Homework 2

Kitu Komya

October 21, 2016

1. Reading and cleaning data

1.1 Reading in large data files

1.

On average, my computer took 2.987468 seconds to load flights.csv using `read.csv()`. On average, my computer took 0.6371236 seconds to load flights.csv using `read_csv()`.

Wow! This is quite a huge difference in speed. `read_csv()` read in the data faster because it only looked at the first 1,000 observations to determine the class of a variable. This saves a lot of time in comparison to `read.csv()` reading in all 336,776 observations which takes an emormous amount of time to process such large memory space.

2.

Viewing the two data sets (in a separate R script), I would imagine that the class of `origin` would be a `string`, while the class of `time_hour` be `POSIX`, a time related class introduced in class.

```r
# I am reading in the data
flights_base <- read.csv(file = "flights.csv")
flights_readr <- read_csv(file = "flights.csv")
```

```
## Parsed with column specification:
## cols(
##   year = col_integer(),
##   month = col_integer(),
##   day = col_integer(),
##   dep_time = col_integer(),
##   sched_dep_time = col_integer(),
##   dep_delay = col_integer(),
##   arr_time = col_integer(),
##   sched_arr_time = col_integer(),
##   arr_delay = col_integer(),
##   carrier = col_character(),
##   flight = col_integer(),
##   tailnum = col_character(),
##   origin = col_character(),
##   dest = col_character(),
##   air_time = col_integer(),
##   hour = col_integer(),
##   minute = col_integer(),
##   time_hour = col_datetime(format = "")
## )
```

```r
# I am determining the classes
class(flights_base$origin)
```

```
## [1] "factor"
```

```r
class(flights_base$time_hour)
```

```
## [1] "factor"
```

```r
class(flights_readr$origin)
```

```
## [1] "character"
```

```r
class(flights_readr$time_hour)
```

```
## [1] "POSIXct" "POSIXt"
```

The class of `origin` and `time_hour` in `flights_base`, respectively, are `factor` and `factor`.

The class of `origin` and `time_hour` in `flights_readr`, respectively, are `character` and `POSIXct/POSIXt`

It is interesting to note that `flights_base` assigns any integer related variable with factor, which is kind of like a catch-all for all numerical vectors, even though origin is a categorical variable. I expected it to be precise in assigning a class, but apparently there is no need to do that.

However, `flights_readr` has assignment similar to what I was expecting. Instead of `strings`, however, it has assigned the origin to the class `character`, which is interesting because the values in that variable are all three-lettered characters, which I assumed would mean a string since it contains multiple characters. But `flights_readr` must store those three characters as one character.

1.2 Cleaning data with `base` methods

1.

```r
# I have read in the planes data in R Script, since R Markdown won't allow me to access it in html
planes <- read.csv(file = "planes.csv")
```

2.

```r
# I am changing the variable names from uppercase to lowercase
names(planes) = c("tailnum", "year", "type", "manufacturer", "model", "engines", "seats", "speed", "eng
```

3.

```r
# I have replaced the "-" with NA
planes[planes == "-"] <- NA
```

4.

```r
# This outputs the number of levels in tailnum
nlevels(planes$tailnum)
```

```
## [1] 3322
```

As seen, the number of levels in tailnum, 3322, is the exact same as the number of observations in planes. Thus, labeling it as a categorical variable is wrong because the levels are unique and they are not grouped by any categorical variable.

5. This is just a note that requires no response from me.

6.

```r
# This changes the class of the tailnum to character
planes$tailnum <- as.character(planes$tailnum)
```

7.

```r
# I have subsetted the data frame to only include the data with values in speed
plane_speed <- planes[!is.na(planes$speed), 8]
```

8.

```r
# I have converted plane_speed from a factor to a numeric
as.numeric(plane_speed)
```

```
##  [1] 13 13  8  9  2 11  3  5  7  8  6 14 12 10  4 12  2 12 12 12 12 12 12
```

I don't quite understand how R converted the values to numeric. The original values from factors included the actual values from the data set. When converted to numeric, the number has changed dramatically, and I don't quite know the conversion. For instance, the factor values/actual values had numbers such as 107, but once converted to numeric, all the numbers are under 20. Strange!

