Bureau of Transportation Statistics

2022-07-27

Introduction

In this lab we explore flights, specifically a random sample of domestic flights that departed from the three major New York City airport in 2013. We will generate simple graphical and numerical summaries of data on these flights and explore delay times. As this is a large data set, along the way you'll also learn the indispensable skills of data processing and subsetting.

```
library(statsr)
library(dplyr)
library(ggplot2)
```

Data

The Bureau of Transportation Statistics (BTS) is a statistical agency that is a part of the Research and Innovative Technology Administration (RITA). As its name implies, BTS collects and makes available transportation data, such as the flights data we will be working with in this lab.

```
data(nycflights)
```

To view the names of the variables, type the command

```
names(nycflights)
```

```
##  [1] "year"      "month"     "day"       "dep_time"  "dep_delay" "arr_time
##  [7] "arr_delay" "carrier"   "tailnum"   "flight"    "origin"    "dest"
## [13] "air_time"  "distance"  "hour"      "minute"
```

A very useful function for taking a quick peek at your data frame, and viewing its dimensions and data types is str, which stands for structure.

```
str(nycflights)
```

```
## tibble [32,735 x 16] (S3: tbl_df/data.frame)
##  $ year     : int [1:32735] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2
```

```
013 ...
##  $ month    : int [1:32735] 6 5 12 5 7 1 12 8 9 4 ...
##  $ day      : int [1:32735] 30 7 8 14 21 1 9 13 26 30 ...
##  $ dep_time : int [1:32735] 940 1657 859 1841 1102 1817 1259 1920 725 1323
 ...
##  $ dep_delay: num [1:32735] 15 -3 -1 -4 -3 -3 14 85 -10 62 ...
##  $ arr_time : int [1:32735] 1216 2104 1238 2122 1230 2008 1617 2032 1027 1
549 ...
##  $ arr_delay: num [1:32735] -4 10 11 -34 -8 3 22 71 -8 60 ...
##  $ carrier  : chr [1:32735] "VX" "DL" "DL" "DL" ...
##  $ tailnum  : chr [1:32735] "N626VA" "N3760C" "N712TW" "N914DL" ...
##  $ flight   : int [1:32735] 407 329 422 2391 3652 353 1428 1407 2279 4162
...
##  $ origin   : chr [1:32735] "JFK" "JFK" "JFK" "JFK" ...
##  $ dest     : chr [1:32735] "LAX" "SJU" "LAX" "TPA" ...
##  $ air_time : num [1:32735] 313 216 376 135 50 138 240 48 148 110 ...
##  $ distance : num [1:32735] 2475 1598 2475 1005 296 ...
##  $ hour     : num [1:32735] 9 16 8 18 11 18 12 19 7 13 ...
##  $ minute   : num [1:32735] 40 57 59 41 2 17 59 20 25 23 ...
```

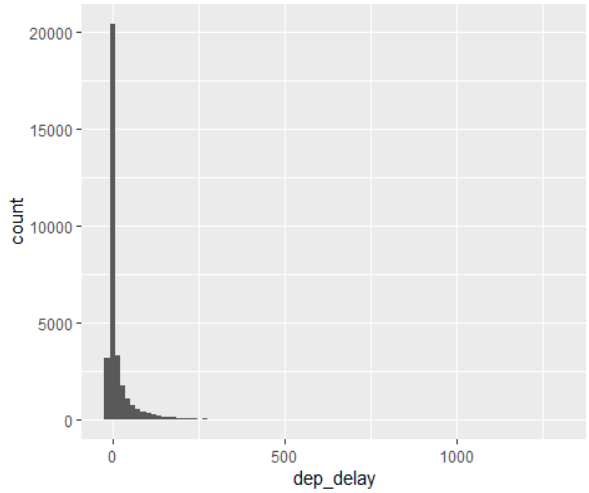
Seven verbs

The dplyr package offers seven verbs (functions) for basic data manipulation:

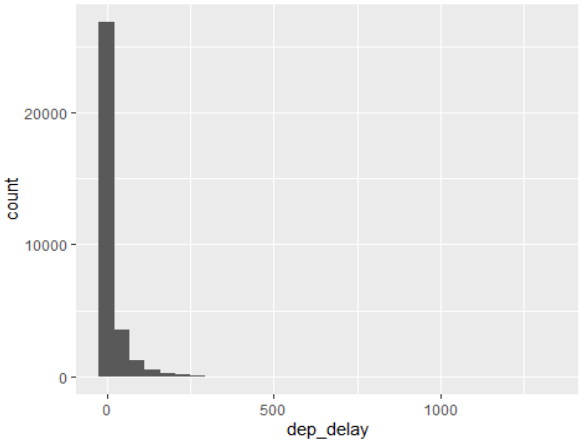
filter()
arrange()
select()
distinct()
mutate()
summarise()
sample_n()

