/no_vending
```

The proportion of station entrances without vending allow entrance is 0.3770492.

1.3 Reformat Dataset

1. Reformat data so that route number and route name are distinct variables.

```
# Split route to long format (pivot_longer)
Reformat_subway_df = subway_df |>
  mutate_at(vars(route1:route11), as.character) |>
  pivot_longer(
    cols = route1:route11,
    names_to = "route_number",
    values_to = "route_name"
    ) |>
  filter(!is.na(route_name))  # remove NA

head(Reformat_subway_df)
```

```
## # A tibble: 6 x 11
##   line     station_name station_latitude station_longitude entry vending
##   <chr>    <chr>                   <dbl>             <dbl> <chr> <chr>
## 1 4 Avenue 25th St                  40.7             -74.0 YES   YES
## 2 4 Avenue 25th St                  40.7             -74.0 YES   YES
## 3 4 Avenue 36th St                  40.7             -74.0 YES   YES
## 4 4 Avenue 36th St                  40.7             -74.0 YES   YES
## 5 4 Avenue 36th St                  40.7             -74.0 YES   YES
## 6 4 Avenue 36th St                  40.7             -74.0 YES   YES
## # i 5 more variables: entrance_type <chr>, ada <lgl>, entry_logical <lgl>,
## #   route_number <chr>, route_name <chr>
```

2. How many distinct stations serve the A train?

```
# number of stations which serve A train
a_train_stations = Reformat_subway_df |>
  filter(route_name == "A") |>
  distinct(station_name, line) |>
  nrow()
```

There are 60 stations serve the A train.

3. Of the stations that serve the A train, how many are ADA compliant?

```
# number of ADA compliant stations which serve A train
a_train_stations_ADA = Reformat_subway_df |>
  filter(route_name == "A", ada == TRUE) |>
  distinct(station_name, line) |>
  nrow()
```

There are 17 ADA compliant stations serve the A train.

II. Problem 2

2.1 Load the Mr. Trash Wheel Sheet

- Import the **Mr. Trash Wheel** sheet, while omitting non-data entries;
- Omit rows that do not include dumpster-specific data;
- Round the number of `sports_balls`.

```
# Load the Trash Wheel xlsx
trash_wheel_path = "./202409 Trash Wheel Collection Data.xlsx"
MTW =
  readxl::read_excel(trash_wheel_path, sheet = "Mr. Trash Wheel",
                     skip = 1, na = c("NA", ".", "")) |>
  janitor::clean_names() |>
  select(dumpster:homes_powered) |>
  drop_na(dumpster) |>
  mutate(
    sports_balls = as.integer(round(sports_balls))
    )
```

Similarly, import the **Professor Trash Wheel** and **Gwynnda Trash Wheel** sheets.

```r
# Load the PTW and GTW sheet
PTW =
  readxl::read_excel(trash_wheel_path, sheet = "Professor Trash Wheel",
                     skip = 1, na = c("NA", ".", "")) |>
  janitor::clean_names() |>
  select(dumpster:homes_powered) |>
  drop_na(dumpster, month)

GTW =
  readxl::read_excel(trash_wheel_path, sheet = "Gwynnda Trash Wheel",
                     skip = 1, na = c("NA", ".", "")) |>
  janitor::clean_names() |>
  select(dumpster:homes_powered) |>
  drop_na(dumpster, month)
```

2.2 Combine PTW and GTW with MTW

