Final Reflection

2022-04-20

R Markdown

In this reflection i worked with various data frames in addition to data from nycflights13 package.This package contains information about all flights that departed from NYC (e.g. EWR, JFK and LGA) to destinations in the United States, Puerto Rico, and the American Virgin Islands) in 2013. The package provides the following tables.

- flights: all flights that departed from NYC in 2013
- weather: hourly meteorological data for each airport
- planes: construction information about each plane
- airports: airport names and locations
- airlines: translation between two letter carrier codes and names.

The analysis helped me meet the following objectives.

1. To import manage and clean data
2. To create graphical displays and numerical summaries of data for exploratory analysis and presentations.
3. To write R programs for simulations from probability models and randomization-based analysis and presentations.
4. To use source documentation and other resources to troubleshoot and extend R programs.
5. To write clear, efficient and well documented R programs.

1. To Import, Manage and Clean data

To meet this objective i learnt how to load flat files with readr package which is part of the core tidyverse. The readr functions turns flat files into data frames. I used read_csv which reads comma delimited files. To manage data i learnt how to export a csv file to excel with write_excel_csv() function. To clean data i used the tidyr package which is a member of the core tidyverse.

Import

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(nycflights13)
```

```
fatalities_from_terrorism_1_ <- read_csv("~/fatalities-from-terrorism (1).csv")
```

```
## Rows: 4373 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Entity, Code
## dbl (2): Year, Terrorism fatalities (GTD, 2018)
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
View(fatalities_from_terrorism_1_)
```

Manage

To manage data i learnt how to export a csv file to excel with write_excel_csv() function.

```
write_excel_csv(fatalities_from_terrorism_1_,"fatilities.xlsx")
```

Data cleaning

To have a clean data three rules must be achieved.

1. Each variable must have its own column.
2. Each value must have its own row.
3. Each value must have its own cell.
   I used tidyr package,a package that provides a bunch of tools to help tidy up messy datasets. tidyr is a member of the core tidyverse.I used the following functions from tidyr.
    1. Pivoting:pivot_longer() to tidy data where the column names are not names of a variable but values of a variable and pivot_wider() when an observation is scattered across multiple rows.
    2. Separating and uniting:separate() to pull apart one column into multiple columns, by splitting wherever a separator character appears and unite() to combine multiple columns into a single column.
    3. Missing values:A value can be missing either explicitly(flagged with NA) or implicitly(not in the data).I handled Missing values using the following functions .
        1. values_drop_na=TRUE:To drop missing values when they are not important in other representations of the data.
        2. complete() takes a set of columns and finds all unique combinations and fills explicit NA where necessary.
        3. fill() takes a set of columns where you want missing values to be replaced by the most relevant non-missing values
        - pivot_longer()

```
table1<-read_csv("country,1999,2000
Brazil,37737,80488
China,212258,213766
Iraq,745,2666")
```

```
## Rows: 3 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): country
## dbl (2): 1999, 2000
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
view(table1)
```

```
table1%>%pivot_longer(c("1999","2000"),names_to="year",values_to="cases")
```

```
## # A tibble: 6 x 3
##   country year   cases
##   <chr>   <chr>  <dbl>
## 1 Brazil  1999   37737
## 2 Brazil  2000   80488
## 3 China   1999  212258
## 4 China   2000  213766
## 5 Iraq    1999     745
## 6 Iraq    2000    2666
```

```
view(table1)
```

- pivot_wider()

```
table2<-read_csv("country,year,type,count
                 iraq,1999,cases,745
                 iraq,1999,population,19987071
                 iraq,2000,cases,2666
                 iraq,2000,population,20595360
                 Brazil,1999,cases,37731
                 Brazil,1999,population,172006362
                 Brazil,2000,cases,80488
                 Brazil,2000,population,174504898")
```

```
## Rows: 8 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): country, type
## dbl (2): year, count
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
view(table2)
```

```
table2%>%pivot_wider(names_from=type,values_from=count)
```

```
## # A tibble: 4 x 4
##   country  year cases population
##   <chr>   <dbl> <dbl>      <dbl>
## 1 iraq     1999   745   19987071
## 2 iraq     2000  2666   20595360
## 3 Brazil   1999 37731  172006362
## 4 Brazil   2000 80488  174504898
```

```
view(table2)
```

- separate()

```
table3<-read_csv("country,year,rate
                 Iraq,1999,745/19987071
                 Iraq,2000,2666/20595360
                 Brazil,1999,37737/172006362
                 Brazil,2000,80488/174504898
                 China,1999,212258/1272915272
                 China,2000,213766/1280428583")
```

```
## Rows: 6 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): country, rate
## dbl (1): year
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
view(table3)
```

```
table3%>%separate(rate,into=c("cases","population"))
```

```
## # A tibble: 6 x 4
##   country  year cases  population
##   <chr>   <dbl> <chr>  <chr>
## 1 Iraq     1999 745    19987071
## 2 Iraq     2000 2666   20595360
## 3 Brazil   1999 37737  172006362
## 4 Brazil   2000 80488  174504898
## 5 China    1999 212258 1272915272
## 6 China    2000 213766 1280428583
```

```
table<-table3%>%separate(rate,into=c("cases","population"))
```

- unite()

```
table%>%unite(new,cases,population,sep="/")
```

```
## # A tibble: 6 x 3
##   country  year new
##   <chr>   <dbl> <chr>
## 1 Iraq     1999 745/19987071
## 2 Iraq     2000 2666/20595360
## 3 Brazil   1999 37737/172006362
## 4 Brazil   2000 80488/174504898
## 5 China    1999 212258/1272915272
## 6 China    2000 213766/1280428583
```

- missing values

```
stocks <- tibble(
  year   = c(2015, 2015, 2015, 2015, 2016, 2016, 2016),
  qtr    = c(   1,    2,    3,    4,    2,    3,    4),
  return = c(1.88, 0.59, 0.35,   NA, 0.92, 0.17, 2.66)
)
view(stocks)
```

- values_drop_na=TRUE

```
stocks %>%
  pivot_wider(names_from = year, values_from = return) %>%
  pivot_longer(
    cols = c(`2015`, `2016`),
    names_to = "year",
    values_to = "return",
    values_drop_na = TRUE
  )
```

```
## # A tibble: 6 x 3
##     qtr year  return
##   <dbl> <chr>  <dbl>
## 1     1 2015    1.88
## 2     2 2015    0.59
## 3     2 2016    0.92
## 4     3 2015    0.35
## 5     3 2016    0.17
## 6     4 2016    2.66
```

- complete()

```
stocks %>%
  complete(year, qtr)
```

```
## # A tibble: 8 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
## 5  2016     1  NA
## 6  2016     2   0.92
## 7  2016     3   0.17
## 8  2016     4   2.66
```

- fill()