Analysis

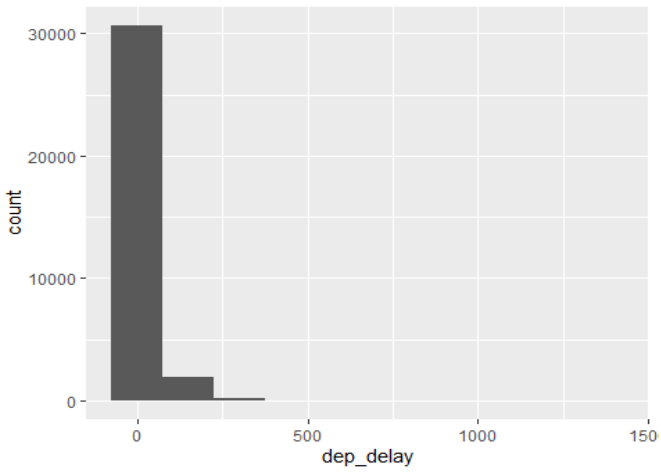
```
ggplot(data = nycflights, aes(x = dep_delay)) +
  geom_histogram()
```




```
ggplot(data = nycflights, aes(x = dep_delay)) +
  geom_histogram(binwidth = 15)
```

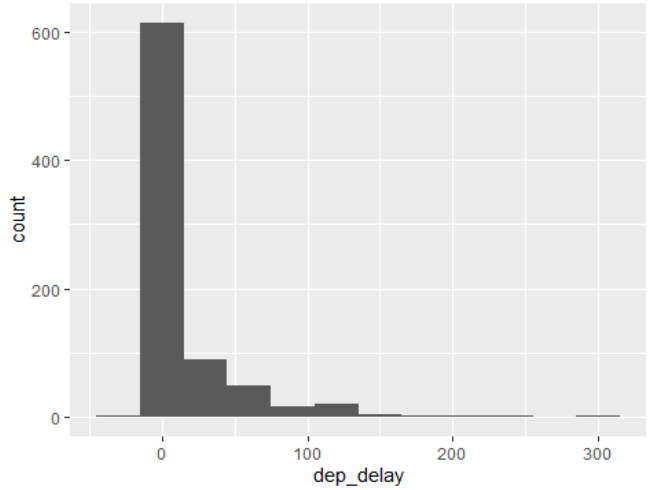



```
ggplot(data = nycflights, aes(x = dep_delay)) +
  geom_histogram(binwidth = 150)
```



If we want to focus on departure delays of flights headed to RDU only, we need to first filter the data for flights headed to RDU (dest == "RDU") and then make a histogram of only departure delays of only those flights.

```
rdu_flights <- nycflights %>%
  filter(dest == 'RDU')
ggplot(data = rdu_flights, aes(x=dep_delay)) +
  geom_histogram(binwidth = 30)
```



Note that in the summarise function we created a list of two elements. The names of these elements are user defined, like mean_dd, sd_dd, n, and you could customize these names as you like (just don't use spaces in your names). Calculating these summary statistics also require that you know the function calls. Note that n() reports the sample size.

```
rdu_flights %>%
  summarise(mean_dd = mean(dep_delay), sd_dd = sd(dep_delay),n= n())

## # A tibble: 1 x 3
##   mean_dd sd_dd     n
##     <dbl> <dbl> <int>
## 1    11.7  35.6   801
```

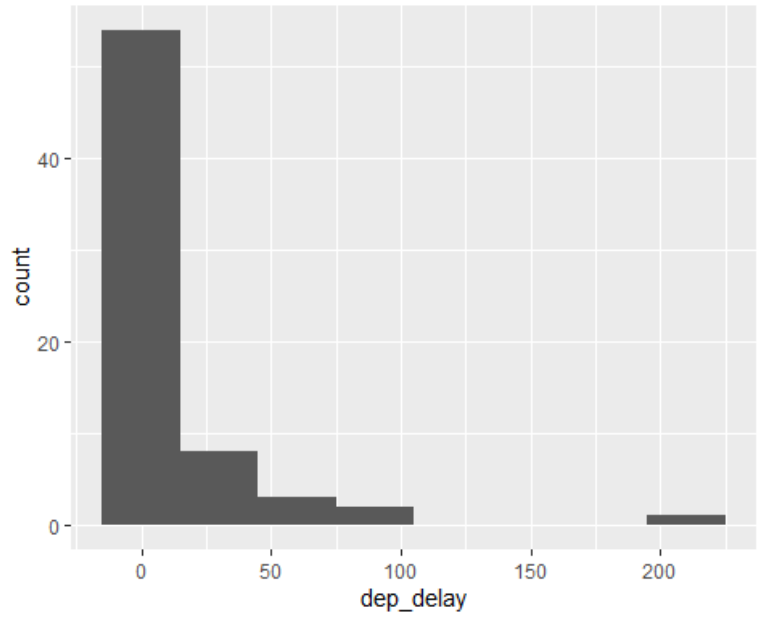
We can also filter based on multiple criteria. Suppose we are interested in flights headed to San Francisco (SFO) in February:

```r
sfo_feb_flights <- nycflights %>%
  filter(dest == "SFO", month == 2)
dim(sfo_feb_flights)
```

```
## [1] 68 16
```

