Homerwork 1

Brent Lewis

5/14/23

# Data Manipulation

## Problem 1: Use logical operators to find flights that:

- Had an arrival delay of two or more hours (\> 120 minutes)  
- Flew to Houston (IAH or HOU)  
- Were operated by United (`UA`), American (`AA`), or Delta (`DL`)  
- Departed in summer (July, August, and September)  
- Arrived more than two hours late, but didn't leave late  
- Were delayed by at least an hour, but made up over 30 minutes in flight

# Had an arrival delay of two or more hours (> 120 minutes)  
library(tidyverse)  
library(nycflights13)  
library(skimr)  
  
delay\_2\_hr <- flights %>%   
 filter(arr\_delay>"120")  
  
# Flew to Houston (IAH or HOU)  
houston\_flights <- flights %>%   
 filter(dest == "HOU"| dest =="IAH")  
  
houston\_flights

# A tibble: 9,313 × 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 1 623 627 -4 933 932  
 4 2013 1 1 728 732 -4 1041 1038  
 5 2013 1 1 739 739 0 1104 1038  
 6 2013 1 1 908 908 0 1228 1219  
 7 2013 1 1 1028 1026 2 1350 1339  
 8 2013 1 1 1044 1045 -1 1352 1351  
 9 2013 1 1 1114 900 134 1447 1222  
10 2013 1 1 1205 1200 5 1503 1505  
# ℹ 9,303 more rows  
# ℹ 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>

# Were operated by United (`UA`), American (`AA`), or Delta (`DL`)  
UA\_AA\_DL\_operated\_flights <- flights %>%   
 filter(carrier=="UA"|carrier=="AA"|carrier=="DL")  
  
UA\_AA\_DL\_operated\_flights

# A tibble: 139,504 × 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 1 554 600 -6 812 837  
 5 2013 1 1 554 558 -4 740 728  
 6 2013 1 1 558 600 -2 753 745  
 7 2013 1 1 558 600 -2 924 917  
 8 2013 1 1 558 600 -2 923 937  
 9 2013 1 1 559 600 -1 941 910  
10 2013 1 1 559 600 -1 854 902  
# ℹ 139,494 more rows  
# ℹ 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>

# Departed in summer (July, August, and September)  
summer\_flights <-flights %>%   
 filter(month=="7"|month=="8"|month=="9")  
summer\_flights

# A tibble: 86,326 × 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 7 1 1 2029 212 236 2359  
 2 2013 7 1 2 2359 3 344 344  
 3 2013 7 1 29 2245 104 151 1  
 4 2013 7 1 43 2130 193 322 14  
 5 2013 7 1 44 2150 174 300 100  
 6 2013 7 1 46 2051 235 304 2358  
 7 2013 7 1 48 2001 287 308 2305  
 8 2013 7 1 58 2155 183 335 43  
 9 2013 7 1 100 2146 194 327 30  
10 2013 7 1 100 2245 135 337 135  
# ℹ 86,316 more rows  
# ℹ 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>

# Arrived more than two hours late, but didn't leave late  
two\_hours\_late <-flights %>%   
 filter(arr\_delay >120) %>%   
 filter(dep\_delay <= 0)  
  
  
# Were delayed by at least an hour, but made up over 30 minutes in flight  
made\_up\_time <- flights %>%   
 filter(dep\_delay >= 60) %>%   
 filter(dep\_delay-arr\_delay>30)  
  
made\_up\_time

# A tibble: 1,844 × 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 2205 1720 285 46 2040  
 2 2013 1 1 2326 2130 116 131 18  
 3 2013 1 3 1503 1221 162 1803 1555  
 4 2013 1 3 1839 1700 99 2056 1950  
 5 2013 1 3 1850 1745 65 2148 2120  
 6 2013 1 3 1941 1759 102 2246 2139  
 7 2013 1 3 1950 1845 65 2228 2227  
 8 2013 1 3 2015 1915 60 2135 2111  
 9 2013 1 3 2257 2000 177 45 2224  
10 2013 1 4 1917 1700 137 2135 1950  
# ℹ 1,834 more rows  
# ℹ 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>

## Problem 2: What months had the highest and lowest proportion of cancelled flights? Interpret any seasonal patterns. To determine if a flight was cancelled use the following code

flights %>%   
 filter(is.na(dep\_time))

# What months had the highest and lowest % of cancelled flights?  
  
# Total number of cancelled flights, summarised as a new column and saved as a new tibble  
cancelled\_flights<- flights %>%   
 filter(is.na(dep\_time)) %>%   
 group\_by(month) %>%   
 summarise(cancelled\_flights\_count = n())  
  
cancelled\_flights

# A tibble: 12 × 2  
 month cancelled\_flights\_count  
 <int> <int>  
 1 1 521  
 2 2 1261  
 3 3 861  
 4 4 668  
 5 5 563  
 6 6 1009  
 7 7 940  
 8 8 486  
 9 9 452  
10 10 236  
11 11 233  
12 12 1025

# Total number of flights overall in dataset, summarised as a new column and saved as a new tibble  
total\_flights<- flights %>%   
 group\_by(month) %>%   
 summarise(total\_flights\_count = n())  
  
total\_flights

# A tibble: 12 × 2  
 month total\_flights\_count  
 <int> <int>  
 1 1 27004  
 2 2 24951  
 3 3 28834  
 4 4 28330  
 5 5 28796  
 6 6 28243  
 7 7 29425  
 8 8 29327  
 9 9 27574  
10 10 28889  
11 11 27268  
12 12 28135