```
stocks%>%fill(return)
```

```
## # A tibble: 7 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4   0.35
## 5  2016     2   0.92
## 6  2016     3   0.17
## 7  2016     4   2.66
```

2.To Create graphical displays and numerical summaries of data for exploratory analysis and presentations.

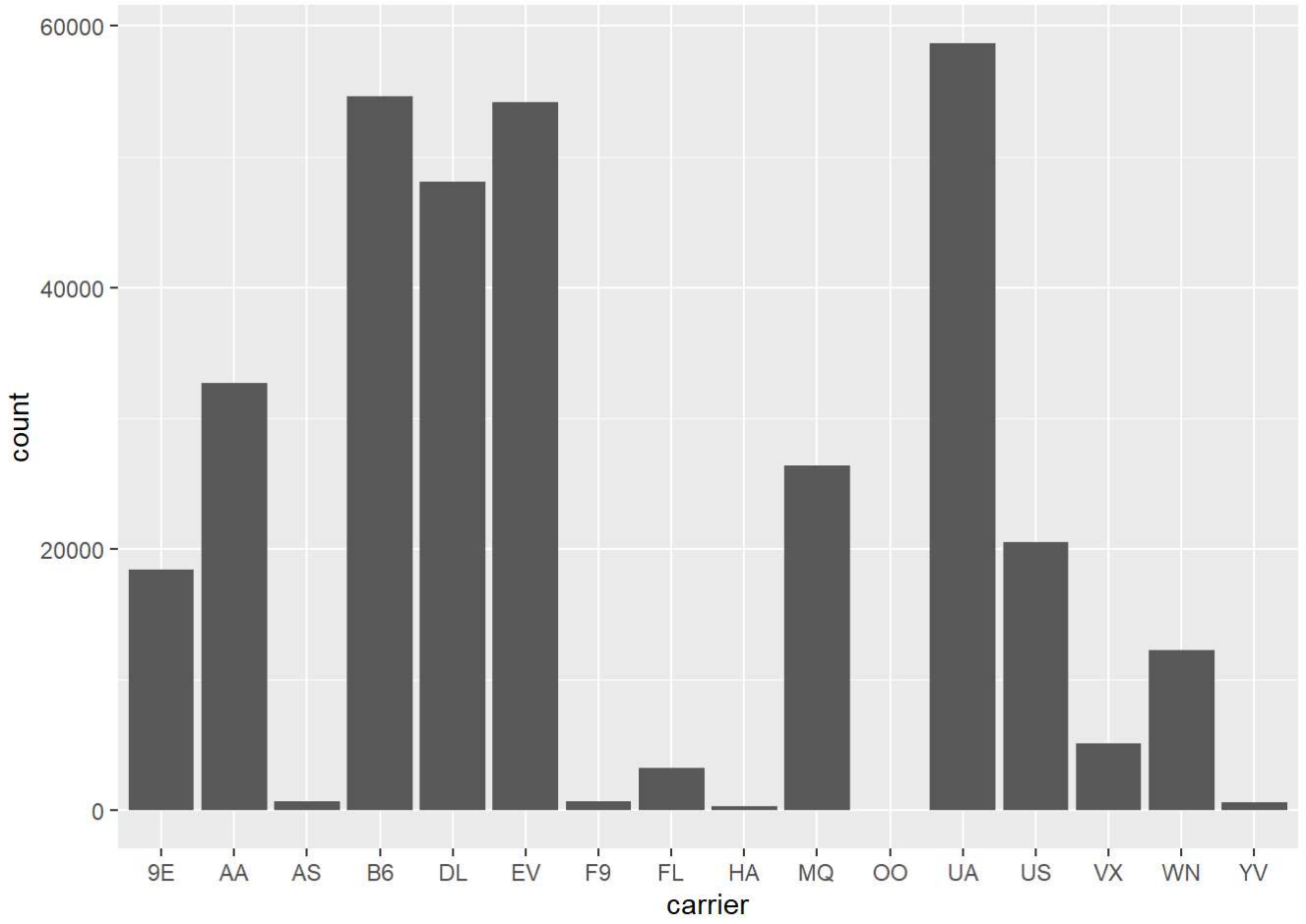
To meet this objective i learnt how to do exploratory data analysis and presentation(through graphical visualization). In exploratory data analysis,I looked at

1. Variation in the data by
    - Visualizing distributions.For continuous and for categorical variables
    - Checking typical values.
    - Checking unusual values.
2. Missing values.
3. Covariation.
4. Patterns and Models.

   For presentation of analysis,I created graphs using ggplot2 package.ggplot2 is the most versatile and elegant system of making graphs in R.Some of the graphical representations i created include
    - barcharts
    - scatterplots
    - facets
    - boxplots
    - smoothing lines
        - visualising distributions for categorical variable

```
library(nycflights13)
flights<-nycflights13::flights
view(flights)
```

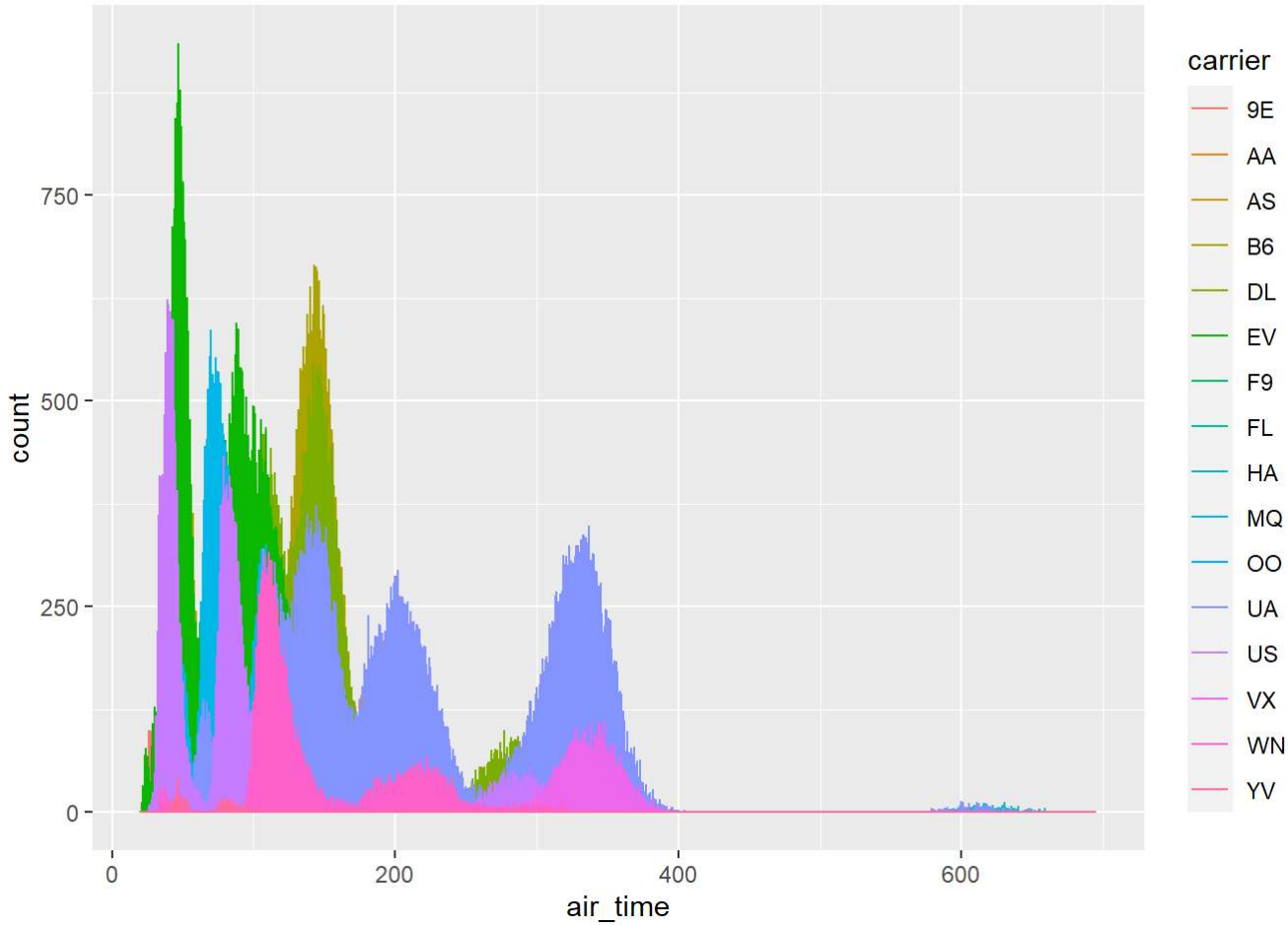
```
ggplot(data = flights) +
  geom_bar(mapping = aes(x = carrier))
```



- visualising distributions for continuous variable