Make a histogram and calculate appropriate summary statistics for arrival delays of sfo_feb_flights. Which of the following is false?

```r
ggplot(data = sfo_feb_flights, aes(x=dep_delay))+
  geom_histogram(binwidth = 30)
```



Another useful functionality is being able to quickly calculate summary statistics for various groups in your data frame. For example, we can modify the above command using the group_by function to get the same summary stats for each origin airport:

```r
rdu_flights %>%
  group_by(origin)%>%
  summarise(mean_dd = mean(dep_delay), sd_dd = sd(dep_delay),n=n())
```

```
## # A tibble: 3 x 4
##   origin mean_dd sd_dd     n
##   <chr>    <dbl> <dbl> <int>
## 1 EWR      13.4   32.1   145
```

```
## 2 JFK      15.4   40.3   300
## 3 LGA       7.90  32.2   356
```

Calculate the median and interquartile range for arr_delays of flights in the sfo_feb_flights data frame, grouped by carrier. Which carrier has the highest IQR of arrival delays?

```
sfo_feb_flights%>%
  group_by(carrier)%>%
  summarise(iqr = IQR(arr_delay))%>%
  arrange(desc(iqr))
```

```
## # A tibble: 5 x 2
##   carrier   iqr
##   <chr>   <dbl>
## 1 DL       22
## 2 UA       22
## 3 VX       21.2
## 4 AA       17.5
## 5 B6       12.2
```

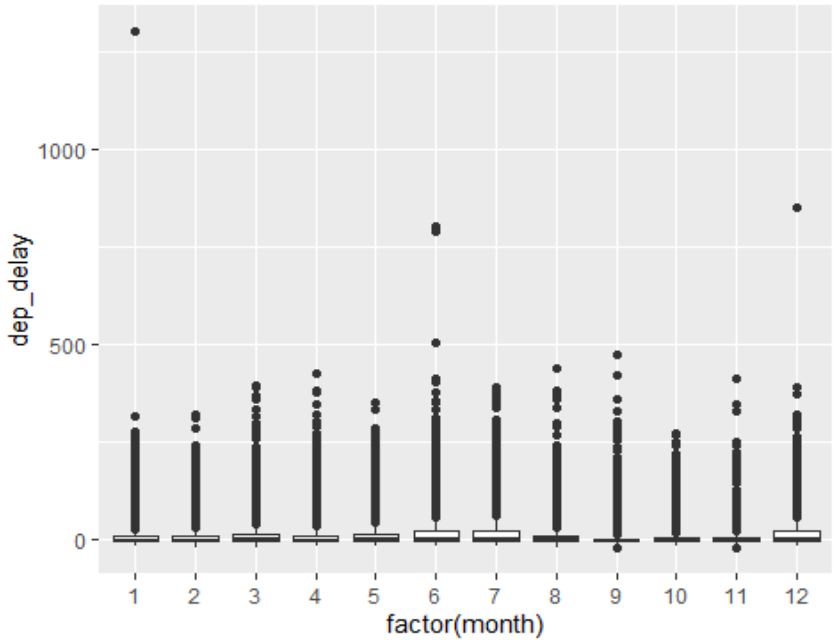
Departure delays over months

```
nycflights %>%
  group_by(month) %>%
  summarise(mean_dd = mean(dep_delay)) %>%
  arrange(desc(mean_dd))
```

```
## # A tibble: 12 x 2
##    month mean_dd
##    <int>   <dbl>
##  1     7   20.8
##  2     6   20.4
##  3    12   17.4
##  4     4   14.6
##  5     3   13.5
##  6     5   13.3
##  7     8   12.6
##  8     2   10.7
##  9     1   10.2
## 10     9    6.87
## 11    11    6.10
## 12    10    5.88
```

We can also visualize the distributions of departure delays across months using side-by-side box plots:

```
ggplot(nycflights, aes(x = factor(month), y = dep_delay)) +
  geom_boxplot()
```



There is some new syntax here: We want departure delays on the y-axis and the months on the x-axis to produce side-by-side box plots. Side-by-side box plots require a categorical variable on the x-axis, however in the data frame month is stored as a numerical variable (numbers 1 - 12). Therefore we can force R to treat this variable as categorical, what R calls a factor, variable with factor(month).