# Proportion of total flights that are cancelled, calculated by left joining the cancelled\_flights tibble to the total\_flights tibble by the "month" variable (no issue with lost rows, since both tibbles have the same number of rows), creating a new "prop" variable to calculate the proportion of cancelled flights at each month and arranging from largest proportion to smallest  
  
cancelled\_flights\_prop <-total\_flights %>%   
 left\_join(cancelled\_flights, by = "month") %>%   
 mutate (prop = cancelled\_flights\_count / total\_flights\_count) %>%   
 arrange(desc(prop))  
  
cancelled\_flights\_prop

# A tibble: 12 × 4  
 month total\_flights\_count cancelled\_flights\_count prop  
 <int> <int> <int> <dbl>  
 1 2 24951 1261 0.0505   
 2 12 28135 1025 0.0364   
 3 6 28243 1009 0.0357   
 4 7 29425 940 0.0319   
 5 3 28834 861 0.0299   
 6 4 28330 668 0.0236   
 7 5 28796 563 0.0196   
 8 1 27004 521 0.0193   
 9 8 29327 486 0.0166   
10 9 27574 452 0.0164   
11 11 27268 233 0.00854  
12 10 28889 236 0.00817

# Intepretation: No months show especially high proportions of cancelled flights; February is the largest at 5%. We could infer that the higher proportions of cancelled flights towards the end of the calendar and academic/financial yeard are somehow related to travellers' changes of plans around holiday periods.

## Problem 3: What plane (specified by the tailnum variable) traveled the most times from New York City airports in 2013? Please left\_join() the resulting table with the table planes (also included in the nycflights13 package).

For the plane with the greatest number of flights and that had more than 50 seats, please create a table where it flew to during 2013.

skim(flights)

Data summary

|  |  |
| --- | --- |
| Name | flights |
| Number of rows | 336776 |
| Number of columns | 19 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 4 |
| numeric | 14 |
| POSIXct | 1 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

| skim\_variable | n\_missing | complete\_rate | min | max | empty | n\_unique | whitespace |
| --- | --- | --- | --- | --- | --- | --- | --- |
| carrier | 0 | 1.00 | 2 | 2 | 0 | 16 | 0 |
| tailnum | 2512 | 0.99 | 5 | 6 | 0 | 4043 | 0 |
| origin | 0 | 1.00 | 3 | 3 | 0 | 3 | 0 |
| dest | 0 | 1.00 | 3 | 3 | 0 | 105 | 0 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| year | 0 | 1.00 | 2013.00 | 0.00 | 2013 | 2013 | 2013 | 2013 | 2013 | ▁▁▇▁▁ |
| month | 0 | 1.00 | 6.55 | 3.41 | 1 | 4 | 7 | 10 | 12 | ▇▆▆▆▇ |
| day | 0 | 1.00 | 15.71 | 8.77 | 1 | 8 | 16 | 23 | 31 | ▇▇▇▇▆ |
| dep\_time | 8255 | 0.98 | 1349.11 | 488.28 | 1 | 907 | 1401 | 1744 | 2400 | ▁▇▆▇▃ |
| sched\_dep\_time | 0 | 1.00 | 1344.25 | 467.34 | 106 | 906 | 1359 | 1729 | 2359 | ▁▇▇▇▃ |
| dep\_delay | 8255 | 0.98 | 12.64 | 40.21 | -43 | -5 | -2 | 11 | 1301 | ▇▁▁▁▁ |
| arr\_time | 8713 | 0.97 | 1502.05 | 533.26 | 1 | 1104 | 1535 | 1940 | 2400 | ▁▃▇▇▇ |
| sched\_arr\_time | 0 | 1.00 | 1536.38 | 497.46 | 1 | 1124 | 1556 | 1945 | 2359 | ▁▃▇▇▇ |
| arr\_delay | 9430 | 0.97 | 6.90 | 44.63 | -86 | -17 | -5 | 14 | 1272 | ▇▁▁▁▁ |
| flight | 0 | 1.00 | 1971.92 | 1632.47 | 1 | 553 | 1496 | 3465 | 8500 | ▇▃▃▁▁ |
| air\_time | 9430 | 0.97 | 150.69 | 93.69 | 20 | 82 | 129 | 192 | 695 | ▇▂▂▁▁ |
| distance | 0 | 1.00 | 1039.91 | 733.23 | 17 | 502 | 872 | 1389 | 4983 | ▇▃▂▁▁ |
| hour | 0 | 1.00 | 13.18 | 4.66 | 1 | 9 | 13 | 17 | 23 | ▁▇▇▇▅ |
| minute | 0 | 1.00 | 26.23 | 19.30 | 0 | 8 | 29 | 44 | 59 | ▇▃▆▃▅ |

**Variable type: POSIXct**

| skim\_variable | n\_missing | complete\_rate | min | max | median | n\_unique |
| --- | --- | --- | --- | --- | --- | --- |
| time\_hour | 0 | 1 | 2013-01-01 05:00:00 | 2013-12-31 23:00:00 | 2013-07-03 10:00:00 | 6936 |