```
ggplot(data = flights, mapping = aes(x = air_time, colour = carrier)) +
  geom_freqpoly(binwidth = 0.1)
```

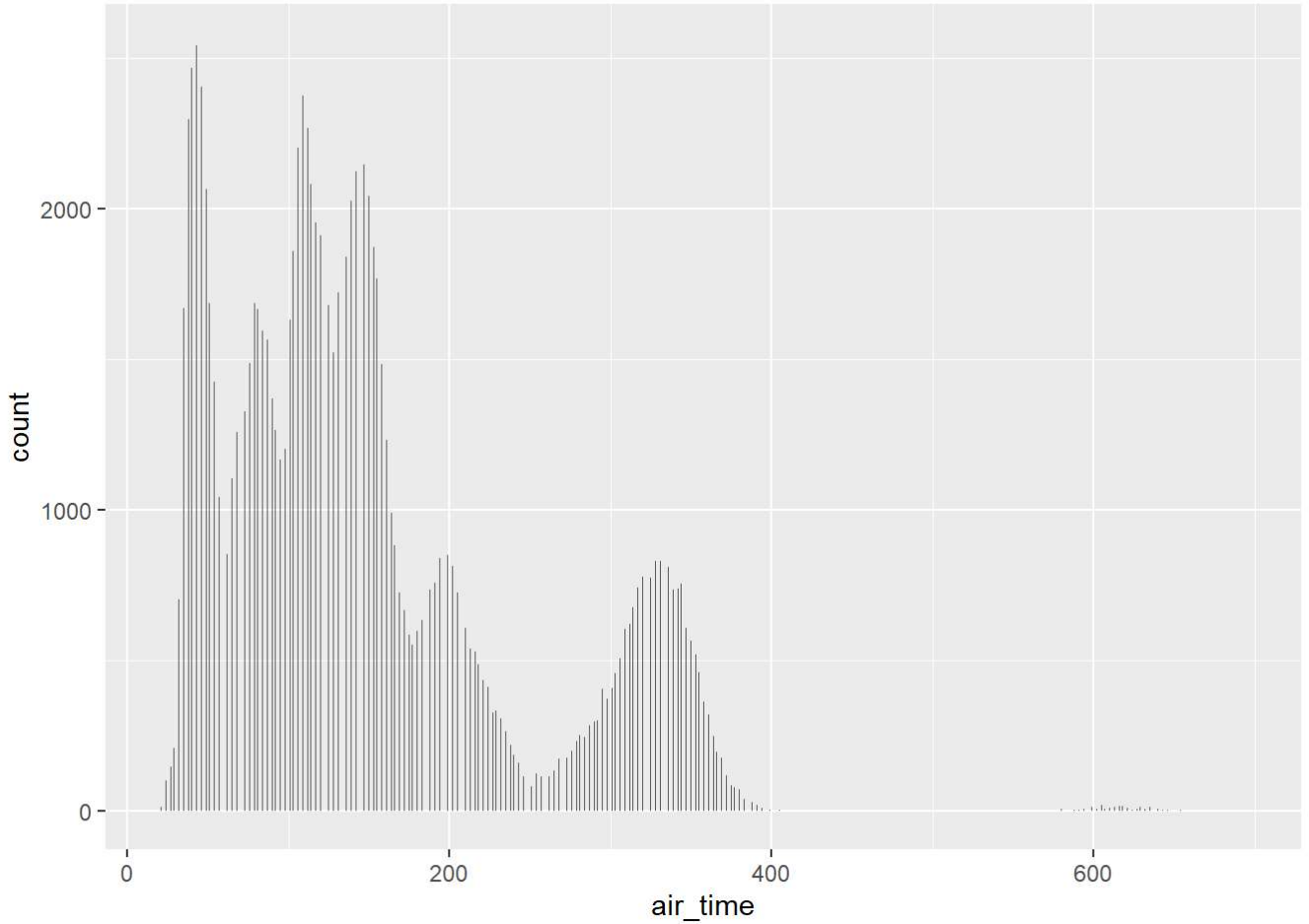
```
## Warning: Removed 9430 rows containing non-finite values (stat_bin).
```



- checking typical values
  This is important because it turns the graphical representation of distributions above into useful questions by looking for anything unexpected.For example in the graph below clusters of similar values suggested that subgroups exist in the data.This can answer questions like,how are the observations in each cluster similar to each other?,how are observations in separate clusters different from each other?,how can one describe the clusters? or is the appearance of clusters misleading?

```
ggplot(data = flights, mapping = aes(x = air_time)) +
  geom_histogram(binwidth = 0.2)
```

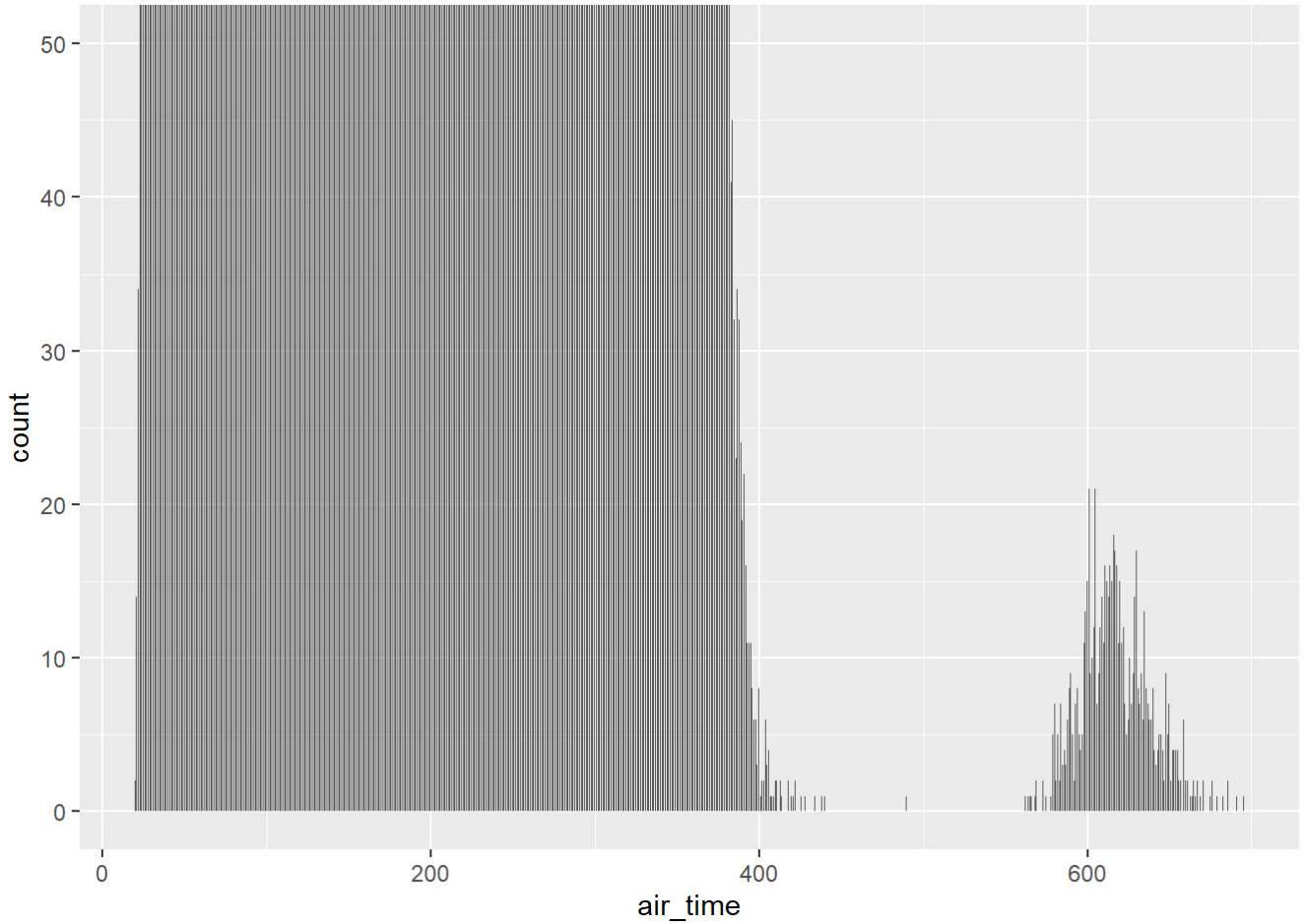
```
## Warning: Removed 9430 rows containing non-finite values (stat_bin).
```



- Checking unusual values
  This is important in spotting outliers. Outliers can be data entry errors or may suggest important information.To make it easier to see unusual values,i zoomed to small values of the y axis with coord_cartesian() function.

```
ggplot(flights) +
  geom_histogram(mapping = aes(x = air_time), binwidth = 0.5) +
  coord_cartesian(ylim = c(0, 50))
```