9.

```r
# I have converted plane_speed from a factor to a character
plane_speed <- as.character(plane_speed)
plane_speed
```

```
##  [1] "90"  "90"  "162" "167" "105" "232" "107" "112" "127" "162" "126"
## [12] "95"  "432" "202" "108" "432" "105" "432" "432" "432" "432" "432"
## [23] "432"
```

```r
plane_speed <- as.numeric(plane_speed)
plane_speed
```

```
##  [1]  90  90 162 167 105 232 107 112 127 162 126  95 432 202 108 432 105
## [18] 432 432 432 432 432 432
```

The only difference in the outputs is that in the character output, the values are surrounded by quotes. This transformation doesn't change the values from one step to another unlike the previous step, where who knows what transformation was made! Thus, in order to preserve the original value of a factor, one must convert to character and then to numeric!

10.

```r
# I have created a data frame with only three of the variables, as asked
planes_new <- planes[, c(1, 4, 5)]
```

1.3 Cleaning data with tidyverse methods

1.

```r
# I am reading in the data
airports <- read_csv(file = "airports.csv")
```

```
## Parsed with column specification:
## cols(
##   FAA = col_character(),
##   NAME = col_character(),
##   LAT = col_double(),
##   LON = col_double(),
##   ALT = col_integer(),
##   TZ = col_integer(),
##   DST = col_character()
## )
```

2.

```
# I have created a new data frame with select variables
airports_new <- select(airports, FAA, LAT, LON)
```

3.

```
library(dplyr)
# I have renamed the variables from uppercase to lowercase
airports_new <- rename(airports_new, faa = FAA, lat = LAT, lon = LON)
```

2. Joining data together

1.

```
# I am reading in the data for flights
flights <- read_csv(file = "flights.csv")
```

```
## Parsed with column specification:
## cols(
##   year = col_integer(),
##   month = col_integer(),
##   day = col_integer(),
##   dep_time = col_integer(),
##   sched_dep_time = col_integer(),
##   dep_delay = col_integer(),
##   arr_time = col_integer(),
##   sched_arr_time = col_integer(),
##   arr_delay = col_integer(),
##   carrier = col_character(),
##   flight = col_integer(),
##   tailnum = col_character(),
##   origin = col_character(),
##   dest = col_character(),
##   air_time = col_integer(),
##   hour = col_integer(),
##   minute = col_integer(),
##   time_hour = col_datetime(format = "")
## )
```

2.

```
# This returns the dimensions of the data frame
dim.data.frame(flights)
```

```
## [1] 336776     18
```

```
# This returns the variable names
names(flights)
```

```
##  [1] "year"           "month"          "day"            "dep_time"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
##  [9] "arr_delay"      "carrier"        "flight"         "tailnum"
## [13] "origin"         "dest"           "air_time"       "hour"
## [17] "minute"         "time_hour"
```

The dimensions are 336776 by 18 and the variable names are listed.

3. I have noticed this fact.

4. They both have the variable `tailnum` that uniquely identifies each flight.

5.

```
# I am joining the two datasets into one
flights_new <- left_join(flights, planes_new,
                         by = c("tailnum" = "tailnum"))
```

6.

```
# This returns the dimensions of the data frame
dim.data.frame(flights_new)
```

```
## [1] 336776     20
```

```
# This returns the variable names
names(flights_new)
```

```
##  [1] "year"           "month"          "day"            "dep_time"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
##  [9] "arr_delay"      "carrier"        "flight"         "tailnum"
## [13] "origin"         "dest"           "air_time"       "hour"
## [17] "minute"         "time_hour"      "manufacturer"   "model"
```

Yay! The dimensions are 336776 by 20, just as we expected! This join has been successful. The variable names have also been outputted.