# This code assumes that Newark Airport is not a New York City airport   
  
# Create a new dataframe  
frequent\_flyer<-flights %>%  
# Filter in only LGA and JFK (i.e. exclude EWR)  
 filter(origin %in% c("LGA","JFK")) %>%  
# Group by tail number (since we are counting flights on the basis of tail number)  
 group\_by(tailnum) %>%   
# Generate a new column called tail\_num\_flights, which is just a count of the total flights per each distinct tail number in the frequent\_flyer dataframe  
 summarise(tail\_num\_flights = n()) %>%  
# Arrange in descending order (i.e. most flights at the top)   
 arrange(desc(tail\_num\_flights))  
  
print(frequent\_flyer)

# A tibble: 3,592 × 2  
 tailnum tail\_num\_flights  
 <chr> <int>  
 1 <NA> 1906  
 2 N725MQ 575  
 3 N722MQ 513  
 4 N723MQ 507  
 5 N711MQ 486  
 6 N713MQ 483  
 7 N735MQ 396  
 8 N328AA 393  
 9 N258JB 391  
10 N338AA 388  
# ℹ 3,582 more rows

# Left join the frequent\_flyer datafram to planes dataframe on the basis of tail number  
planes\_updated <-planes %>%   
 left\_join(frequent\_flyer, by = "tailnum") %>%   
 arrange(desc(tail\_num\_flights))  
  
print(planes\_updated)

# A tibble: 3,322 × 10  
 tailnum year type manufacturer model engines seats speed engine  
 <chr> <int> <chr> <chr> <chr> <int> <int> <int> <chr>   
 1 N711MQ 1976 Fixed wing multi… GULFSTREAM … G115… 2 22 NA Turbo…  
 2 N328AA 1986 Fixed wing multi… BOEING 767-… 2 255 NA Turbo…  
 3 N258JB 2006 Fixed wing multi… EMBRAER ERJ … 2 20 NA Turbo…  
 4 N338AA 1987 Fixed wing multi… BOEING 767-… 2 255 NA Turbo…  
 5 N327AA 1986 Fixed wing multi… BOEING 767-… 2 255 NA Turbo…  
 6 N335AA 1987 Fixed wing multi… BOEING 767-… 2 255 NA Turbo…  
 7 N298JB 2009 Fixed wing multi… EMBRAER ERJ … 2 20 NA Turbo…  
 8 N353JB 2012 Fixed wing multi… EMBRAER ERJ … 2 20 NA Turbo…  
 9 N351JB 2012 Fixed wing multi… EMBRAER ERJ … 2 20 NA Turbo…  
10 N228JB 2006 Fixed wing multi… EMBRAER ERJ … 2 20 NA Turbo…  
# ℹ 3,312 more rows  
# ℹ 1 more variable: tail\_num\_flights <int>

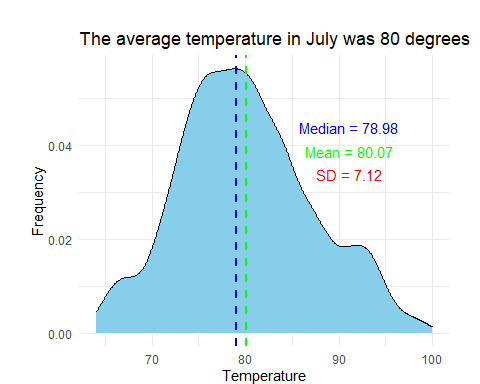
## Problem 4: The nycflights13 package includes a table (weather) that describes the weather during 2013. Use that table to answer the following questions:

- What is the distribution of temperature (`temp`) in July 2013? Identify any important outliers in terms of the `wind\_speed` variable.  
- What is the relationship between `dewp` and `humid`?  
- What is the relationship between `precip` and `visib`?

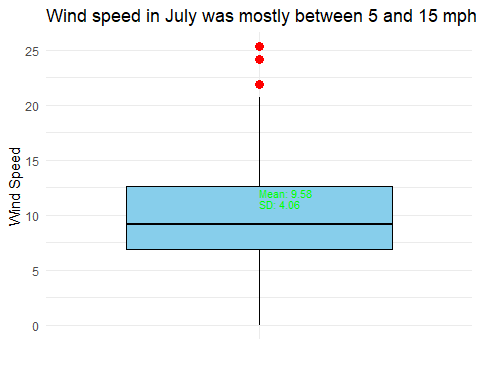
# Create a new tibble called july\_weather by filtering the only for month 7 (i.e. July) and filtering out non-recorded wind\_speed observations   
july\_weather <- filter(weather, month == 7 & !is.na(wind\_speed))  
  
# Create new tibbles for median, mean and standard deviation of the july\_weather temp and wind\_speed variables. We will use these later  
median\_temp <- median(july\_weather$temp)  
mean\_temp <- mean(july\_weather$temp)  
sd\_temp <- sd(july\_weather$temp)  
  
median\_wind\_speed <- median(july\_weather$wind\_speed)  
mean\_wind\_speed <- mean(july\_weather$wind\_speed)  
sd\_wind\_speed <- sd(july\_weather$wind\_speed)  
  