On time departure rate for NYC airports

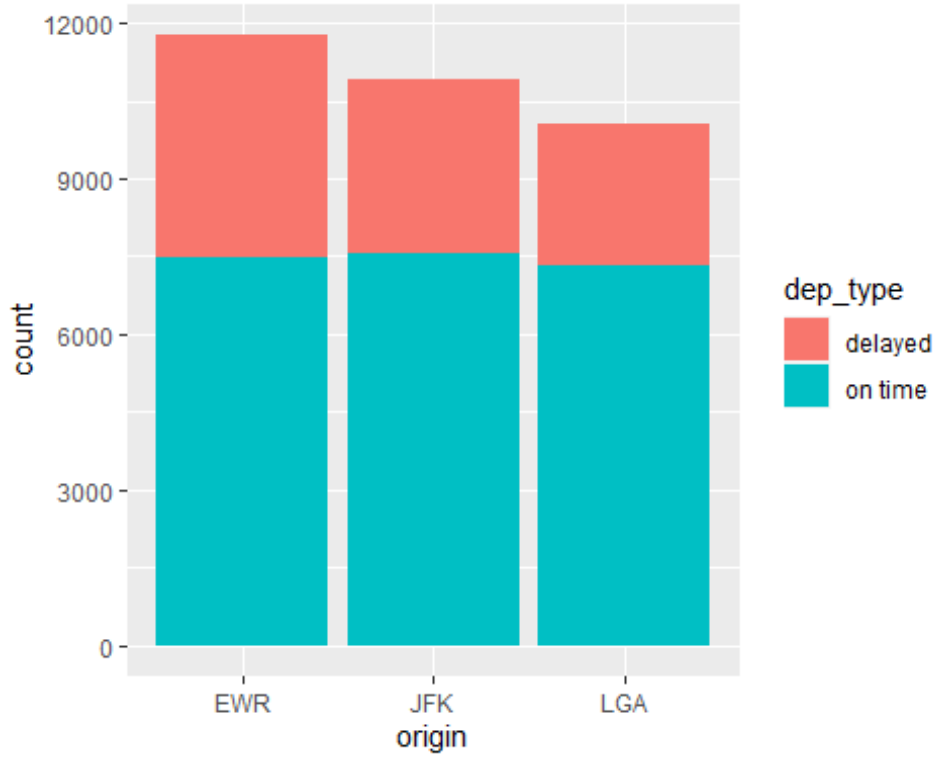
```
nycflights <- nycflights%>%
  mutate(dep_type = ifelse(dep_delay < 5, 'on time', 'delayed'))

nycflights %>%
  group_by(origin)%>%
  summarise(ot_dep_rate = sum(dep_type == 'on time')/n())%>%
  arrange(desc(ot_dep_rate))
```

```
## # A tibble: 3 x 2
##   origin ot_dep_rate
##   <chr>        <dbl>
## 1 LGA          0.728
```

```
## 2 JFK          0.694
## 3 EWR          0.637
```

We can also visualize the distribution of on on time departure rate across the three airports using a segmented bar plot.

```
ggplot(data=nycflights, aes(x=origin, fill =dep_type))+
  geom_bar()
```



Mutate the data frame so that it includes a new variable that contains the average speed, avg_speed traveled by the plane for each flight (in mph). What is the tail number of the plane with the fastest avg_speed? Hint: Average speed can be calculated as distance divided by number of hours of travel, and note that air_time is given in minutes. If you just want to show the avg_speed and tailnum and none of the other variables, use the select function at the end of your pipe to select just these two variables with select(avg_speed, tailnum). You can Google this tail number to find out more about the aircraft.

```
flight_speed <- nycflights %>%
  mutate(average_speed = distance*60/air_time)
```

```
flight_speed %>%
  select(average_speed,tailnum) %>%
  arrange(desc(average_speed))
```

```
##       average_speed tailnum
## 1          703.3846  N666DN
## 2          557.4419  N779JB
## 3          554.2197  N571JB
## 4          547.8857  N568JB
## 5          547.8857  N5EHAA
## 6          547.8857  N656JB
## 7          544.7727  N789JB
## 8          538.6517  N516JB
## 9          535.6425  N648JB
## 10         535.6425  N510JB
## 11         533.0387  N38268
## 12         533.0387  N53442
## 13         533.0387  N75858
## 14         532.6667  N624JB
```

```
ggplot(data= flight_speed, aes(x=distance, y = average_speed))+
  geom_point()
```