```
## Warning: Removed 9430 rows containing non-finite values (stat_bin).
```



- Missing values
  When encountered by unusual values in a dataset,there is always two options.If the data contains NA its always good to set na.rm=TRUE in order to surpress warnings from ggplot2 when plotting.
  - Drop the entire raw with the strange values

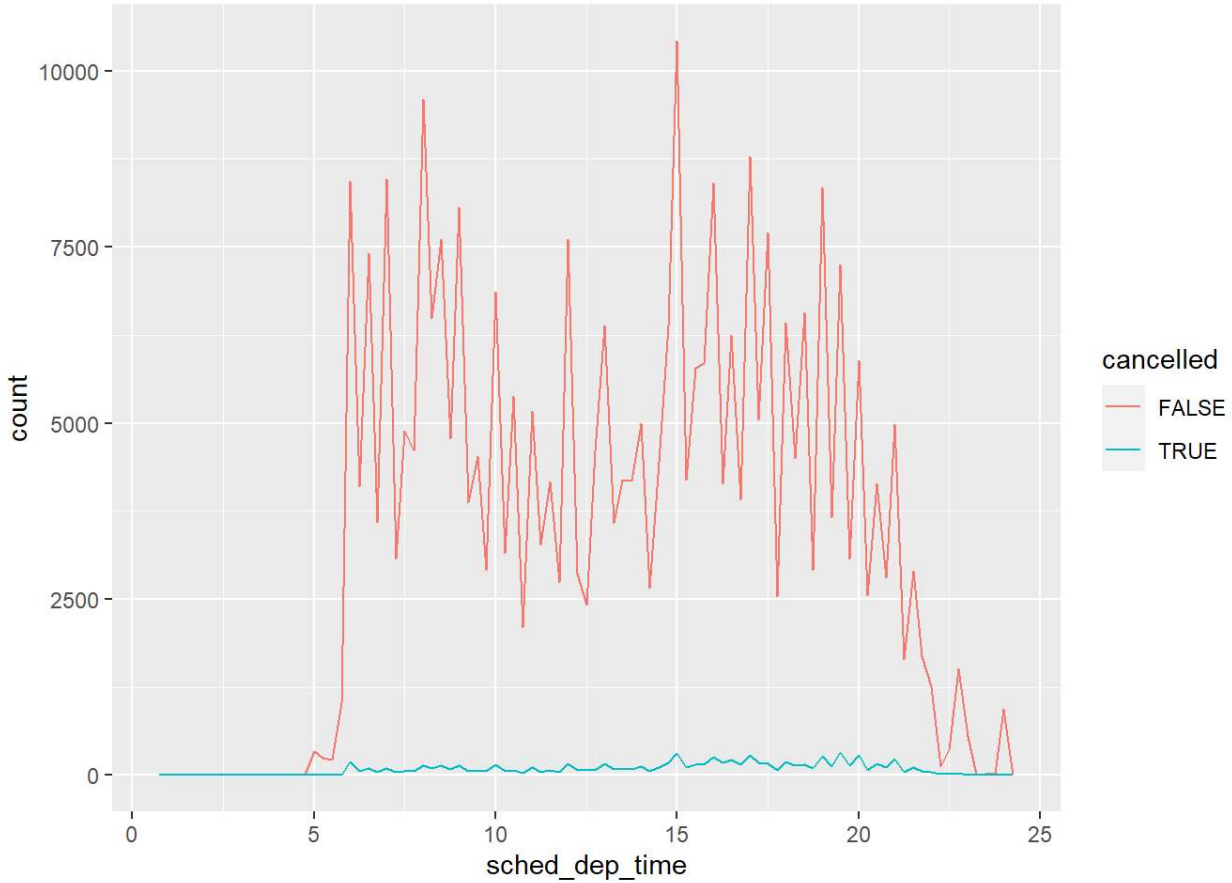
    ```
    stocks <- stocks %>%
      filter(between(return, 3, 7))
    view(stocks)
    ```

  - Replace unusual values with missing values using mutate()

```
stocks <- stocks %>%
  mutate(return = ifelse(return< 3 | return> 20, NA, return))
view(stocks)
```

It is always good to understand what makes observations with missing values different to observations with recorded values. For example, in nycflights13::flights, missing values in the dep_time variable indicate that the flight was cancelled. So I compared the scheduled departure times for cancelled and non-cancelled times.I did this by making a new variable with is.na().

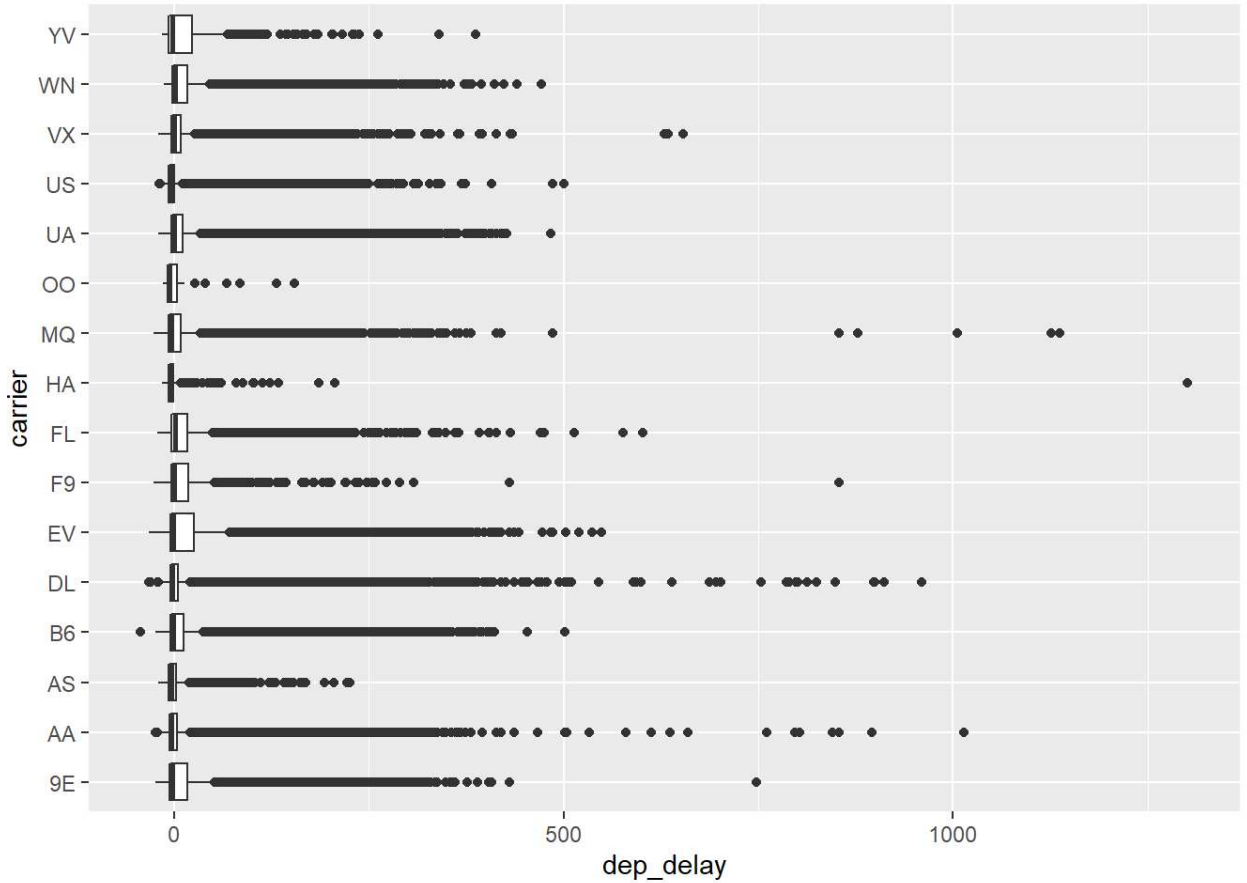
```
nycflights13::flights %>%
  mutate(
    cancelled = is.na(dep_time),
    sched_hour = sched_dep_time %/% 100,
    sched_min = sched_dep_time %% 100,
    sched_dep_time = sched_hour + sched_min / 60
  ) %>%
  ggplot(mapping = aes(sched_dep_time)) +
    geom_freqpoly(mapping = aes(colour = cancelled), binwidth = 1/4)
```



- Covariation of a categorical and continuous variable
  Describes the behavior between variables i.e values of two or more variables varying together in a related way.The best way to spot covariation is to visualise the relationship between two or more variables.For example i explored how the departure delay of a flight varies with its carrier using a boxplot.The graph below shows AA,B6,DL and US have a lower departure delay time on average.

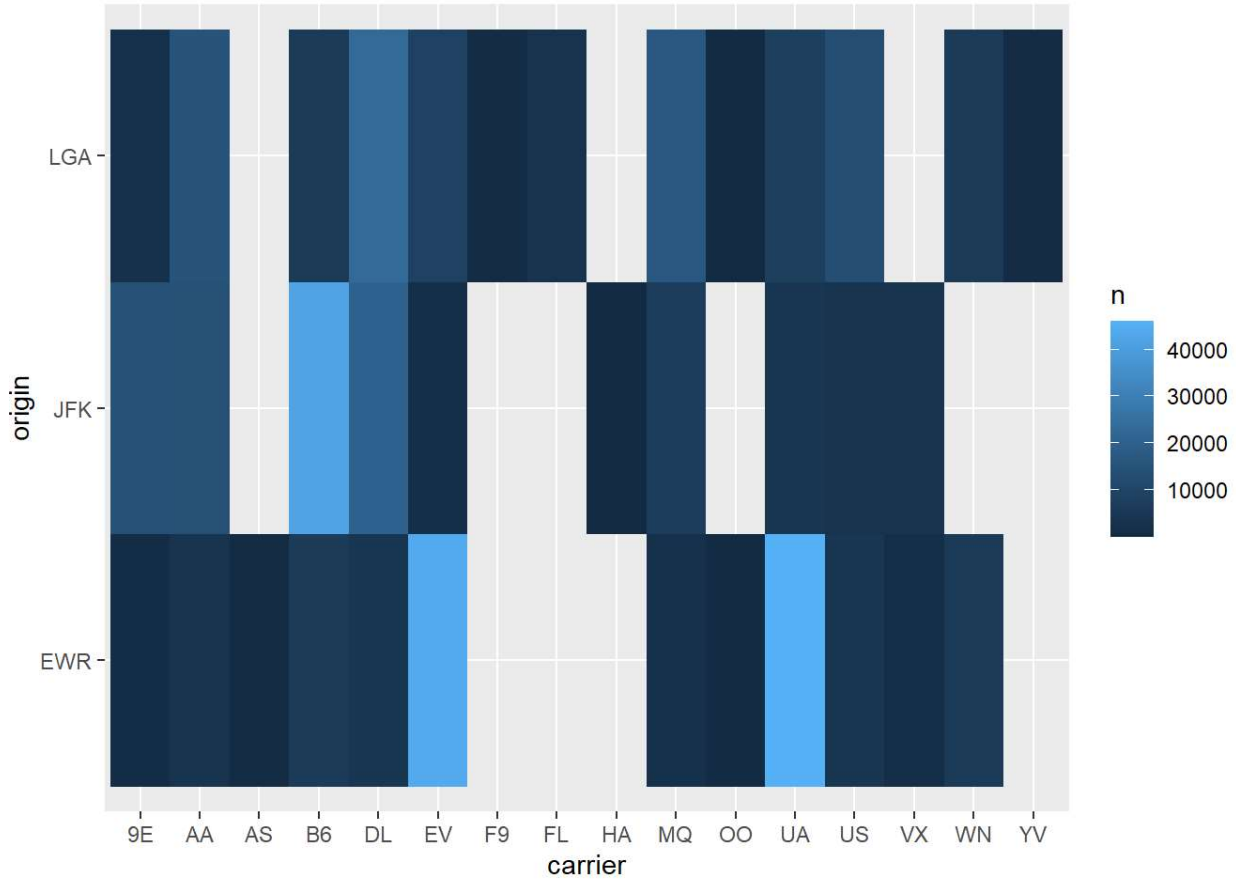
```
ggplot(data = flights, mapping = aes(x = carrier, y = dep_delay)) +
  geom_boxplot()+
  coord_flip()
```