7. The variable from `flights_new` is origin and the variable from `airports_new` is faa. They both contain the same category of character

8.

```
# I am joining the two data sets together
left_join(flights_new, airports_new,
          by = c("origin" = "faa"))
```

```
## # A tibble: 336,776 x 22
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <int>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 15 more variables: sched_arr_time <int>,
## #   arr_delay <int>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <int>, hour <int>, minute <int>,
## #   time_hour <dttm>, manufacturer <fctr>, model <fctr>, lat <dbl>,
## #   lon <dbl>
```

9.

```
# I am assigning the join to flights_new
flights_new <- left_join(flights_new, airports_new,
```

```
         by = c("origin" = "faa"))

# I am renaming two of the variables
flights_new <- rename(flights_new, origin_lat = lat, origin_lon = lon)

# This returns the dimensions of the data frame
dim.data.frame(flights_new)
```

```
## [1] 336776     22
```

```
# This returns the variable names
names(flights_new)
```

```
##  [1] "year"           "month"          "day"            "dep_time"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
##  [9] "arr_delay"      "carrier"        "flight"         "tailnum"
## [13] "origin"         "dest"           "air_time"       "hour"
## [17] "minute"         "time_hour"      "manufacturer"   "model"
## [21] "origin_lat"     "origin_lon"
```

The dimemsions of flights_new is 336776 by 24, and the variable names are listed.

10.

```
# I am joining the two data sets together
flights_new <- left_join(flights_new, airports_new,
         by = c("dest" = "faa"))

# I am renaming two of the variables
flights_new <- rename(flights_new, dest_lat = lat, dest_lon = lon)

# This returns the dimensions of the data frame
dim.data.frame(flights_new)
```

```
## [1] 336776     24
```

```
# This returns the variable names
names(flights_new)
```

```
##  [1] "year"           "month"          "day"            "dep_time"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
##  [9] "arr_delay"      "carrier"        "flight"         "tailnum"
## [13] "origin"         "dest"           "air_time"       "hour"
## [17] "minute"         "time_hour"      "manufacturer"   "model"
## [21] "origin_lat"     "origin_lon"     "dest_lat"       "dest_lon"
```

The dimensions of flights_new is 336776 by 24, and the variable names are listed. Yay, this is what we expected, so we did this correctly!

11.

```
# I am adding another variable to this data frame called distance
flights_new <- mutate(flights_new, distance = sqrt((dest_lat - origin_lat)^2 + (dest_lon - origin_lon)^
```

3. Combining data manipulations and plots

1.

```r
library(knitr)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.2
```

```r
# I have created an ascending table of the top 5 manufacturers used! So cool!
flights_new %>%
  group_by(manufacturer) %>%
  summarize(planes_used = n()) %>%
  top_n(n = 5) %>%
  arrange(desc(planes_used)) %>%
  kable()
```

```
## Selecting by planes_used
```

manufacturer	planes_used
BOEING	82912
EMBRAER	66068
NA	52606
AIRBUS	47302
AIRBUS INDUSTRIE	40891

The output shows the top 5 manufacturers of the airplanes.

2. `flights_new` is first `grouped by` manufacturer of each plane so that it is ordered neatly. Then, `summarize` counts the number of planes used from each manufacturer. Then, `top_n` selects the top 5 manufacturers that were calculated using the previous step. Finally, the table is `arranged` in a descending order of manufacturer popularity.

3.

```r
# I am combining the two manufacturers to join them into 1 Airbus
flights_new$manufacturer[flights_new$manufacturer == "AIRBUS INDUSTRIE"] <- "AIRBUS"

# I am creating an ascending table of the top 3 manufacturers used
flights_new %>%
  group_by(manufacturer) %>%
  summarize(planes_used = n()) %>%
  top_n(n = 3) %>%
  arrange(desc(planes_used)) %>%
  kable()
```

```
## Selecting by planes_used
```

