Data 607 - Assignment 4 - Tidying and Transforming Data

William Jasmine

2022-09-27

Introduction

The goal of this assignment is to take some messy (untidy) airport arrival data, clean it, and then analyze it. The hope is that doing so will elucidate a number of insights regarding the arrival patterns of two different airlines.

Import Data

The data is stored in a csv file here, and is imported as a R data frame in the chunk below:

```
raw_data <- getURL('https://raw.githubusercontent.com/williamzjasmine/CUNY_SPS_DS/master/DATA_607/Homew

# File was tab delimited, hence the sep = "\t" option
df <- data.frame(read.csv(text=raw_data, sep = "\t"))
df
```

```
##         X     X.1 Los.Angeles Phoenix San.Diego San.Francisco Seattle
## 1  ALASKA on time         497     221       212           503    1841
## 2         delayed          62      12        20           102     305
## 3                          NA      NA        NA            NA      NA
## 4 AM WEST on time         694    4840       383           320     201
## 5         delayed         117     415        65           129      61
```

The output above reveals that the data in its current format has a number of issues to be addressed:

1. The first two columns were unnamed, and thus the import function gave them the generic `X` and `X.1` labels.
2. The third row only contains missing data (both `NA` values and empty strings).
3. The airline name is missing in two of the cells where it looks like it should be populated (rows 2 and 5, column `X`).
4. The data is in a wide format:

- The names of the cities should not be their own columns but rather condensed into a single column.
- Once the categorical variables all have their own column, the values in the cells can also be condensed into a single column.

5. Once the city names become values in a single column, the `.` characters in their names need to be replaced by a space character. Though the `.` was not present in the city names in the original csv file, they were added during the import to prevent there being spaces in the dataframe column names.

The following section works to fix each of the above issues in order.

Tidy Data

The cell below reaplces the column names `X` and `X.1` with what they should actually be: `airline` and `arrival_status`:

```
colnames(df)[1] <- "airline"
colnames(df)[2] <- "arrival_status"
df
```

```
##   airline arrival_status Los.Angeles Phoenix San.Diego San.Francisco Seattle
## 1  ALASKA        on time         497     221       212           503    1841
## 2                delayed          62      12        20           102     305
## 3                                 NA      NA        NA            NA      NA
## 4 AM WEST        on time         694    4840       383           320     201
## 5                delayed         117     415        65           129      61
```

As shown in the output above, these first two columns now have appropriate names.

The code chunk below uses the `drop_na()` function to remove the third row entirely, since it contains no useful information:

```
df <- drop_na(df)
df
```

```
##   airline arrival_status Los.Angeles Phoenix San.Diego San.Francisco Seattle
## 1  ALASKA        on time         497     221       212           503    1841
## 2                delayed          62      12        20           102     305
## 3 AM WEST        on time         694    4840       383           320     201
## 4                delayed         117     415        65           129      61
```

As is shown in the output above, the once third row containing only empty and `NA` values is no longer present in the dataframe.

The code below fills in the empty values in the `airline` column using the `fill()` function. This function fills a `NA` cell with the value of the column immediately above it, which in this case exactly what this dataframe needs. However, to use the function those two empty values are first converted to `NA`.

```
df[df==""]<-NA
df <- fill(df, airline)
df
```

```
##   airline arrival_status Los.Angeles Phoenix San.Diego San.Francisco Seattle
## 1  ALASKA        on time         497     221       212           503    1841
## 2  ALASKA        delayed          62      12        20           102     305
## 3 AM WEST        on time         694    4840       383           320     201
## 4 AM WEST        delayed         117     415        65           129      61
```

As shown in the output above, the missing `airline` values are now included in the dataframe.

Now that there is no missing data, the next step is to transform the dataframe into a long format. This can be easily done using the `pivot_longer()` function, and specifying in the `names_to` column that the city name columns are to be melted into a single field called `city`. The values in the cells are then also melted into a single field called `frequency`.

```
df <- pivot_longer(df, cols = !c(airline, arrival_status),
                   names_to = 'city', values_to = 'frequency')
head(df)
```

```
## # A tibble: 6 x 4
##   airline arrival_status city          frequency
```

```
##   <chr>   <chr>          <chr>             <int>
## 1 ALASKA  on time        Los.Angeles         497
## 2 ALASKA  on time        Phoenix             221
## 3 ALASKA  on time        San.Diego           212
## 4 ALASKA  on time        San.Francisco       503
## 5 ALASKA  on time        Seattle            1841
## 6 ALASKA  delayed        Los.Angeles          62
```

The `head()` of the dataframe above shows that it has been successfully converted into a long format.

