Data607-Week_Four-Assignment

Shri Tripathi

09/28/2024

Assignment – Tidying and Transforming Data

The chart above describes arrival delays for two airlines across five destinations. Your task is to:

1. Create a .CSV file (or optionally, a MySQL database!) that includes all of the information above. You're encouraged to use a "wide" structure similar to how the information appears above, so that you can practice tidying and transformations as described below.
2. Read the information from your .CSV file into R, and use tidyr and dplyr as needed to tidy and transform your data.
3. Perform analysis to compare the arrival delays for the two airlines.
4. Your code should be in an R Markdown file, posted to rpubs.com, and should include narrative descriptions of your data cleanup work, analysis, and conclusions. Please include in your homework submission:
    - The URL to the .Rmd file in your GitHub repository.
    - The URL for your rpubs.com web page.

Data Cleanup:

First, I loaded the necessary packages for the analysis and imported an 'airlines.csv' file, which I had previously saved in the working directory and also uploaded to GitHub for easy access. After loading the dataset, I reviewed its structure. To organize the data by airline and flight status, I first replaced empty strings with NA values. Using the `tidyr` package, I filled missing airline names downward with the `fill` function. Then, I renamed columns for clarity using `dplyr`'s `rename()` function, and finally, I removed rows with missing statuses by applying the `filter()` function.

```
# Load required libraries
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)

# Read data from GitHub repository
airlines_dat <- read.csv(url('https://raw.githubusercontent.com/Shriyanshh/Data607-Week_Four-Assignment/
# Display the data
airlines_dat
```

```
##         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
```

```
# Replace empty strings in 'X' column with NA
airlines_dat$X[airlines_dat$X==""] <- NA

# Clean and organize the data
clean_data <- airlines_dat %>%
  fill(X, .direction='down') %>%  # Fill missing airline names downwards
  rename(Airline = X, Status = X.1) %>%  # Rename columns for clarity
  filter(Status != "")  # Remove rows where Status is empty

# Display the cleaned data
clean_data
```

```
##   Airline  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
```

To streamline the dataset, I transformed the multiple city columns into a single column. Using the `tidyr` package's `gather` function, I converted the dataset from wide to long format, creating a new 'city' column to consolidate city names and a 'count' column to aggregate the counts. I excluded the 'Airlines' and 'Status' columns from this transformation to maintain their original format. This method efficiently condensed the city data into a more manageable form.

```
# Combine city columns into a single column while preserving airline and status data
clean_data = clean_data %>%
  gather(key = "City", value = "Count", -Airline, -Status)  # Converts multiple city columns into two c

clean_data
```

```
##    Airline  Status          City Count
## 1   ALASKA on time   Los.Angeles   497
## 2   ALASKA delayed   Los.Angeles    62
```

```
## 3  AM WEST on time   Los.Angeles   694
## 4  AM WEST delayed   Los.Angeles   117
## 5   ALASKA on time       Phoenix   221
## 6   ALASKA delayed       Phoenix    12
## 7  AM WEST on time       Phoenix  4840
## 8  AM WEST delayed       Phoenix   415
## 9   ALASKA on time     San.Diego   212
## 10  ALASKA delayed     San.Diego    20
## 11 AM WEST on time     San.Diego   383
## 12 AM WEST delayed     San.Diego    65
## 13  ALASKA on time San.Francisco   503
## 14  ALASKA delayed San.Francisco   102
## 15 AM WEST on time San.Francisco   320
## 16 AM WEST delayed San.Francisco   129
## 17  ALASKA on time       Seattle  1841
## 18  ALASKA delayed       Seattle   305
## 19 AM WEST on time       Seattle   201
## 20 AM WEST delayed       Seattle    61
```

The final step in refining the data involved removing any empty columns. This was achieved by applying the `filter` function to exclude columns that contained no data.

```
# Remove rows where 'Count' is NA to ensure data integrity
clean_data = clean_data %>%
  filter(!is.na(Count))

clean_data
```

```
##    Airline  Status          City Count
## 1   ALASKA on time   Los.Angeles   497
## 2   ALASKA delayed   Los.Angeles    62
## 3  AM WEST on time   Los.Angeles   694
## 4  AM WEST delayed   Los.Angeles   117
## 5   ALASKA on time       Phoenix   221
## 6   ALASKA delayed       Phoenix    12
## 7  AM WEST on time       Phoenix  4840
## 8  AM WEST delayed       Phoenix   415
## 9   ALASKA on time     San.Diego   212
## 10  ALASKA delayed     San.Diego    20
## 11 AM WEST on time     San.Diego   383
## 12 AM WEST delayed     San.Diego    65
## 13  ALASKA on time San.Francisco   503
## 14  ALASKA delayed San.Francisco   102
## 15 AM WEST on time San.Francisco   320
## 16 AM WEST delayed San.Francisco   129
## 17  ALASKA on time       Seattle  1841
## 18  ALASKA delayed       Seattle   305
## 19 AM WEST on time       Seattle   201
## 20 AM WEST delayed       Seattle    61
```