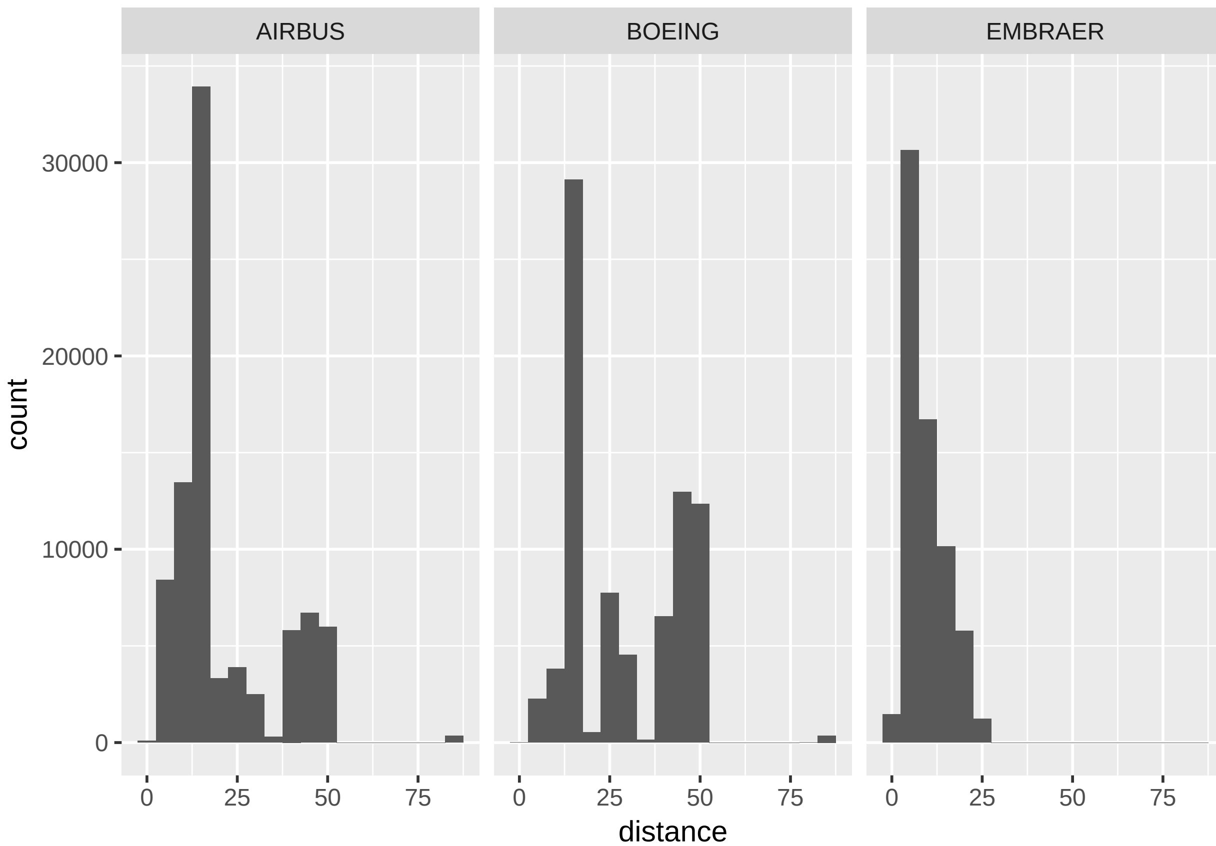
manufacturer	planes_used
AIRBUS	88193
BOEING	82912
EMBRAER	66068

Yay! We can see that we were successful in Part 3, since Airbus has combined its data and is the leading manufacturer!

4.

```r
# I have created a faceted histogram! Wow!
flights_new %>%
  filter(manufacturer %in% c("AIRBUS", "BOEING", "EMBRAER")) %>%
  ggplot() +
  geom_histogram(aes(x = distance), binwidth = 5) +
  facet_wrap(~manufacturer, ncol = 3)
```

```
## Warning: Removed 6096 rows containing non-finite values (stat_bin).
```