```
## Warning: Removed 8255 rows containing non-finite values (stat_boxplot).
```



- Covariation in two categorical variables
  The of color of each block in the plot below displays how many observations occurred at each combination of values. Covariation will appear as a strong correlation between specific x values and specific y values.

```
flights %>%
  count(carrier,origin ) %>%
  ggplot(mapping = aes(x = carrier, y = origin)) +
    geom_tile(mapping = aes(fill = n))
```

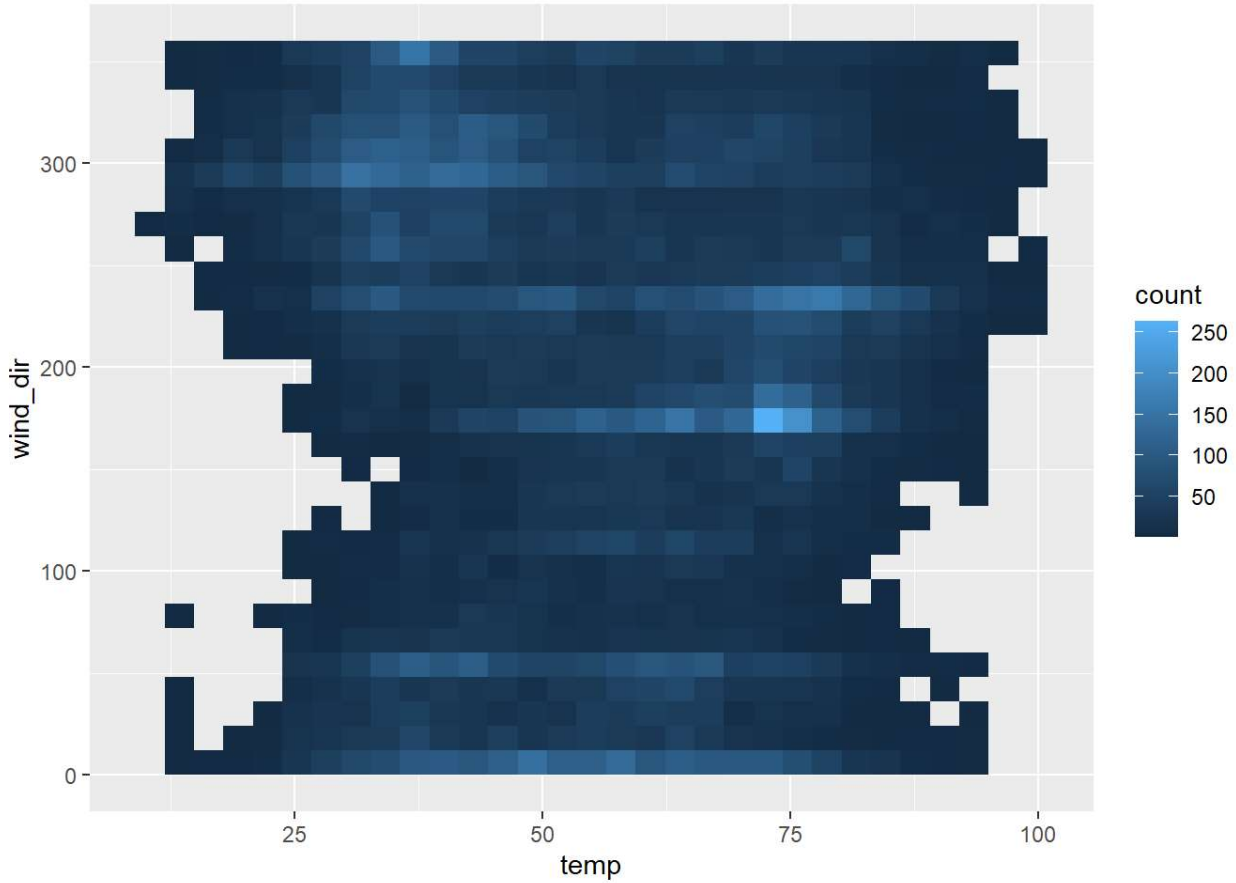


- Covariation between two continuous variables
  I used geom_bin2d() and geom_hex() to divide the coordinate plane into 2d bins and then used a fill color to display how many points fell into each bin. geom_bin2d() creates rectangular bins. geom_hex() creates hexagonal bins.

```
weather<-nycflights13::weather
view(weather)
```

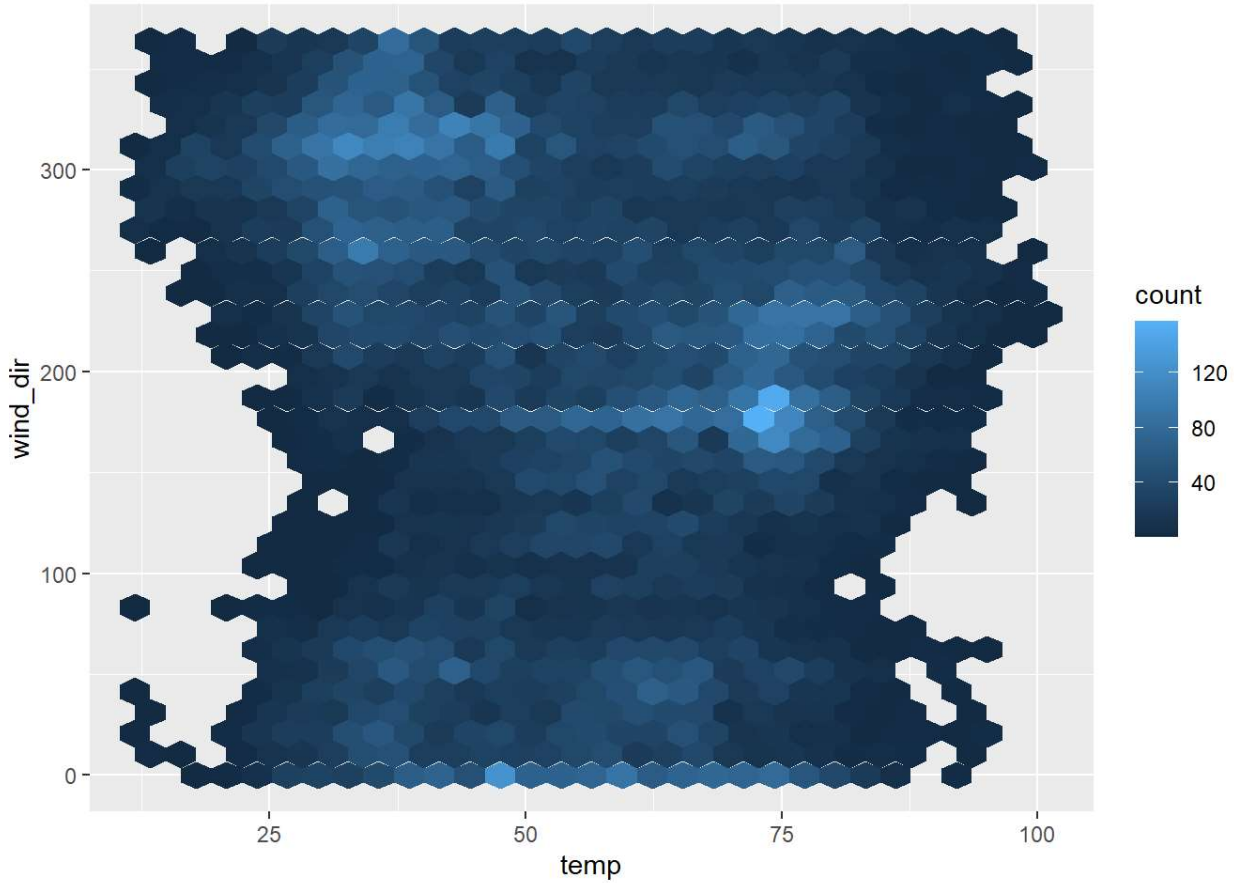
```
ggplot(data = weather) +
  geom_bin2d(mapping = aes(x = temp, y = wind_dir))
```