```r
MTW = MTW |>
  mutate(category = "Mr._Trash_Wheel") |>
  mutate(year = as.character(year))
PTW = PTW |>
  mutate(category = "Professor_Trash_Wheel") |>
  mutate(year = as.character(year))
GTW = GTW |>
  mutate(category = "Gwynnda_Trash_Wheel") |>
  mutate(year = as.character(year))

trash_wheel_df =
  bind_rows(MTW, PTW, GTW) |>
  relocate(category)
```

- The number of observations in the resulting datasets is as follows, where `trash_wheel_df` represents the final merged dataset:

	Mr. Trash Wheel	Professor Trash Wheel	Gwynnda Trash Wheel	trash_wheel_df
observation	651	118	263	1032
variable	15	14	13	15

Compared to **Mr. Trash Wheel**, **Professor Trash Wheel** and **Gwynnda Trash Wheel** are missing `sports_balls` and `glass_bottles` & `sports_balls`, respectively. These missing values result in a large number of "NA" in the **trash_wheel_df**. But they are still valid data that represents their own category, so **trash_wheel_df** is tidy. In addition, only `sports_balls` and `homes_powered` trash exist as multiple decimal places, while the values of other garbage types are integers.

```
# Total weight of trash collected by Professor Trash Wheel
PTW_weight = PTW |>
  summarise(total_weight = sum(weight_tons, na.rm = TRUE))

# Total number of cigarette butts collected by Gwynnda in June 2022
GTW_cigarette = GTW |>
  filter(month(date) == 6,
         year(date) == 2022) |>
  summarise(total_cigarette_butts = sum(cigarette_butts, na.rm = TRUE))
```

- The total weight of trash collected by Professor Trash Wheel is 246.74.
- The total number of cigarette butts collected by Gwynnda in June of 2022 is 18120.

III. Problem 3

3.1 Load all 4 csv

bakers_df:

- Load bakers.csv;
- Split the player's first name from `bakers_name` as `baker` so that it can be used as a key for later dataset merging;
- Ensure there are no duplicate bakers;

```
bakers_df = read.csv("./gbb_datasets/bakers.csv",
                     na = c("NA", ".", "")) |>
  janitor::clean_names() |>
  mutate(baker = sub(" .*", " ", baker_name)) |>
  mutate(baker = iconv(baker, from = "latin1", to = "UTF-8", sub = "")) |>
  mutate(baker = trimws(baker)) |> # remove " "
  distinct() |>
  arrange(baker) |>
  relocate(baker)
```

bakes_df:

- Load bakes.csv;
- Ensure there are no duplicate bakers;
- Noticed that the name format of the player `"Jo"` is inconsistent with that of other players, since the double quotation marks are added. Modify it with `casematch`.

```
bakes_df = read.csv("./gbb_datasets/bakes.csv",
                    na = c("NA", ".", "")) |>
  janitor::clean_names() |>
  distinct() |>
  mutate(baker = case_match(
    baker,
```

```
    '"Jo"' ~ "Jo",
    .default = baker)
    ) |> #keep other values unchanged
  mutate(baker = iconv(baker, from = "latin1", to = "UTF-8", sub = "")) |>
  mutate(baker = trimws(baker)) |>  # remove " "
  arrange(baker) |>
  relocate(baker)
```

results_df:

- Load results.csv;

```
results_df = read.csv("./gbb_datasets/results.csv", skip = 2,
                      na = c("NA", ".", "")) |>
  mutate(baker = iconv(baker, from = "latin1", to = "UTF-8", sub = "")) |>
  mutate(baker = trimws(baker)) |>  # remove " "
  janitor::clean_names()
```

3.2 Check the Completeness

- Identify if any baker in `results_df` is missing from the `bakers_df`.

```
#anti_join(x, y, by = "key")  x have while y donot have
missing_bakers = anti_join(results_df, bakers_df, by = "baker")

missing_bakers
```

```
##   series episode  baker technical      result
## 1      2       1 Joanne        11          IN
## 2      2       2 Joanne        10          IN
## 3      2       3 Joanne         1          IN
## 4      2       4 Joanne         8          IN
## 5      2       5 Joanne         6          IN
## 6      2       6 Joanne         1 STAR BAKER
## 7      2       7 Joanne         3          IN
## 8      2       8 Joanne         1      WINNER
```

The results show that Joanne's series 2 episodes 1-6 is present in the `results_df`, but not in the `bakers_df`.

- Identify if any baker's bake in `results_df` is missing from the `bakes_df`.