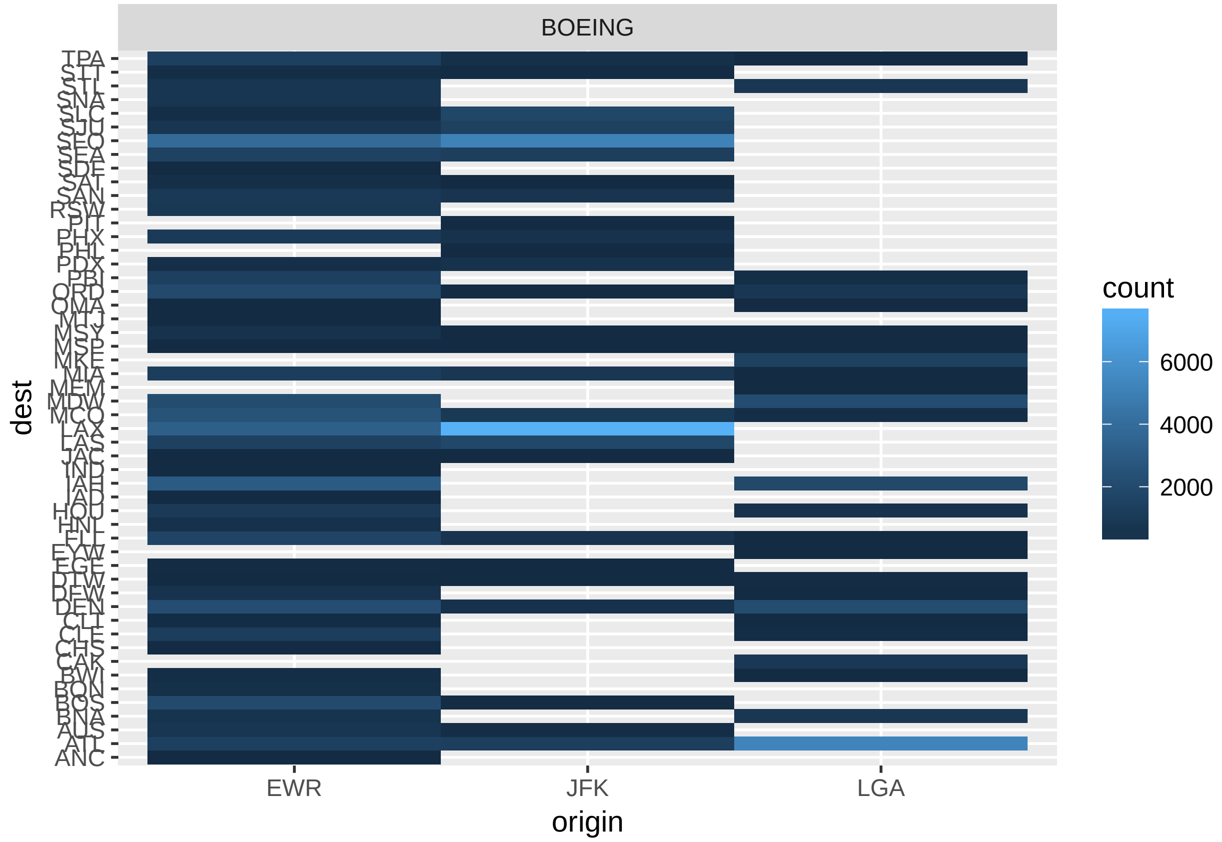
Look at the beauty I have created. It's such an easy histogram to read!

5. `flights_new` is first filtered so that only those values in the three manufacturer list is `filtered`. Then, a `ggplot` is requested. A `histogram` is created where on the x-axis lies distance with a bin width of 5. Finally, to see all three separately, a `facet_wrap` is used to simplify the data by `manufacturer`, and thus in 3 columns.

6. All three manufacturers have models that are slightly skewed right, meaning that the majority of their flights are local since they don't fly far. Airbus is closer to an even distribution, where the frequency of flights are about the same with increasing distance. The majority of the flight distance for Airbus is about 10. All three have different means, meaning that all of their average flights have different distances, where Embraer has the smallest mean. Airbus and Boeing have a larger spread meaning they can travel a larger range of distances. Moreover, Embraer does not fly over a value of 25, while the others do, meaning it is mostly concentrated in local flights. All three are very notably flying distances within 0-10 distances where their modes peak, so it seems like they are mostly flown for closer distances.

7.

```r
# I am attempting to create a heat map, but I am getting so many errors because I do not know the synta
flights_new %>%
  filter(manufacturer %in% c("BOEING")) %>%
  ggplot() +
  geom_bin2d(aes(x = origin, y = dest), binwidth = 5) +
  facet_wrap(~manufacturer, ncol = 3, nrow = 5)
```



Here is a heat map of all of the destinatations each flight of Boeing lands, according to origin.

```r
# I have created a table of the top 5 destinations of Boeing since the heatmap is beyond my abilities,
flights_new %>%
  filter(manufacturer == "BOEING") %>%
  group_by(dest) %>%
  summarize(destinations_visited = n()) %>%
  top_n(n = 5) %>%
  arrange(desc(destinations_visited)) %>%
  kable()
```