```
## Warning: Removed 461 rows containing non-finite values (stat_bin2d).
```




```
#install.packages("hexbin")
ggplot(data = weather) +
  geom_hex(mapping = aes(x = temp, y = wind_dir))
```

```
## Warning: Removed 461 rows containing non-finite values (stat_binhex).
```



3.To write R programs for simulations from probability models and randomization-based experiments

For this objective i built a linear model to fit the temperature in the weather table from nycflights13 package.

```
library(nycflights13)
weather<-nycflights13::weather
```

- I visualized the average recorded temperature every day using ggplot2 to watch the long term trend

```
#install.packages("lubridate")
```

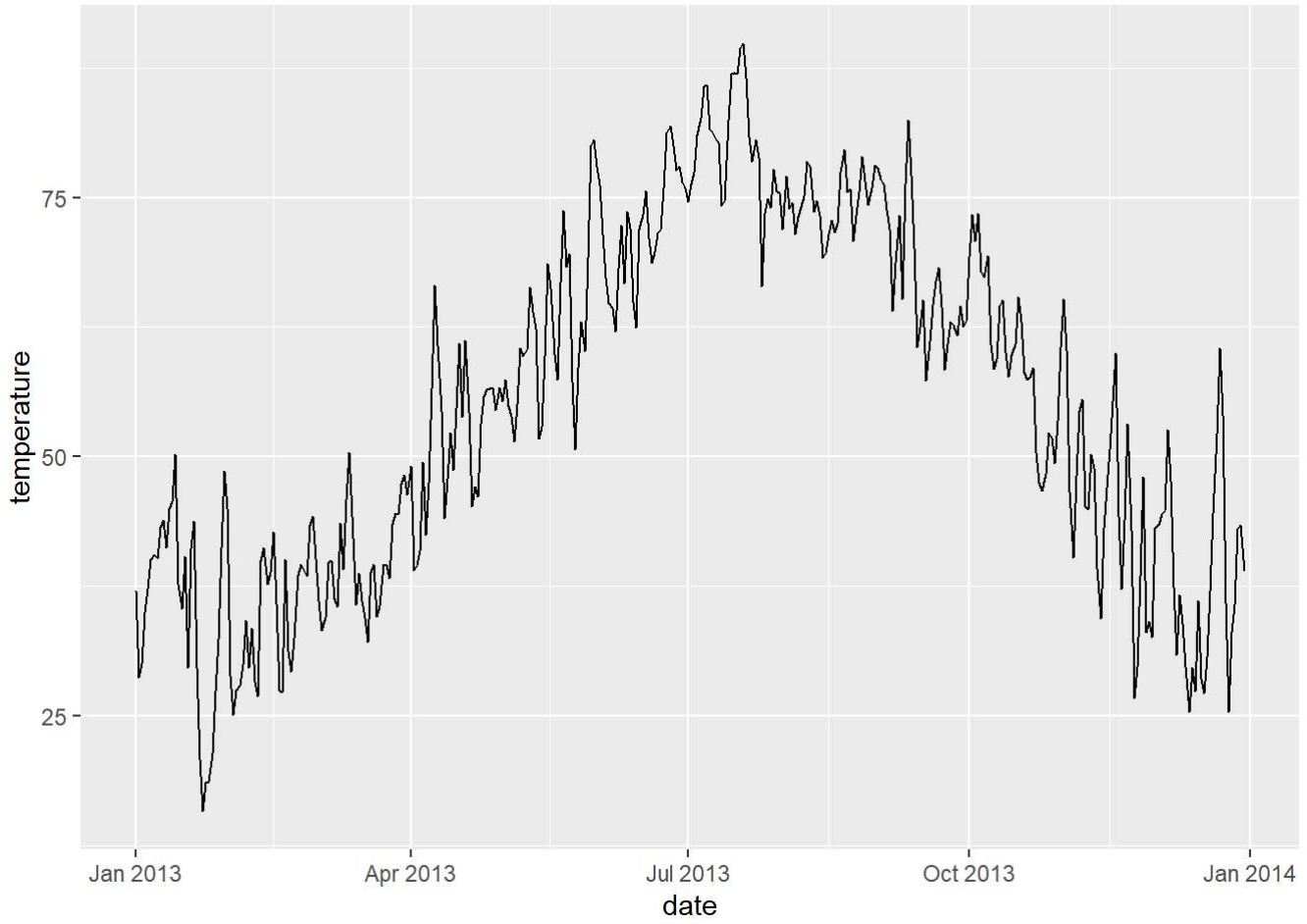
```
#Library(tidyverse)
library(modelr)
options(na.action = na.warn)

#Library(nycflights13)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

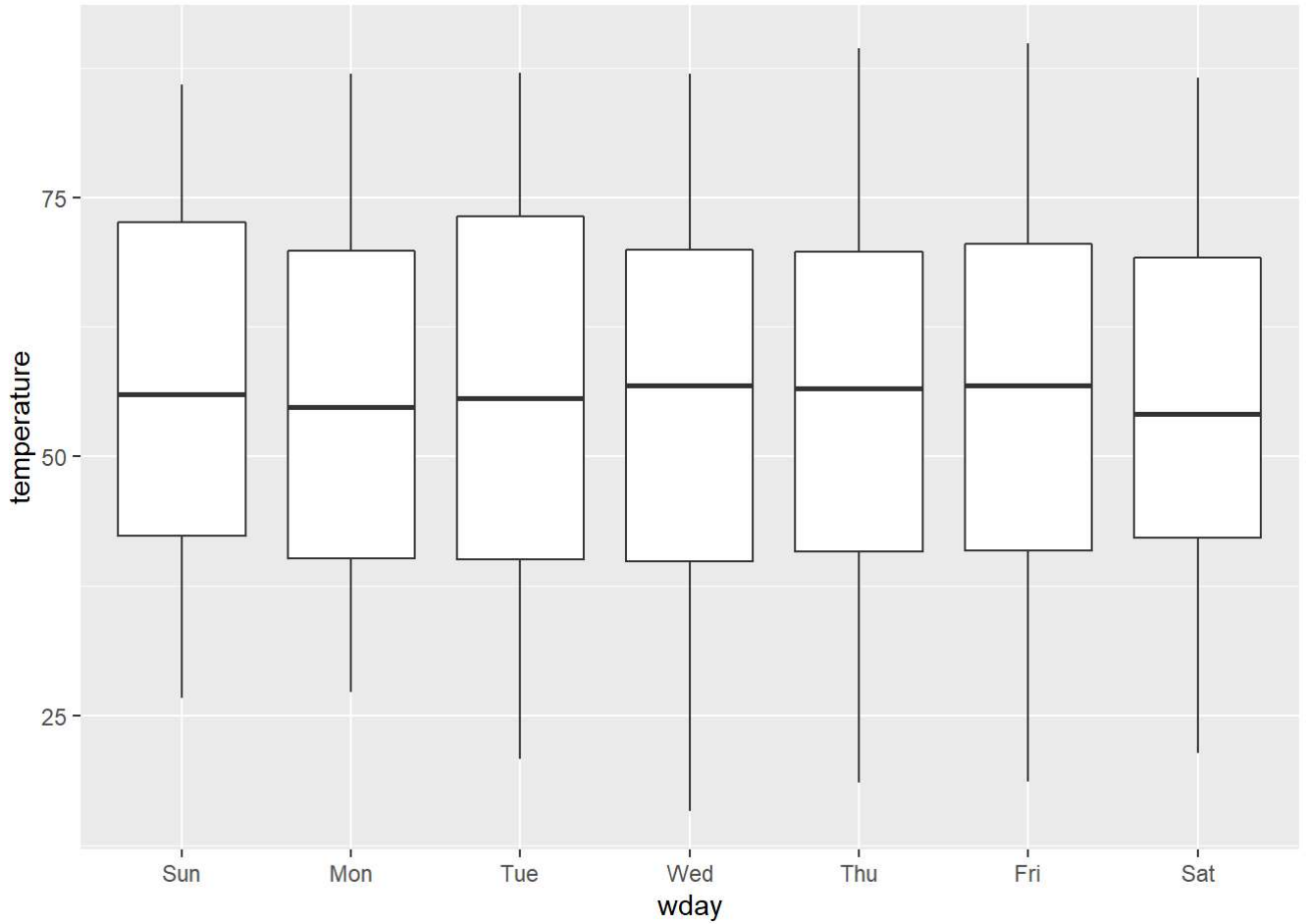
```
daily <- weather%>%
  mutate(date = make_date(year, month, day)) %>%
  group_by(date) %>%
  summarise(temperature=mean(temp,na.rm=TRUE))
```

```
ggplot(daily, aes(date, temperature)) +
  geom_line()
```