Analysis

For the assignment analysis on arrival delays between two airlines, I first filtered the cleaned dataset to include only entries where the Status was 'delayed'. Using `dplyr`, I then grouped the data first by 'Airline'

and then by 'City' to prepare for detailed comparative analysis. This structured approach allowed us to focus specifically on delay patterns across different cities for each airline.

```
# Filter for delayed flights and group data by Airline and City for focused analysis
summary_stats = clean_data %>%
  filter(Status == "delayed") %>%
  group_by(Airline, City)

summary_stats
```

```
## # A tibble: 10 x 4
## # Groups:   Airline, City [10]
##    Airline Status  City          Count
##    <chr>   <chr>   <chr>         <int>
##  1 ALASKA  delayed Los.Angeles      62
##  2 AM WEST delayed Los.Angeles     117
##  3 ALASKA  delayed Phoenix          12
##  4 AM WEST delayed Phoenix         415
##  5 ALASKA  delayed San.Diego        20
##  6 AM WEST delayed San.Diego        65
##  7 ALASKA  delayed San.Francisco   102
##  8 AM WEST delayed San.Francisco   129
##  9 ALASKA  delayed Seattle         305
## 10 AM WEST delayed Seattle          61
```

After analyzing the dataset, it was found that Alaska Airlines experienced fewer delays compared to AM West at four out of five airports. However, in Seattle, Alaska had more delays than AM West, suggesting AM West as a better choice for avoiding delays in that city. The most significant difference was observed at Phoenix Airport, where AM West had 415 delays compared to Alaska's 12. In contrast, the difference was least noticeable at San Francisco Airport with AM West and Alaska having 129 and 102 delays, respectively.

To calculate the proportion of delayed flights per city and airline, I first aggregated the total flight counts for each airline and city. Then, I retrieved the number of delayed flights for each grouping. By performing an inner join on these datasets, I obtained both the number of delays and total flights. Subsequently, I calculated the delay proportion and converted these figures to percentages. The resulting table displays the percentage of delayed flights, helping to identify which airline performs better in each city.

```
# Summarize total flights by airline and city
total_flights = clean_data %>%
  group_by(Airline, City) %>%
  summarize(Total_Count = sum(Count))
```

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

```
# Filter for delayed flights and summarize by airline and city
delayed_flights = clean_data %>%
  filter(Status == "delayed") %>%
  group_by(Airline, City) %>%
  summarize(Delayed_Count = sum(Count))
```

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

```r
# Join datasets to get total and delayed counts together
delayed_and_total = inner_join(total_flights, delayed_flights, by = c("Airline", "City"))

# Calculate the ratio of delayed flights
prop_delays = delayed_and_total %>%
  mutate(Delayed_Ratio = Delayed_Count / Total_Count)

# Arrange by city and select columns of interest
prop_delays = prop_delays %>%
  select(Airline, City, Delayed_Ratio) %>%
  arrange(City)

# Convert ratio to percent and round off
prop_delays = prop_delays %>%
  mutate(Delayed_Percent = round(Delayed_Ratio * 100, 1))

# Load knitr library for table creation
library(knitr)

# Create a table displaying the percentage of delayed flights by city and airline
prop_delays %>%
  select(Airline, City, Delayed_Percent) %>%
  kable(caption = "Percent of Delayed Flights by City and Airline")
```