# To show the distribution of temperatures in july\_weather, create a density plot for temperatures, with temperature on the x-axis   
ggplot(july\_weather, aes(x = temp, y = ..density..)) +  
 geom\_density(fill = "skyblue", color = "black") +  
# Add dashed vertical lines for the median and mean temperatures  
 geom\_vline(aes(xintercept = median\_temp), color = "blue", linetype = "dashed", size = 1) +  
 geom\_vline(aes(xintercept = mean\_temp), color = "green", linetype = "dashed", size = 1) +  
#Write notes for each of the mean, median and standard deviation of temperature and line them up according to to the placement of the mean\_temp recording on the x-axis  
 annotate("text", x = mean\_temp + 11, y = 0.04, label = paste("Median =",round(median\_temp,2)), color = "blue", vjust = -1) +  
 annotate("text", x = mean\_temp + 11, y = 0.035, label = paste("Mean =",round(mean\_temp,2)), color = "green", vjust = -1) +  
 annotate("text", x = mean\_temp +11, y = 0.03, label = paste("SD =", round(sd\_temp, 2)), color = "red", vjust = -1) +  
# Label the x-axis "Temperature" and the y-axis "Frequency")   
 labs(x = "Temperature", y = "Frequency") +  
# Insert a title  
 ggtitle("The average temperature in July was 80 degrees") +  
 theme\_minimal() +  
# Insert appropriate margins to expand or contract the graph size  
 theme(plot.margin = margin(0.8, 0.8, 0, 0.8, "cm"))

Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
ℹ Please use `linewidth` instead.

Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.  
ℹ Please use `after\_stat(density)` instead.



# To identify outliers, create a box plot of the wind\_speed variable. Don't include anything on the x-axis, since we only need to show range in one variable and not necessarily over a specific period  
ggplot(july\_weather, aes(x = "", y = wind\_speed)) +  
 geom\_boxplot(fill = "skyblue", color = "black") +  
# To show outliers, add a geom\_point that filters in (from the wind\_speed variable of the july\_weather dataframe) only the wind\_speed observations in the 99.7th or 0.3rd percentiles (i.e. ~ 3 standard deviations from the mean) and colour these red to make them distinct  
 geom\_point(data = filter(july\_weather, wind\_speed > quantile(wind\_speed, 0.997) |  
 wind\_speed < quantile(wind\_speed, 0.003)),  
 aes(x = 1, y = wind\_speed), color = "red", shape = 16, size = 3) +  
 labs(x = "", y = "Wind Speed") +  
# Apply the same annotations as for the temp graph above  
 annotate("text", x = 1, y = mean\_wind\_speed + 2, label = paste("Mean:", round(mean\_wind\_speed, 2)),  
 color = "green", size = 3, hjust = 0, vjust = 0) +  
 annotate("text", x = 1, y = mean\_wind\_speed + 1, label = paste("SD:", round(sd\_wind\_speed, 2)),  
 color = "green", size = 3, hjust = 0, vjust = 0) +  
# Apply a title that provides the conclusion  
 ggtitle("Wind speed in July was mostly between 5 and 15 mph") +  
 theme\_minimal()

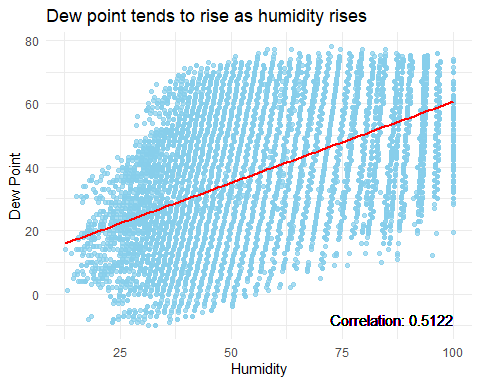


# To show the relationship between the humidity and dew point measure, first create a new tibble recording the correlation coefficient of the humid and dewp variables in the weather dataframe (note that this question doesn't ask us to filter specifically for July).  
  
correlation\_humid\_dewp <- cor(weather$humid, weather$dewp, use = "complete.obs")   
  
# Use only complete observations (i.e. nothing recorded as "NA")  
  
# Map this relationship using a scatter plot, with humidity on the x-axis and dew point measurement on the y-axis  
ggplot(weather, aes(x = humid, y = dewp)) +  
# Since some of the points overlap, set the alpha (i.e. transparency) slightly above half (0.7 should do) to show overlapped points that don't distort the image  
 geom\_point(color = "skyblue", alpha = 0.7) +  
# Insert a line of best fit to demonstrate the relationship between the variables  
 geom\_smooth(method = "lm", color = "red", se = FALSE) +  
# Insert an annotation to show the correlation coefficient  
 geom\_text(x = max(weather$humid, na.rm = TRUE), y = min(weather$dewp, na.rm = TRUE), label = paste("Correlation:", round(correlation\_humid\_dewp, 4)), hjust = 1, vjust = 0, color = "black", size = 4) +  
#Insert axis labels   
 labs(x = "Humidity", y = "Dew Point") +  
# Insert a title  
 ggtitle("Dew point tends to rise as humidity rises") +  
 theme\_minimal()

`geom\_smooth()` using formula = 'y ~ x'

Warning: Removed 1 rows containing non-finite values (`stat\_smooth()`).

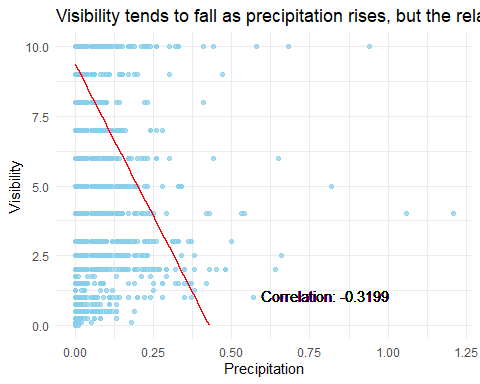
Warning: Removed 1 rows containing missing values (`geom\_point()`).