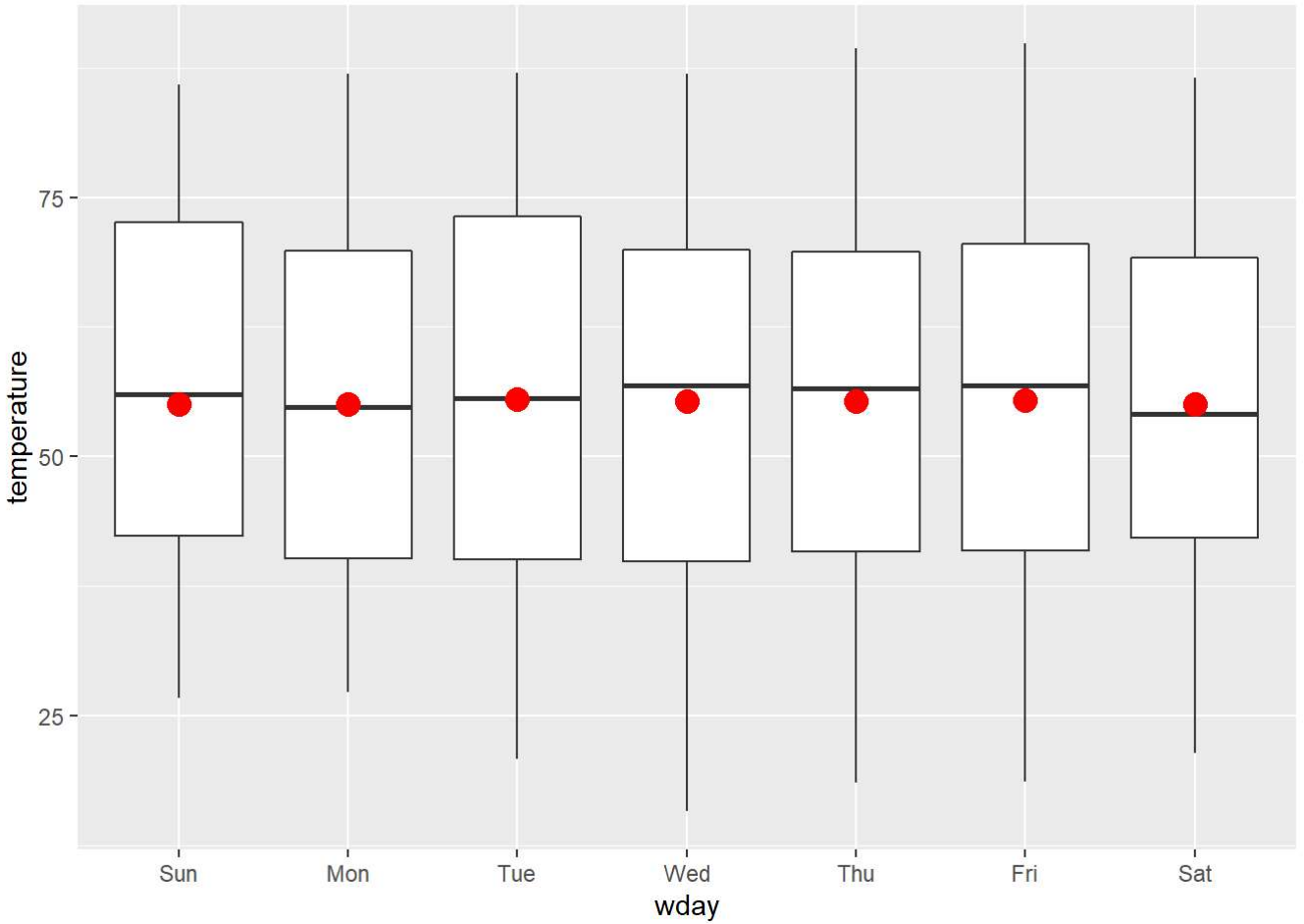
- I then visualized the distribution of the temperature levels by day of the week to see the shorter trend.From the graph below mondays and saturdays experienced lower temperatures

```
daily <- daily %>%
  mutate(wday = wday(date, label = TRUE))
ggplot(daily, aes(wday, temperature)) +
  geom_boxplot()
```



- I fitted a linear model and displayed its predictions overlaid on the original data

```
mod <- lm(temperature ~ wday, data = daily)

grid <- daily %>%
  data_grid(wday) %>%
  add_predictions(mod, "temperature")

ggplot(daily, aes(wday, temperature)) +
  geom_boxplot() +
  geom_point(data = grid, colour = "red", size = 4)
```