Table 1: Percent of Delayed Flights by City and Airline

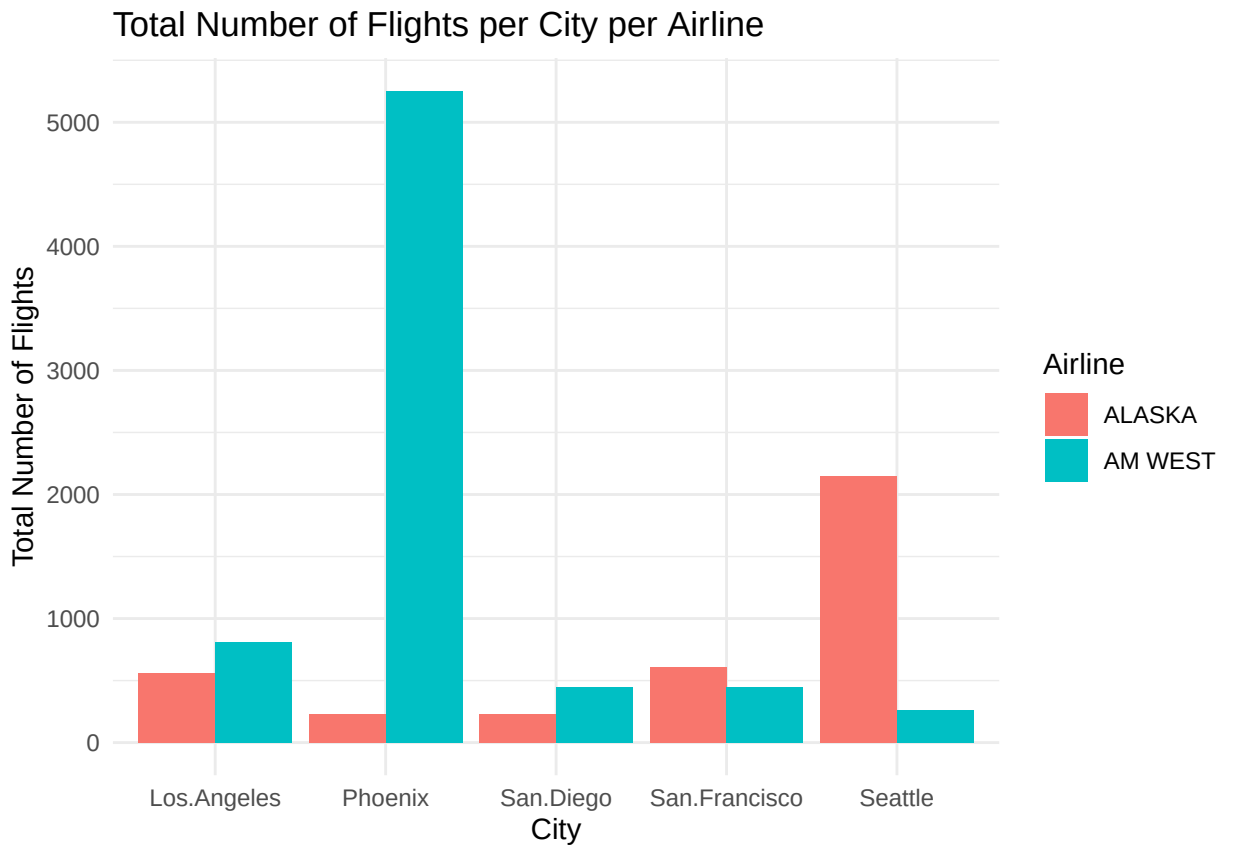
Airline	City	Delayed_Percent
ALASKA	Los.Angeles	11.1
AM WEST	Los.Angeles	14.4
ALASKA	Phoenix	5.2
AM WEST	Phoenix	7.9
ALASKA	San.Diego	8.6
AM WEST	San.Diego	14.5
ALASKA	San.Francisco	16.9
AM WEST	San.Francisco	28.7
ALASKA	Seattle	14.2
AM WEST	Seattle	23.3

When comparing the ratio of delayed flights to total flights for each airline in various cities, the percentages are relatively similar between ALASKA and AM WEST. For instance, in San Diego, 8% of ALASKA's flights are delayed compared to 14.5% for AM WEST. Similarly, in San Francisco, the delay rates are 16.9% for ALASKA and 28.7% for AM WEST. To visually represent this data, I used ggplot to create a graph displaying the total number of flights by airline for each city.

```r
# Load the ggplot2 library for creating visualizations
library(ggplot2)

# Create a bar chart with total flight counts by city and airline
ggplot(total_flights, aes(x=City, y=Total_Count, fill=Airline)) +
  geom_bar(stat="identity", position="dodge") +
  labs(title="Total Number of Flights per City per Airline",
       x="City",
```

```
      y="Total Number of Flights",
      fill="Airline") +
 theme_minimal()
```



The visualization clearly shows that Phoenix airport handles a significantly higher number of flights from AM WEST compared to ALASKA, which may contribute to their higher delay rates due to the increased volume of flights. Conversely, ALASKA operates more flights to Seattle than AM WEST, yet ALASKA still manages fewer delays there. This suggests that the volume of flights could be a factor in the frequency of delays, though other operational efficiencies likely also play a role.

Conclusion

In this assignment, I utilized the Tidyr and dplyr packages for their robust data cleaning and organizing capabilities, which streamlined handling complex or poorly formatted datasets. Starting with a wide-format CSV containing airline data, I transformed it into a format digestible by R. This cleaned dataset then allowed me to analyze and compare airline delays. By calculating the delay proportions and percentages per city and airline, and visualizing these in both tabular and graphical formats, I could effectively present the density of flights to various cities by each airline.