Data 607 - Week 5 Assignment

Adam Douglas

9/27/2018

Loading the Data

First we load the CSV file by using the `read_csv` function from the `readr` package. The CSV file was created by copying exactly what was presented in the assignment PDF (shown below). This includes missing fields, blank lines, and all.

```
rawdata <- read_csv("week5.csv",col_names=TRUE)

rawdata
```

```
## # A tibble: 5 x 7
##   X1      X2     `Los Angeles` Phoenix `San Diego` `San Francisco` Seattle
##   <chr>   <chr>          <int>   <int>       <int>           <int>   <int>
## 1 Alaska  on-ti~           491     221         212             503    1841
## 2 <NA>    delay~            62      12          20             102     503
## 3 <NA>    <NA>              NA      NA          NA              NA      NA
## 4 AM WEST on-ti~           694    4840         383             320     201
## 5 <NA>    delay~           117     415          65             129      61
```

Looking at what was read into R, we see a few different issues with our raw data:

1. Missing column names (which were assigned unintuitive names of X1 and X2)
2. Observations that are entirely blank
3. Missing values in the first column

We'll take each of these in turn in the next section and show how we can correct them.

Data Cleansing

Missing Column Names

First we tackel the two columns that are missing names. This is relatively easy to correct thanks to the `dplyr` package and the `rename` function:

```
rawdata <- rename(rawdata, airline = X1, status = X2)

rawdata
```

```
## # A tibble: 5 x 7
##   airline status `Los Angeles` Phoenix `San Diego` `San Francisco` Seattle
##   <chr>   <chr>          <int>   <int>       <int>           <int>   <int>
## 1 Alaska  on-ti~           491     221         212             503    1841
## 2 <NA>    delay~            62      12          20             102     503
## 3 <NA>    <NA>              NA      NA          NA              NA      NA
## 4 AM WEST on-ti~           694    4840         383             320     201
## 5 <NA>    delay~           117     415          65             129      61
```

Now every column has a meaningful name. Note that the city names are non-standard (they have spaces). We could correct them as well, but when we begin to tidy the data having them with their original names will be useful.

		Los Angeles	Phoenix	San Diego	San Francisco	Seattle
ALASKA	on time	497	221	212	503	1,841
	delayed	62	12	20	102	305
AM WEST	on time	694	4,840	383	320	201
	delayed	117	415	65	129	61

Figure 1: The original table

Blank Observations

We had a blank row in our source file, so it stands to reason that we'd have a blank row in our data after input. To remove this empty observation, we can use the `filter` function in `dplyr`. By looking for observations without a blank status column, we can keep only complete rows.

```
rawdata <- rawdata %>% filter(!is.na(status))

rawdata
```

```
## # A tibble: 4 x 7
##   airline status `Los Angeles` Phoenix `San Diego` `San Francisco` Seattle
##   <chr>   <chr>          <int>   <int>       <int>           <int>   <int>
## 1 Alaska  on-ti~           491     221         212             503    1841
## 2 <NA>    delay~            62      12          20             102     503
## 3 AM WEST on-ti~           694    4840         383             320     201
## 4 <NA>    delay~           117     415          65             129      61
```

Missing Airlines

Finally, because the table the data came from was formatted such that the airline wasn't labeled on each line, we have NAs in that column as well. These are also rectified quite easily by using the `fill` function, which is from the `tidyr` package.

```
rawdata <- fill(rawdata,airline)

rawdata
```

```
## # A tibble: 4 x 7
##   airline status `Los Angeles` Phoenix `San Diego` `San Francisco` Seattle
##   <chr>   <chr>          <int>   <int>       <int>           <int>   <int>
## 1 Alaska  on-ti~           491     221         212             503    1841
## 2 Alaska  delay~            62      12          20             102     503
## 3 AM WEST on-ti~           694    4840         383             320     201
```

```
## 4 AM WEST delay~           117     415          65             129      61
```

Now we have a more complete and clean data set. Our next step will be to tidy it.

Tidying Data

By putting our data into a "tidy" format, we make analysis within R easier to do.

The largest issue with our raw data is that each city is an observation, yet it is stored in separate columns, like variables. What we need to do is bring these columns down into rows.

We use the `gather` function from `tidyr` to do rearrange our data. We pass it the names of the columns we need to gather into observations (here, the city columns), the "key" or name of the new column with the gathered values, and a name for the new value column (called "flights" here):

```
tidydata <- rawdata %>% gather(`Los Angeles`,Phoenix,`San Diego`,`San Francisco`,Seattle, key="city", va

tidydata
```

```
## # A tibble: 20 x 4
##    airline status  city          flights
##    <chr>   <chr>   <chr>           <int>
##  1 Alaska  on-time Los Angeles       491
##  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           503
## 19 AM WEST on-time Seattle           201
## 20 AM WEST delayed Seattle            61
```

Now, each city is an observation as it ought to be.