# To show the relationship between precipitation and visibility, perform the exact same operation as for humidity and precipitation, changing the variables as necessary  
correlation\_precip\_visib <- cor(weather$precip, weather$visib, use = "complete.obs")  
  
ggplot(weather, aes(x = precip, y = visib)) +  
 geom\_point(color = "skyblue", alpha = 0.7) +  
 geom\_smooth(method = "lm", color = "red", se = FALSE) +  
 geom\_text(x = max(weather$precip, na.rm = TRUE), y = min(weather$visib, na.rm = TRUE),  
 label = paste("Correlation:", round(correlation\_precip\_visib, 4)), hjust = 1.5, vjust = -2,  
 color = "black", size = 4) +  
 labs(x = "Precipitation", y = "Visibility") +  
 ggtitle("Visibility tends to fall as precipitation rises, but the relationship is not strong") +  
 theme\_minimal()+  
# Set the lower limit on the y-axis to zero so ensure the image isn't distorted.  
 ylim(0,NA)

`geom\_smooth()` using formula = 'y ~ x'

Warning: Removed 51 rows containing missing values (`geom\_smooth()`).



# Notably, the graph doesn't show a particularly compelling relationship between the two variables, in spite of the correlation coefficient, which shows some relationship. This could be because the visibility variable is divided into different bins

## Problem 5: Use the flights and planes tables to answer the following questions:

- How many planes have a missing date of manufacture?  
- What are the five most common manufacturers?  
- Has the distribution of manufacturer changed over time as reflected by the airplanes flying from NYC in 2013? (Hint: you may need to use case\_when() to recode the manufacturer name and collapse rare vendors into a category called Other.)

# To count how many planes have a missing date of manufacture, simply filter the planes year (of manufacture) variable in the planes dataframe for all "NA" results  
planes %>%   
 filter(is.na(year)) %>%   
 count()

# A tibble: 1 × 1  
 n  
 <int>  
1 70

# To find the five most common manufacturers, create a new dataframe from planes which is grouped by manufacturer (since we will be sorting by this variable)  
manufacturer\_count <- planes %>%   
 group\_by(manufacturer) %>%  
# Recode the observations in the manufacturer column to align the spelling of inconsistentently recorded manufacturer titles  
 mutate(manufacturer=recode(manufacturer,"AIRBUS INDUSTRIE" = "AIRBUS","MCDONNELL DOUGLAS AIRCRAFT CO" = "MCDONNELL DOUGLAS","MCDONNELL DOUGLAS CORPORATION" = "MCDONNELL DOUGLAS")) %>%   
# Create another column counting the number of manufacturers  
 summarise(count=n()) %>%   
# Arrange the tibble in descending order by this count   
 arrange(desc(count))  
  
# Using the manufacturer\_count tibble we just created, create a list of the top  
top\_manufacturers <- top\_n(manufacturer\_count, 5, count)  
  
print(top\_manufacturers)

# A tibble: 5 × 2  
 manufacturer count  
 <chr> <int>  
1 BOEING 1630  
2 AIRBUS 736  
3 BOMBARDIER INC 368  
4 EMBRAER 299  
5 MCDONNELL DOUGLAS 237

# We can see that Boeing and Airbus are the leading manufacturers  
  
  
# To show how the distribution of manufacturers has changed over time as reflected by the airplanes flying from NYC in 2013, we will show how this distribution changes on a day-by-day basis for 2013  
  
# First import the lubridate library to allow us to manipulate date settings in the flights library  
library(lubridate)  
  
# Then create a new variable in this tibble called day\_of\_year that records the specific day of the year (between 1 and 365) based on the available year, month and day variables already provided  
flights\_updated <- flights %>%  
 mutate(day\_of\_year = yday(ymd(paste(year, month, day, sep = "-"))))  
  
# Using this new tibble, filter out cancelled flights  
completed\_flights<-flights\_updated %>%   
 filter(!is.na(dep\_time))  
  
# Right join the planes dataframe to the completed\_flights data frame, using a right join so that individual flight records are not lost. In doing this, we select only the tailnum, manufacturer and year variables form the planes tibble and change the title of year to build\_year to avoid a clash with the existing variable in flights  
flights\_with\_manufacturers <-right\_join(completed\_flights,planes %>%   
 select(tailnum, manufacturer, build\_year=year) %>%   
# Perform the same renaming mutation as above  
 mutate(manufacturer=recode(manufacturer,"AIRBUS INDUSTRIE" = "AIRBUS","MCDONNELL DOUGLAS AIRCRAFT CO" = "MCDONNELL DOUGLAS","MCDONNELL DOUGLAS CORPORATION" = "MCDONNELL DOUGLAS")), by = "tailnum")  
  
# From this tibble, create another tibble that mutates all observations in the manufacturer variable. By referring to the top\_manufacturers list we created above, we identify anything that this not in that list as "Other"  
updated\_flights\_with\_manufacturers <- flights\_with\_manufacturers %>%  
 mutate(manufacturer = case\_when(  
 manufacturer %in% top\_manufacturers$manufacturer ~ manufacturer,  
 TRUE ~ "Other")) %>%   
# Group by day\_of\_year first and then by manufacturer, since we will be sorting by both in that order, and provide a count of each flight  
 group\_by(day\_of\_year,manufacturer) %>%   
 summarise(count=n())