```
missing_bakes = anti_join(results_df, bakes_df, by = c("baker", "episode"))
summary(missing_bakes)
```

```
##      series          episode          baker             technical
##  Min.   : 1.000   Min.   : 1.000   Length:554         Min.   : 1.000
##  1st Qu.: 4.000   1st Qu.: 4.000   Class :character   1st Qu.: 3.000
##  Median : 7.000   Median : 7.000   Mode  :character   Median : 5.000
##  Mean   : 6.699   Mean   : 6.377                      Mean   : 5.075
##  3rd Qu.: 9.000   3rd Qu.: 9.000                      3rd Qu.: 7.000
```

```
##  Max.   :10.000   Max.   :10.000                      Max.   :13.000
##                                                       NA's   :408
##     result
##  Length:554
##  Class :character
##  Mode  :character
##
##
##
##
```

The results shows that 84 bakers' bakes are missing from the `bakes_df`.

3.3 Merge Datasets

```
# Merge all 3 datasets
combined_df =
  results_df |>
  left_join(bakers_df, by = c("baker", "series")) |>
  left_join(bakes_df, by = c("baker", "series", "episode"))

# Reorganize the variables to be meaningful
final_df =
  combined_df |>
  select(series, episode, baker_name, technical, result, signature_bake, show_stopper,
         baker_age, hometown, baker_occupation) |>
  arrange(series, episode,technical)

# export the final_df as csv
write_csv(final_df, "./gbb_datasets/Great British Bake Off.csv")
```

The final dataset `final_df` has 1136 observations and 10 variables.

In line with the preferences of viewers, this article places The Show's `series` and `eposide` at the top of the dataset, followed by `Bakers' Name` and their `technical`. The following are personal characteristics and background information about each baker, including `signature bake`, `show stopper bake`, `age`, `hometown`, and `occupation status`.

3.4 Star Bakers

Filter results for Seasons 5 to 10 and select Star Baker.

```
star_baker_df = results_df |>
  filter(series >= 5 & series <= 10, result %in% c("STAR BAKER", "WINNER")) |>
  select(series, episode, baker, result)
```

Create a table to show star bakers in Season 5 to 10, organizing by series and episode.

```
star_baker_df |>
  arrange(series, episode) |>
  kable(caption = "Star Baker and Winners for Seasons 5 to 10",
```

```
        booktabs = TRUE) |>
  kable_styling() |>
  row_spec(which(star_baker_df$series %in% c(5, 7, 9)), background = "lightgray")
```

Table 2: Star Baker and Winners for Seasons 5 to 10

series	episode	baker	result
5	1	Nancy	STAR BAKER
5	2	Richard	STAR BAKER
5	3	Luis	STAR BAKER
5	4	Richard	STAR BAKER
5	5	Kate	STAR BAKER
5	6	Chetna	STAR BAKER
5	7	Richard	STAR BAKER
5	8	Richard	STAR BAKER
5	9	Richard	STAR BAKER
5	10	Nancy	WINNER
6	1	Marie	STAR BAKER
6	2	Ian	STAR BAKER
6	3	Ian	STAR BAKER
6	4	Ian	STAR BAKER
6	5	Nadiya	STAR BAKER
6	6	Mat	STAR BAKER
6	7	Tamal	STAR BAKER
6	8	Nadiya	STAR BAKER
6	9	Nadiya	STAR BAKER
6	10	Nadiya	WINNER
7	1	Jane	STAR BAKER
7	2	Candice	STAR BAKER
7	3	Tom	STAR BAKER
7	4	Benjamina	STAR BAKER
7	5	Candice	STAR BAKER
7	6	Tom	STAR BAKER
7	7	Andrew	STAR BAKER
7	8	Candice	STAR BAKER
7	9	Andrew	STAR BAKER
7	10	Candice	WINNER
8	1	Steven	STAR BAKER
8	2	Steven	STAR BAKER
8	3	Julia	STAR BAKER
8	4	Kate	STAR BAKER
8	5	Sophie	STAR BAKER
8	6	Liam	STAR BAKER
8	7	Steven	STAR BAKER
8	8	Stacey	STAR BAKER
8	9	Sophie	STAR BAKER
8	10	Sophie	WINNER
9	1	Manon	STAR BAKER
9	2	Rahul	STAR BAKER

9	3	Rahul	STAR BAKER
9	4	Dan	STAR BAKER
9	5	Kim-Joy	STAR BAKER
9	6	Briony	STAR BAKER
9	7	Kim-Joy	STAR BAKER
9	8	Ruby	STAR BAKER
9	9	Ruby	STAR BAKER
9	10	Rahul	WINNER
10	1	Michelle	STAR BAKER
10	2	Alice	STAR BAKER
10	3	Michael	STAR BAKER
10	4	Steph	STAR BAKER
10	5	Steph	STAR BAKER
10	6	Steph	STAR BAKER
10	7	Henry	STAR BAKER
10	8	Steph	STAR BAKER
10	9	Alice	STAR BAKER
10	10	David	WINNER