We also have the reverse problem with variables listed as observations, namely the status column. We need to move these into columns by using the `tidyr` package and the `spread` function.

```
# Since the "on-time" value gives us a non-standard column name, we change it with rename.

tidydata <- spread(tidydata,key=status, value=flights) %>% rename(on_time = `on-time`)

tidydata
```

```
## # A tibble: 10 x 4
##    airline city          delayed on_time
##    <chr>   <chr>           <int>   <int>
```

```
##  1 Alaska  Los Angeles        62     491
##  2 Alaska  Phoenix            12     221
##  3 Alaska  San Diego          20     212
##  4 Alaska  San Francisco     102     503
##  5 Alaska  Seattle           503    1841
##  6 AM WEST Los Angeles       117     694
##  7 AM WEST Phoenix           415    4840
##  8 AM WEST San Diego          65     383
##  9 AM WEST San Francisco     129     320
## 10 AM WEST Seattle            61     201
```

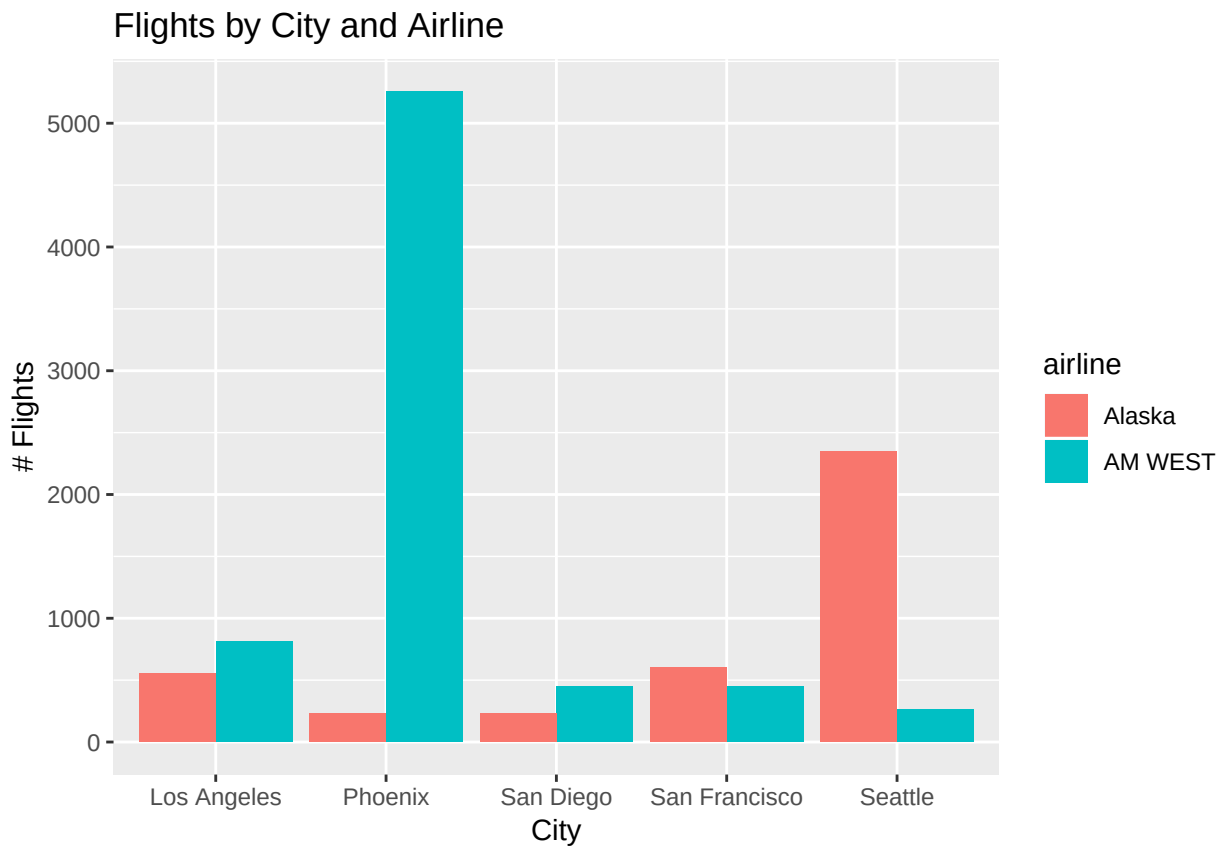
Now the two variables (delayed flights, and on-time flights) are in their proper place in the columns. OUr data set is now tidy.

Analysis

Now that our data is in a tidy format, our analysis can proceed.