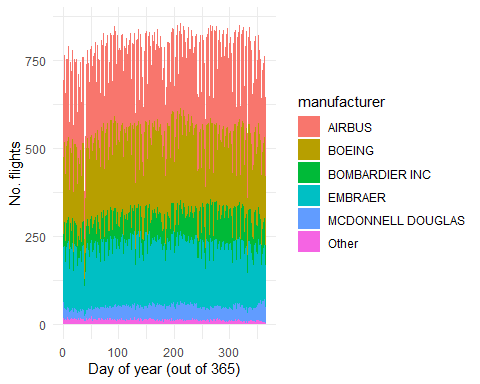
`summarise()` has grouped output by 'day\_of\_year'. You can override using the  
`.groups` argument.

print(updated\_flights\_with\_manufacturers)

# A tibble: 2,193 × 3  
# Groups: day\_of\_year [366]  
 day\_of\_year manufacturer count  
 <dbl> <chr> <int>  
 1 1 AIRBUS 219  
 2 1 BOEING 220  
 3 1 BOMBARDIER INC 36  
 4 1 EMBRAER 158  
 5 1 MCDONNELL DOUGLAS 41  
 6 1 Other 20  
 7 2 AIRBUS 245  
 8 2 BOEING 242  
 9 2 BOMBARDIER INC 61  
10 2 EMBRAER 176  
# ℹ 2,183 more rows

# Plot this tibble as an area graph, with the day of the year on the x-axis showing how the distribution of flights has changed on a daily basis throughout 2013 among the manufacturers  
ggplot(updated\_flights\_with\_manufacturers, aes(x = day\_of\_year, y = count, fill = manufacturer, group = manufacturer)) +  
 geom\_area() +  
 labs(x = "Day of year (out of 365)", y = "No. flights") +  
 theme\_minimal()

Warning: Removed 3 rows containing non-finite values (`stat\_align()`).



## Problem 6: Use the flights and planes tables to answer the following questions:

- What is the oldest plane (specified by the tailnum variable) that flew from New York City airports in 2013?  
- How many airplanes that flew from New York City are included in the planes table?

# To see the oldest plane, using the flights\_with\_manufacturers tibble we created earlier, filter out all results where no build year is recorded  
oldest\_plane <- flights\_with\_manufacturers %>%  
 filter(!is.na(build\_year)) %>%  
# Then sort by build year (ascending, so oldest first)  
 arrange(build\_year) %>%   
# Then take only the first result  
 slice(1)  
  
# Print out only the tailnum column from this resulting tibble  
print(oldest\_plane["tailnum"])

# A tibble: 1 × 1  
 tailnum  
 <chr>   
1 N381AA

# To see how many airplanes that flew from NYC are included in the planes table, first create a new tibble from the flights\_with\_manufacturers tibble we created earlier  
total\_distinct\_planes<-flights\_with\_manufacturers %>%   
# Filter out non-recored build years and tail numbers   
 filter(!is.na(build\_year)&!is.na(tailnum)) %>%  
# Record only distinct tail numbers (i.e. don't permit duplicates)  
 distinct(tailnum) %>%   
# Take a count of the total number of rows  
 summarise(total\_distinct\_planes=n())  
  
print(total\_distinct\_planes)

# A tibble: 1 × 1  
 total\_distinct\_planes  
 <int>  
1 3252

## Problem 7: Use the nycflights13 to answer the following questions:

- What is the median arrival delay on a month-by-month basis in each airport?  
- For each airline, plot the median arrival delay for each month and origin airport.

# To find the median arrival delay on a month-by-month basis in each airport, first find the median arrival delay from the flights table  
median\_arrival\_delay <- flights %>%  
# Filter out cancelled flights  
 filter(!is.na(dep\_time)) %>%  
# Group by month and then destination, since we are sorting by these variables  
 group\_by(month, dest) %>%  
# Create a new column capturing the median arrival delay and filtering out non-recorded results  
 summarise(median\_\_arr\_delay = median(arr\_delay, na.rm = TRUE)) %>%  
 arrange(dest, month)

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

print(median\_arrival\_delay)

# A tibble: 1,112 × 3  
# Groups: month [12]  
 month dest median\_\_arr\_delay  
 <int> <chr> <dbl>  
 1 4 ABQ 14   
 2 5 ABQ -19   
 3 6 ABQ -2.5  
 4 7 ABQ -6   
 5 8 ABQ -14   
 6 9 ABQ -16   
 7 10 ABQ -10   
 8 11 ABQ -6   
 9 12 ABQ 27   
10 5 ACK -8   
# ℹ 1,102 more rows

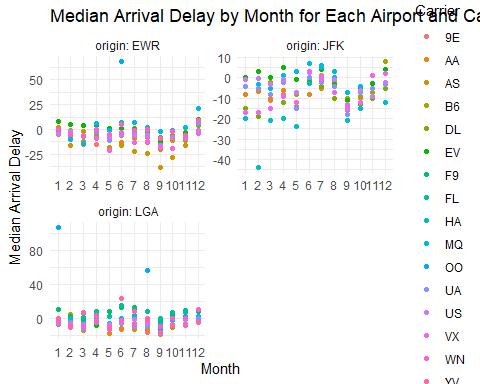
# The below is another way of arranging this data by grouping by month  
flights %>%  
 filter(!is.na(arr\_delay)) %>%  
 group\_by(month, dest) %>%  
 summarise(median\_arr\_delay = median(arr\_delay, na.rm = TRUE)) %>%  
 arrange(month, dest)