```
baker_frequency = star_baker_df |>
  count(baker, sort = TRUE)  # sort = TRUE means descending
head(baker_frequency)
```

```
##     baker n
## 1 Richard 5
## 2 Candice 4
## 3  Nadiya 4
## 4   Steph 4
## 5     Ian 3
## 6   Rahul 3
```

- **Predictable Overall Winners:** *Richard Burr* won STAR BAKER 5 times, which is the most of all bakers. *Candice Brown*, *Nadiya Hussain* and *Steph Blackwell* won 4 times. Additionally, *Richard Burr* from series 5, *Ian Cumming* and *Nadiya Hussain* from series 6, *Steph Blackwell* from series 10 all consistently achieved STAR BAKER during their own series. However, only *Nadiya Hussain* became the final WINNER in series 6 episode 10. To summarize, **Nadiya Hussain** is the most predictable overall winners.
- **Surprises:** It is surprising that **David Atherton** from series 10 was crowned STAR BAKER in episode 10, even though he did not won anyone before.

3.5 viewers_df

Start by importing viewers.csv.

```
viewers_df = read.csv("./gbb_datasets/viewers.csv",
                      na = c("NA", ".", "")) |>
  janitor::clean_names()

head(viewers_df, 10)
```

```
##    episode series_1 series_2 series_3 series_4 series_5 series_6 series_7
## 1        1     2.24     3.10     3.85     6.60    8.510    11.62    13.58
## 2        2     3.00     3.53     4.60     6.65    8.790    11.59    13.45
## 3        3     3.00     3.82     4.53     7.17    9.280    12.01    13.01
## 4        4     2.60     3.60     4.71     6.82   10.250    12.36    13.29
## 5        5     3.03     3.83     4.61     6.95    9.950    12.39    13.12
## 6        6     2.75     4.25     4.82     7.32   10.130    12.00    13.13
## 7        7       NA     4.42     5.10     7.76   10.280    12.35    13.45
## 8        8       NA     5.06     5.35     7.41    9.023    11.09    13.26
## 9        9       NA       NA     5.70     7.41   10.670    12.65    13.44
## 10      10       NA       NA     6.74     9.45   13.510    15.05    15.90
##    series_8 series_9 series_10
## 1      9.46     9.55      9.62
## 2      9.23     9.31      9.38
## 3      8.68     8.91      8.94
## 4      8.55     8.88      8.96
## 5      8.61     8.67      9.26
## 6      8.61     8.91      8.70
## 7      9.01     9.22      8.98
## 8      8.95     9.69      9.19
## 9      9.03     9.50      9.34
## 10    10.04    10.34     10.05
```

```r
average_series_1 = viewers_df |>
  summarize(average = mean(series_1, na.rm = TRUE))
average_series_5 = viewers_df |>
  summarize(average = mean(series_5, na.rm = TRUE))
```

The average viewership in Season 1 is 2.77. In Season 5 is 10.0393.