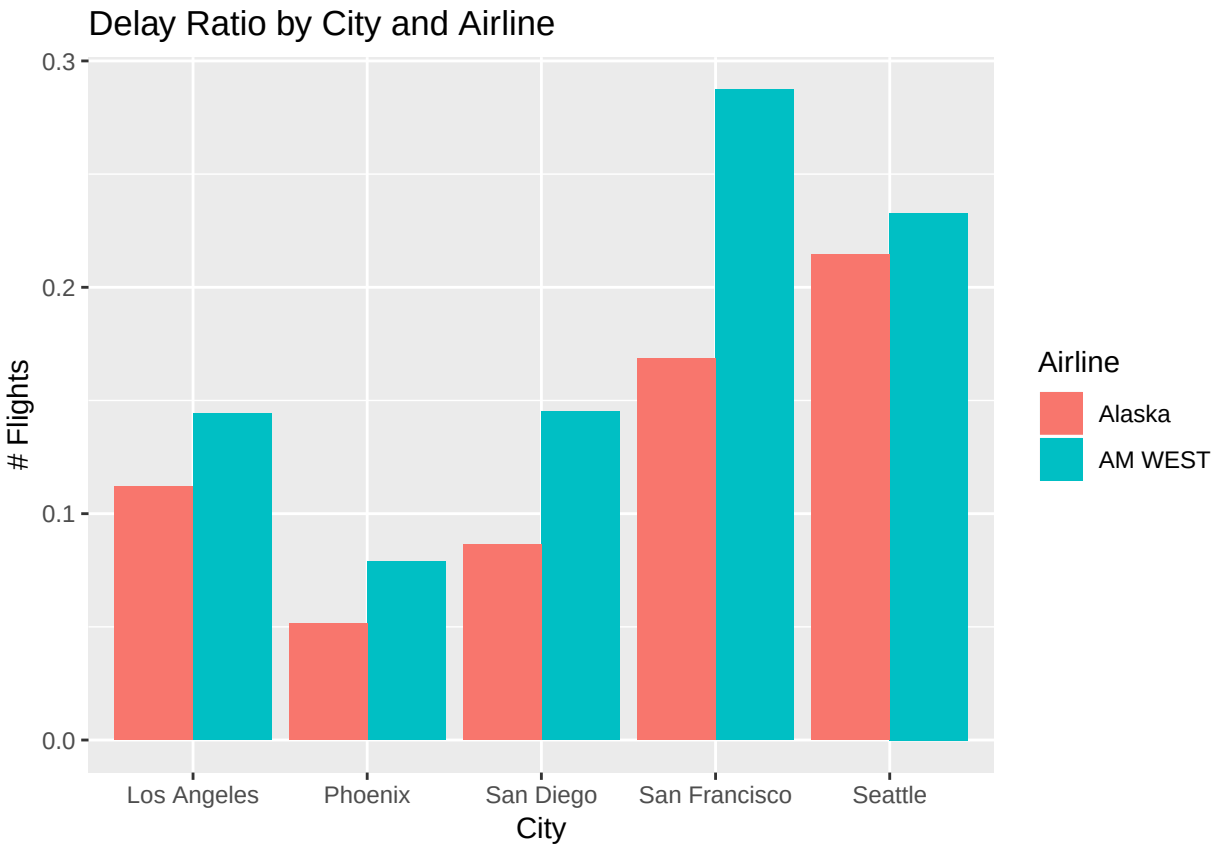
First, we'll look at each airline's service into each airport:

```
# Total flights by city and airline
totalByCity <- tidydata %>%
  group_by(airline, city) %>%
  summarize(total_flights = delayed + on_time)

# Plot
ggplot(totalByCity, aes(x=city, y=total_flights, fill=airline)) +
  geom_col(position="dodge") + ggtitle("Flights by City and Airline") +
  xlab("City") + ylab("# Flights")
```



As expected, each airline does not have the same number of flights coming into the same city, so we cannot compare the delays with raw counts. Instead we need a ratio for each.

```
# Delay ratio by city and airline
delayByCity <- tidydata %>%
  mutate(totalFlights = delayed + on_time, delayRatio = delayed/totalFlights) %>%
  group_by(airline, city) %>%
  summarize(delay = mean(delayRatio))

# Plot
ggplot(delayByCity, aes(x=city, y=delay, fill=airline)) +
  geom_col(position="dodge") + ggtitle("Delay Ratio by City and Airline") +
  xlab("City") + ylab("# Flights") + labs(fill="Airline")
```



Looking at the graph, we can quickly see that AM WEST has a higher ratio of delayed flights to total flights in every city they fly to, when compared to Alaska Airlines. In some cases, the difference is slight (Seattle) while in others it is significant (San Francisco).