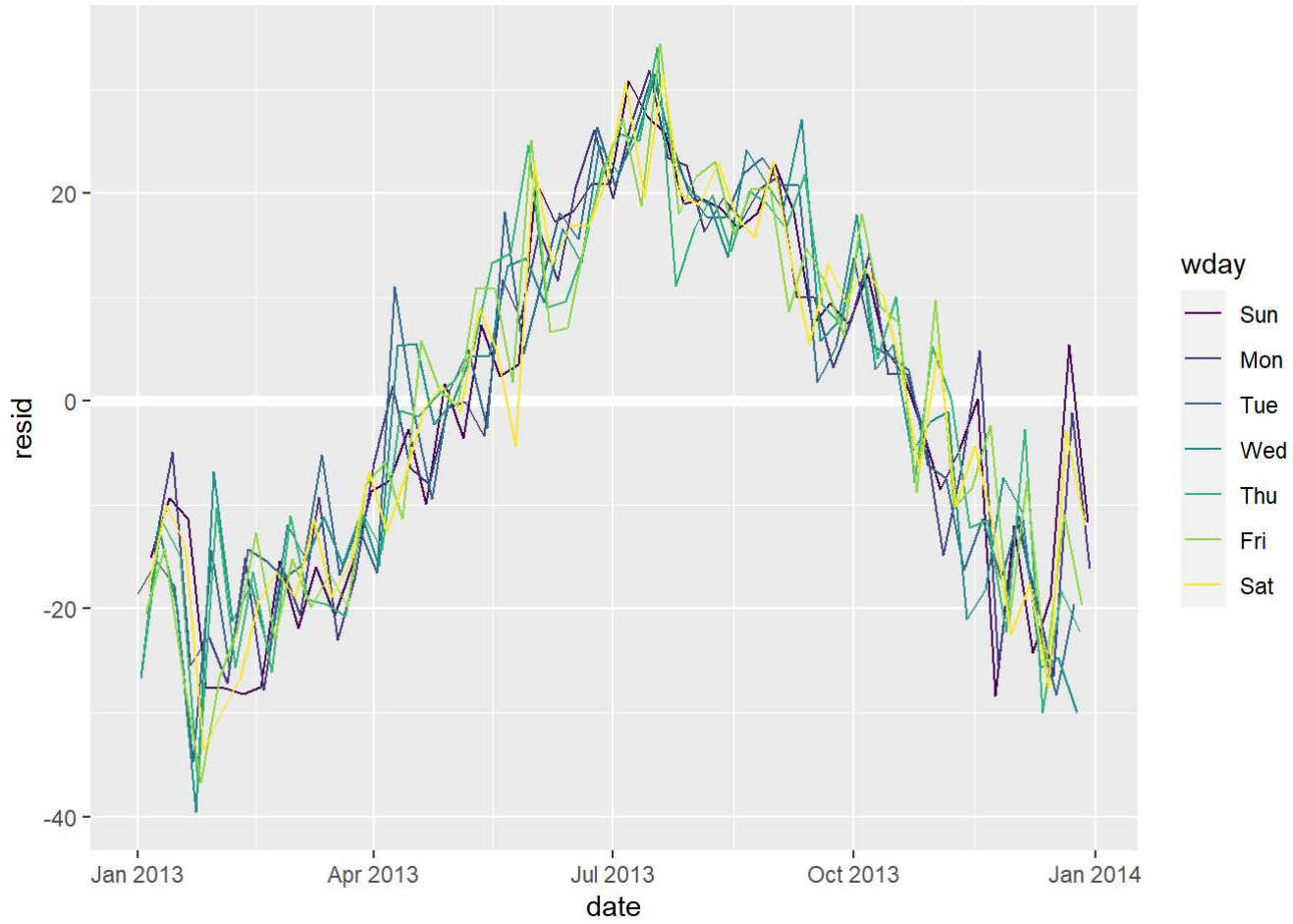


- Then i computed the residuals and visualized them

  .

  From the graph below the model seems to frail all through and hence a better option should be to try another family of models.Which i experienced difficulties doing

```
daily <- daily %>%
  add_residuals(mod)
daily %>%
  ggplot(aes(date, resid,colour=wday)) +
  geom_ref_line(h = 0) +
  geom_line()
```



4.To Use source documentation and other resources to troubleshoot and extend R programs

I achieved this objective by doing data transformations and handling relational databases using the dplr package and data from nycflights13 package. My analysis included;

- Filtering rows with filter():This allows to subset observations based on their values.
- Arranging rows with arrange():This changes the order of the data either ascending or descending provided the column names or other expressions to order by.
- Selecting columns with select():This is used to narrow down variables when working with datasets with hundreds or thousands of columns
- Adding new variables with mutate():This is used to add new columns that are functions of existing columns.

- Grouping and summarizing data with group_by and summarise():It changes the unit of analysis from the complete dataset to individual groups and then collapses the groups to a single row
- left_join():Allows to combine variables from two tables keeping all observations in x<>
- right_join():Allows to combine variables from two tables keeping all observation in y
- full_join
  :Allows to combine variables from two tables keeping all observations in x and y

- filter()

```
filter(flights,month==1)
```

```
## # A tibble: 27,004 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # ... with 26,994 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

- arrange()

```
arrange(flights,arr_time)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     2     2130           2130         0        1             18
##  2  2013     1    11     2157           2000       117        1           2208
##  3  2013     1    11     2253           2249         4        1           2357
##  4  2013     1    14     2122           2130        -8        1              2
##  5  2013     1    14     2246           2250        -4        1              7
##  6  2013     1    15     2304           2245        19        1           2357
##  7  2013     1    16     2018           2025        -7        1           2329
##  8  2013     1    16     2303           2245        18        1           2357
##  9  2013     1    19     2107           2110        -3        1           2355
## 10  2013     1    22     2246           2249        -3        1           2357
## # ... with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

- select()

```
select(flights,day,dep_time,arr_time)
```

```
## # A tibble: 336,776 x 3
##      day dep_time arr_time
##    <int>    <int>    <int>
##  1     1      517      830
##  2     1      533      850
##  3     1      542      923
##  4     1      544     1004
##  5     1      554      812
##  6     1      554      740
##  7     1      555      913
##  8     1      557      709
##  9     1      557      838
## 10     1      558      753
## # ... with 336,766 more rows
```

- mutate()

```
flights_sml <- select(flights,
  year:day,
  ends_with("delay"),
  distance,
  air_time
)
mutate(flights_sml,
  gain = dep_delay - arr_delay,
  speed = distance / air_time * 60
)
```

```
## # A tibble: 336,776 x 9
##     year month   day dep_delay arr_delay distance air_time  gain speed
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl> <dbl>
##  1  2013     1     1         2        11     1400      227    -9  370.
##  2  2013     1     1         4        20     1416      227   -16  374.
##  3  2013     1     1         2        33     1089      160   -31  408.
##  4  2013     1     1        -1       -18     1576      183    17  517.
##  5  2013     1     1        -6       -25      762      116    19  394.
##  6  2013     1     1        -4        12      719      150   -16  288.
##  7  2013     1     1        -5        19     1065      158   -24  404.
##  8  2013     1     1        -3       -14      229       53    11  259.
##  9  2013     1     1        -3        -8      944      140     5  405.
## 10  2013     1     1        -2         8      733      138   -10  319.
## # ... with 336,766 more rows
```

- group_by and summarise()

```
by_day <- group_by(flights, year, month, day)
summarise(by_day, delay = mean(dep_delay, na.rm = TRUE))
```

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

```
## # A tibble: 365 x 4
## # Groups:   year, month [12]
##     year month   day delay
##    <int> <int> <int> <dbl>
##  1  2013     1     1 11.5
##  2  2013     1     2 13.9
##  3  2013     1     3 11.0
##  4  2013     1     4  8.95
##  5  2013     1     5  5.73
##  6  2013     1     6  7.15
##  7  2013     1     7  5.42
##  8  2013     1     8  2.55
##  9  2013     1     9  2.28
## 10  2013     1    10  2.84
## # ... with 355 more rows
```

- left_join()

```
airlines<-nycflights13::airlines
flights %>%
  select(year,month,day,carrier) %>%
  left_join(airlines, by = "carrier")
```

```
## # A tibble: 336,776 x 5
##     year month   day carrier name
##    <int> <int> <int> <chr>   <chr>
##  1  2013     1     1 UA      United Air Lines Inc.
##  2  2013     1     1 UA      United Air Lines Inc.
##  3  2013     1     1 AA      American Airlines Inc.
##  4  2013     1     1 B6      JetBlue Airways
##  5  2013     1     1 DL      Delta Air Lines Inc.
##  6  2013     1     1 UA      United Air Lines Inc.
##  7  2013     1     1 B6      JetBlue Airways
##  8  2013     1     1 EV      ExpressJet Airlines Inc.
##  9  2013     1     1 B6      JetBlue Airways
## 10  2013     1     1 AA      American Airlines Inc.
## # ... with 336,766 more rows
```

- right_join()

```
flights%>%
  select(year,month,day,carrier)%>%
  right_join(airlines,by="carrier")
```

```
## # A tibble: 336,776 x 5
##     year month   day carrier name
##    <int> <int> <int> <chr>   <chr>
##  1  2013     1     1 UA      United Air Lines Inc.
##  2  2013     1     1 UA      United Air Lines Inc.
##  3  2013     1     1 AA      American Airlines Inc.
##  4  2013     1     1 B6      JetBlue Airways
##  5  2013     1     1 DL      Delta Air Lines Inc.
##  6  2013     1     1 UA      United Air Lines Inc.
##  7  2013     1     1 B6      JetBlue Airways
##  8  2013     1     1 EV      ExpressJet Airlines Inc.
##  9  2013     1     1 B6      JetBlue Airways
## 10  2013     1     1 AA      American Airlines Inc.
## # ... with 336,766 more rows
```

- full_join()

```
flights%>%
  full_join(airlines,by="carrier")
```

```
## # A tibble: 336,776 x 20
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # ... with 336,766 more rows, and 12 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>,
## #   name <chr>
```

5.Write clear,efficient and well documented R programs

- This objective was achieved through objective 1,2,3 and 4.
- All the codes were written in Rmarkdown file and then knitted to a html file.

Highlights

1. I experienced problems trying to fit other models,other than linear models
2. My most successful parts of the course have been.
    - Use of Rstudio, github and markdown.
    - Data visualization with with ggplot2.
    - Data transformation e.g pivoting.
    - use of tibbles,data import and tidying data.
    - Dealing with R packages.

- Challenges
    - Statistical modelling and simulation.
    - perfecting a shiny app.
      Generally i feel am at a good position in this class

Grade A