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

# A tibble: 1,112 × 3  
# Groups: month [12]  
 month dest median\_arr\_delay  
 <int> <chr> <dbl>  
 1 1 ALB 6   
 2 1 ATL -2   
 3 1 AUS -2   
 4 1 AVL 23.5  
 5 1 BDL -10   
 6 1 BHM -11   
 7 1 BNA 1   
 8 1 BOS -10   
 9 1 BQN -5   
10 1 BTV -6   
# ℹ 1,102 more rows

# To plot the median arrival delay for each month and origin airport, first filter out all results where no arrival delay observation is recorded  
flights %>%  
 filter(!is.na(arr\_delay)) %>%  
# Group by the variables by which we will be arranging the data   
 group\_by(origin, month, carrier) %>%  
# Calculate the median arrival delay in the same way as above  
 summarise(median\_arr\_delay = median(arr\_delay, na.rm = TRUE)) %>%  
# Plot the results on a scatter plot and facet wrap for origin airport for simplicity, with different colours for each carrier and each graph recording the months on the x-axis  
 ggplot() +  
 geom\_point(aes(x = factor(month), y = median\_arr\_delay, color = carrier)) +  
 facet\_wrap(vars(origin), nrow = 2, scales = "free", labeller = label\_both) +  
 labs(x = "Month", y = "Median Arrival Delay", color = "Carrier") +  
 ggtitle("Median Arrival Delay by Month for Each Airport and Carrier") +  
 theme\_minimal()

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



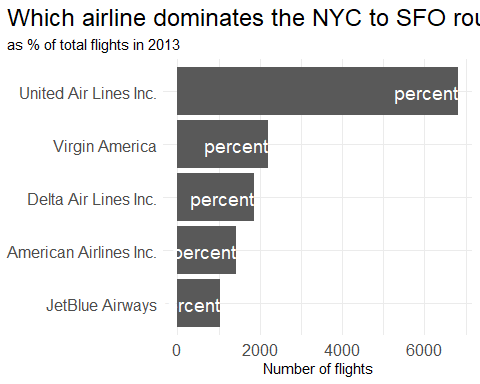
## Problem 8: Let’s take a closer look at what carriers service the route to San Francisco International (SFO). Join the flights and airlines tables and count which airlines flew the most to SFO. Produce a new dataframe, fly\_into\_sfo that contains three variables: the name of the airline, e.g., United Air Lines Inc. not UA, the count (number) of times it flew to SFO, and the percent of the trips that that particular airline flew to SFO.

fly\_into\_sfo<- flights %>%  
# Left join the airlines tibble to the flights tibble by the carrier variable  
 left\_join(airlines,by="carrier") %>%   
# Filter in only SFO as the destination variable  
 filter(dest == "SFO") %>%  
# Group by name, since the new dataframe will count the number of flights for each name   
 group\_by(name) %>%  
# Count the number of flights for each name, recording this as "count"  
 summarise(count = n()) %>%  
# Create a new column that records the count of each airline as a percentage of total flights to SFO  
 mutate(percent\_to\_sfo = count / sum(count) \* 100)  
  
fly\_into\_sfo

# A tibble: 5 × 3  
 name count percent\_to\_sfo  
 <chr> <int> <dbl>  
1 American Airlines Inc. 1422 10.7   
2 Delta Air Lines Inc. 1858 13.9   
3 JetBlue Airways 1035 7.76  
4 United Air Lines Inc. 6819 51.2   
5 Virgin America 2197 16.5

And here is some bonus ggplot code to plot your dataframe

fly\_into\_sfo %>%   
   
 # sort 'name' of airline by the numbers it times to flew to SFO  
 mutate(name = fct\_reorder(name, count)) %>%   
   
 ggplot() +  
   
 aes(x = count,   
 y = name) +  
   
 # a simple bar/column plot  
 geom\_col() +  
   
 # add labels, so each bar shows the % of total flights   
 geom\_text(aes(label = "percent"),  
 hjust = 1,   
 colour = "white",   
 size = 5)+  
   
 # add labels to help our audience   
 labs(title="Which airline dominates the NYC to SFO route?",   
 subtitle = "as % of total flights in 2013",  
 x= "Number of flights",  
 y= NULL) +  
   
 theme\_minimal() +   
   
 # change the theme-- i just googled those , but you can use the ggThemeAssist add-in  
 # https://cran.r-project.org/web/packages/ggThemeAssist/index.html  
   
 theme(#  
 # so title is left-aligned  
 plot.title.position = "plot",  
   
 # text in axes appears larger   
 axis.text = element\_text(size=12),  
   
 # title text is bigger  
 plot.title = element\_text(size=18)  
 ) +  
  
 # add one final layer of NULL, so if you comment out any lines  
 # you never end up with a hanging `+` that awaits another ggplot layer  
 NULL



## Problem 9: Let’s take a look at cancellations of flights to SFO. We create a new dataframe cancellations as follows

cancellations <- flights %>%   
   
 # just filter for destination == 'SFO'  
 filter(dest == 'SFO') %>%   
   
 # a cancelled flight is one with no `dep\_time`   
 filter(is.na(dep\_time))  
  
  
# First I would create a new dataframe that filters for only Newark and JFK as departure destinations, or filters out Laguardia. I would group this by month and then carrier. Then I would generate a geom\_bar plot facet wrapped first by origin and then by carrier

I want you to think how we would organise our data manipulation to create the following plot. No need to write the code, just explain in words how you would go about it.