The last step is to replace the `.` characters in each of the city names with spaces. This is done easily with the `str_replace_all()` function:

```
df$city <- str_replace_all(df$city, "\\.", " ")
df
```

```
## # A tibble: 20 x 4
##    airline arrival_status city          frequency
##    <chr>   <chr>          <chr>             <int>
##  1 ALASKA  on time        Los Angeles         497
##  2 ALASKA  on time        Phoenix             221
##  3 ALASKA  on time        San Diego           212
##  4 ALASKA  on time        San Francisco       503
##  5 ALASKA  on time        Seattle            1841
##  6 ALASKA  delayed        Los Angeles          62
##  7 ALASKA  delayed        Phoenix              12
##  8 ALASKA  delayed        San Diego            20
##  9 ALASKA  delayed        San Francisco       102
## 10 ALASKA  delayed        Seattle             305
## 11 AM WEST on time        Los Angeles         694
## 12 AM WEST on time        Phoenix            4840
## 13 AM WEST on time        San Diego           383
## 14 AM WEST on time        San Francisco       320
## 15 AM WEST on time        Seattle             201
## 16 AM WEST delayed        Los Angeles         117
## 17 AM WEST delayed        Phoenix             415
## 18 AM WEST delayed        San Diego            65
## 19 AM WEST delayed        San Francisco       129
## 20 AM WEST delayed        Seattle              61
```

And that's it! The data is now in the desired form: no missing data, and long as opposed to wide.

Analyze Data

Now that the data has been cleaned, it can be analyzed. The following code chunk uses a combination of `dplyr` functions to determine the percentage of flights that had late or timely arrivals for each airline:

```
airline_df <-
  df %>%
    group_by(airline) %>%
      summarise(
        num_flights = sum(frequency),
        num_delays = sum(ifelse(arrival_status=='delayed', frequency, 0))
      )

airline_df <-
```

```
  airline_df %>%
    mutate(
        num_on_time = num_flights - num_delays,
        delay_rate = num_delays / num_flights,
        on_time_rate = (num_flights - num_delays) / num_flights
    ) %>%
      arrange(delay_rate)
airline_df
```

```
## # A tibble: 2 x 6
##   airline num_flights num_delays num_on_time delay_rate on_time_rate
##   <chr>         <int>      <dbl>       <dbl>      <dbl>        <dbl>
## 1 AM WEST        7225        787        6438      0.109        0.891
## 2 ALASKA         3775        501        3274      0.133        0.867
```

It looks like that while AM WEST (American West Airlines) has more arrival delays in total, they actually had a slightly better `delay_rate` than ALASKA (Alaska Airlines) by a little over two percentage points: in other words, flights from AM WEST arrived on time for a higher percentage of their flights.

The following cell performs the same analysis, except now aggregates the information on a city level:

```
city_df <-
  df %>%
    group_by(city) %>%
      summarise(
        num_flights = sum(frequency),
        num_delays = sum(ifelse(arrival_status=='delayed', frequency, 0))
      )

city_df <-
  city_df %>%
    mutate(
        num_on_time = num_flights - num_delays,
        delay_rate = num_delays / num_flights,
        on_time_rate = (num_flights - num_delays) / num_flights
    ) %>%
      arrange(delay_rate)

city_df
```

```
## # A tibble: 5 x 6
##   city          num_flights num_delays num_on_time delay_rate on_time_rate
##   <chr>               <int>      <dbl>       <dbl>      <dbl>        <dbl>
## 1 Phoenix              5488        427        5061     0.0778        0.922
## 2 San Diego             680         85         595     0.125         0.875
## 3 Los Angeles          1370        179        1191     0.131         0.869
## 4 Seattle              2408        366        2042     0.152         0.848
## 5 San Francisco        1054        231         823     0.219         0.781
```

Looking at the `delay_rates` for individual cities, its clear theres actually a pretty big range. Phoenix was the easiest city to get to for both airlines, with about 92% of their flights arriving on time. However, San Francisco seemed to pose some trouble, as only 78% of flights made it there without delay. The following cell zooms in on both of these cities's flight info:

```
df %>%
  filter(city == 'San Francisco' | city == 'Phoenix') %>%
```

```
    group_by(city, airline) %>%
      summarise(
        num_flights = sum(frequency),
        num_delays = sum(ifelse(arrival_status=='delayed', frequency, 0))
      ) %>%
        mutate(
          num_on_time = num_flights - num_delays,
          delay_rate = num_delays / num_flights,
          on_time_rate = (num_flights - num_delays) / num_flights
        ) %>%
          arrange(city, airline)
```

```
## `summarise()` has grouped output by 'city'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 4 x 7
## # Groups:   city [2]
##   city          airline num_flights num_delays num_on_time delay_rate on_time_~1
##   <chr>         <chr>         <int>      <dbl>       <dbl>      <dbl>      <dbl>
## 1 Phoenix       ALASKA          233         12         221     0.0515      0.948
## 2 Phoenix       AM WEST        5255        415        4840     0.0790      0.921
## 3 San Francisco ALASKA          605        102         503     0.169       0.831
## 4 San Francisco AM WEST         449        129         320     0.287       0.713
## # ... with abbreviated variable name 1: on_time_rate
```

The information above shows that while both airlines had pretty close `on_time_rate`s for Phoenix, AM WEST seems more to blame for San Franciso's overall `delay_rate`: only 71% of AM WEST's flights to San Francisco arrived on time, compared to 83% of flights flown by ALASKA. Given that San Francisco is part of the American West, you might assume that AM WEST could do better, but maybe that's why they went out of business...

Output Data

The final step is to output the data so that anyone can build upon the analysis done here. This is done in the chunk below:

```
write.csv(df,"clean_airport_data.csv")
```