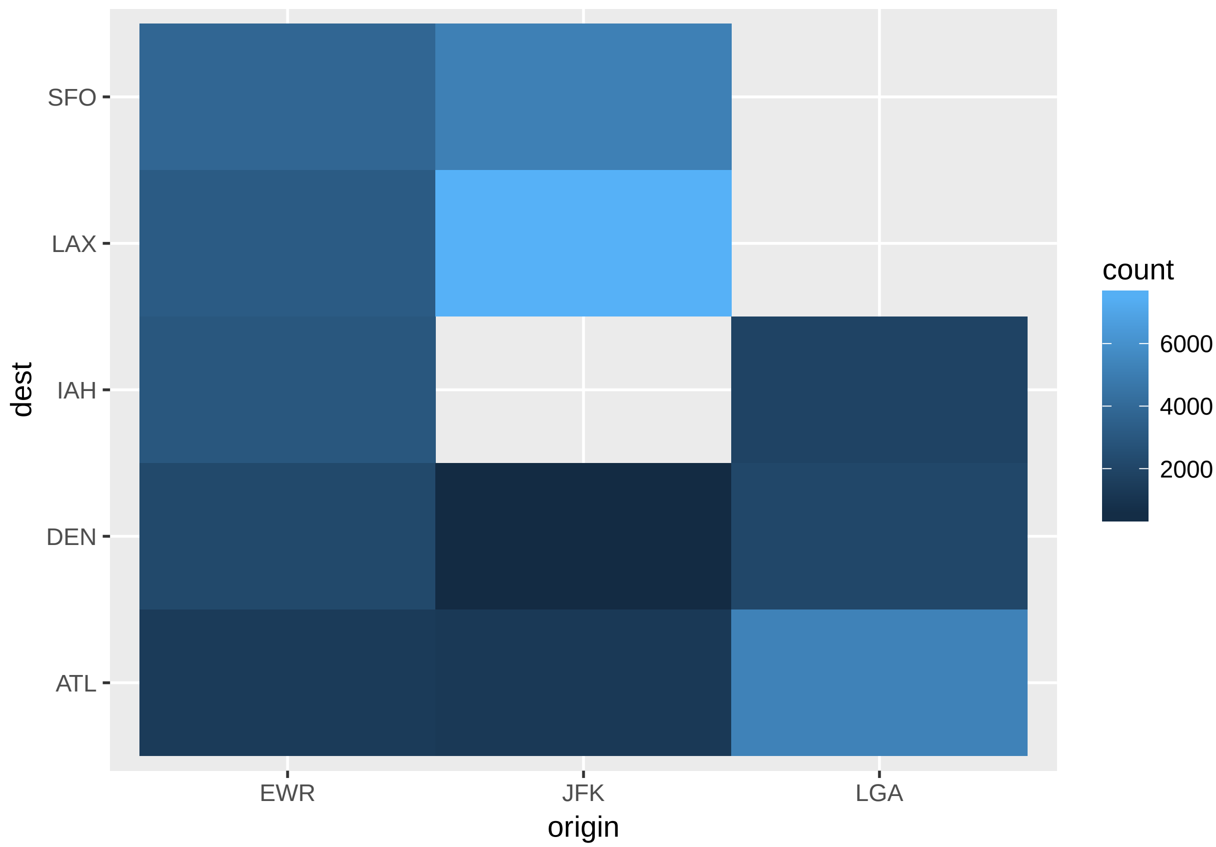
```
## Selecting by destinations_visited
```

dest	destinations_visited
LAX	10794
SFO	8928
ATL	7827
IAH	4766
DEN	4529

Since we have not learned how to create a heat map, my frustration is impeding me from trying to figure it out. So instead, I have created a table where we see that the top 5 destinations visted are LAX, SFO, ATL, IAH, DEN.

8.

```
# I am creating a table of the top destination of Boeing after taking off from JFK
flights_new %>%
  filter(manufacturer == "BOEING") %>%
  filter(dest %in% c("LAX", "SFO", "ATL", "IAH", "DEN")) %>%
  #group_by(origin) %>%
  #summarize(JFK_popular_land = n()) %>%
  #top_n(n = 5) %>%
  #arrange(desc(JFK_popular_land)) %>%
  #kable()
  ggplot() +  geom_bin2d(aes(x = origin, y = dest))
```



As you can see, LAX is the most frequented destination from the airport at JFK. This can be figured out using pipes, not heat maps!