## Problem 10: On your own – Hollywood Age Gap

The website <https://hollywoodagegap.com> is a record of *THE AGE DIFFERENCE IN YEARS BETWEEN MOVIE LOVE INTERESTS*. This is an informational site showing the age gap between movie love interests and the data follows certain rules:

* The two (or more) actors play actual love interests (not just friends, coworkers, or some other non-romantic type of relationship)
* The youngest of the two actors is at least 17 years old
* No animated characters

The age gaps dataset includes “gender” columns, which always contain the values “man” or “woman”. These values appear to indicate how the characters in each film identify and some of these values do not match how the actor identifies. We apologize if any characters are misgendered in the data!

The following is a data dictionary of the variables used

| variable | class | description |
| --- | --- | --- |
| movie\_name | character | Name of the film |
| release\_year | integer | Release year |
| director | character | Director of the film |
| age\_difference | integer | Age difference between the characters in whole years |
| couple\_number | integer | An identifier for the couple in case multiple couples are listed for this film |
| actor\_1\_name | character | The name of the older actor in this couple |
| actor\_2\_name | character | The name of the younger actor in this couple |
| character\_1\_gender | character | The gender of the older character, as identified by the person who submitted the data for this couple |
| character\_2\_gender | character | The gender of the younger character, as identified by the person who submitted the data for this couple |
| actor\_1\_birthdate | date | The birthdate of the older member of the couple |
| actor\_2\_birthdate | date | The birthdate of the younger member of the couple |
| actor\_1\_age | integer | The age of the older actor when the film was released |
| actor\_2\_age | integer | The age of the younger actor when the film was released |

age\_gaps <- readr::read\_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2023/2023-02-14/age\_gaps.csv')

Rows: 1155 Columns: 13  
── Column specification ────────────────────────────────────────────────────────  
Delimiter: ","  
chr (6): movie\_name, director, actor\_1\_name, actor\_2\_name, character\_1\_gend...  
dbl (5): release\_year, age\_difference, couple\_number, actor\_1\_age, actor\_2\_age  
date (2): actor\_1\_birthdate, actor\_2\_birthdate  
  
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.

How would you explore this data set? Here are some ideas of tables/ graphs to help you with your analysis

* How is age\_difference distributed? What’s the ‘typical’ age\_difference in movies?
* The half plus seven\ rule. Large age disparities in relationships carry certain stigmas. One popular rule of thumb is the [half-your-age-plus-seven](https://en.wikipedia.org/wiki/Age_disparity_in_sexual_relationships#The_.22half-your-age-plus-seven.22_rule) rule. This rule states you should never date anyone under half your age plus seven, establishing a minimum boundary on whom one can date. In order for a dating relationship to be acceptable under this rule, your partner’s age must be:

$$\frac{\text{Your age}}{2} + 7 \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\< \text{Partner Age} \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\< (\text{Your age} - 7) \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\* 2$$

How frequently does this rule apply in this dataset?

* Which movie has the greatest number of love interests?
* Which actors/ actresses have the greatest number of love interests in this dataset?
* Is the mean/median age difference staying constant over the years (1935 - 2022)?
* How frequently does Hollywood depict same-gender love interests?

# Deliverables

There is a lot of explanatory text, comments, etc. You do not need these, so delete them and produce a stand-alone document that you could share with someone. Render the edited and completed Quarto Markdown (qmd) file as a Word document (use the “Render” button at the top of the script editor window) and upload it to Canvas. You must be commiting and pushing tour changes to your own Github repo as you go along.

# Details

* Who did you collaborate with: TYPE NAMES HERE
* Approximately how much time did you spend on this problem set: ANSWER HERE
* What, if anything, gave you the most trouble: ANSWER HERE

**Please seek out help when you need it,** and remember the [15-minute rule](https://mam2022.netlify.app/syllabus/#the-15-minute-rule). You know enough R (and have enough examples of code from class and your readings) to be able to do this. If you get stuck, ask for help from others, post a question on Slack– and remember that I am here to help too!

As a true test to yourself, do you understand the code you submitted and are you able to explain it to someone else?

# Rubric

13/13: Problem set is 100% completed. Every question was attempted and answered, and most answers are correct. Code is well-documented (both self-documented and with additional comments as necessary). Used tidyverse, instead of base R. Graphs and tables are properly labelled. Analysis is clear and easy to follow, either because graphs are labeled clearly or you’ve written additional text to describe how you interpret the output. Multiple Github commits. Work is exceptional. I will not assign these often.

8/13: Problem set is 60–80% complete and most answers are correct. This is the expected level of performance. Solid effort. Hits all the elements. No clear mistakes. Easy to follow (both the code and the output). A few Github commits.

5/13: Problem set is less than 60% complete and/or most answers are incorrect. This indicates that you need to improve next time. I will hopefully not assign these often. Displays minimal effort. Doesn’t complete all components. Code is poorly written and not documented. Uses the same type of plot for each graph, or doesn’t use plots appropriate for the variables being analyzed. No Github commits.