Homerwork 1

Edward Davies

2023-05-14

# Data Manipulation

All information below has been committed to my own github folder: https://github.com/edwardhdavies/mydsb2023

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

# Had an arrival delay of two or more hours (> 120 minutes)  
summary(flights)

year month day dep\_time sched\_dep\_time  
 Min. :2013 Min. : 1.000 Min. : 1.00 Min. : 1 Min. : 106   
 1st Qu.:2013 1st Qu.: 4.000 1st Qu.: 8.00 1st Qu.: 907 1st Qu.: 906   
 Median :2013 Median : 7.000 Median :16.00 Median :1401 Median :1359   
 Mean :2013 Mean : 6.549 Mean :15.71 Mean :1349 Mean :1344   
 3rd Qu.:2013 3rd Qu.:10.000 3rd Qu.:23.00 3rd Qu.:1744 3rd Qu.:1729   
 Max. :2013 Max. :12.000 Max. :31.00 Max. :2400 Max. :2359   
 NA's :8255   
 dep\_delay arr\_time sched\_arr\_time arr\_delay   
 Min. : -43.00 Min. : 1 Min. : 1 Min. : -86.000   
 1st Qu.: -5.00 1st Qu.:1104 1st Qu.:1124 1st Qu.: -17.000   
 Median : -2.00 Median :1535 Median :1556 Median : -5.000   
 Mean : 12.64 Mean :1502 Mean :1536 Mean : 6.895   
 3rd Qu.: 11.00 3rd Qu.:1940 3rd Qu.:1945 3rd Qu.: 14.000   
 Max. :1301.00 Max. :2400 Max. :2359 Max. :1272.000   
 NA's :8255 NA's :8713 NA's :9430   
 carrier flight tailnum origin   
 Length:336776 Min. : 1 Length:336776 Length:336776   
 Class :character 1st Qu.: 553 Class :character Class :character   
 Mode :character Median :1496 Mode :character Mode :character   
 Mean :1972   
 3rd Qu.:3465   
 Max. :8500   
   
 dest air\_time distance hour   
 Length:336776 Min. : 20.0 Min. : 17 Min. : 1.00   
 Class :character 1st Qu.: 82.0 1st Qu.: 502 1st Qu.: 9.00   
 Mode :character Median :129.0 Median : 872 Median :13.00   
 Mean :150.7 Mean :1040 Mean :13.18   
 3rd Qu.:192.0 3rd Qu.:1389 3rd Qu.:17.00   
 Max. :695.0 Max. :4983 Max. :23.00   
 NA's :9430   
 minute time\_hour   
 Min. : 0.00 Min. :2013-01-01 05:00:00.00   
 1st Qu.: 8.00 1st Qu.:2013-04-04 13:00:00.00   
 Median :29.00 Median :2013-07-03 10:00:00.00   
 Mean :26.23 Mean :2013-07-03 05:22:54.64   
 3rd Qu.:44.00 3rd Qu.:2013-10-01 07:00:00.00   
 Max. :59.00 Max. :2013-12-31 23:00:00.00

flights %>%   
 filter(arr\_delay >= 120)

# A tibble: 10,200 × 19  
 year month day dep\_time sched\_de…¹ dep\_d…² arr\_t…³ sched…⁴ arr\_d…⁵ carrier  
 <int> <int> <int> <int> <int> <dbl> <int> <int> <dbl> <chr>   
 1 2013 1 1 811 630 101 1047 830 137 MQ   
 2 2013 1 1 848 1835 853 1001 1950 851 MQ   
 3 2013 1 1 957 733 144 1056 853 123 UA   
 4 2013 1 1 1114 900 134 1447 1222 145 UA   
 5 2013 1 1 1505 1310 115 1638 1431 127 EV   
 6 2013 1 1 1525 1340 105 1831 1626 125 B6   
 7 2013 1 1 1549 1445 64 1912 1656 136 EV   
 8 2013 1 1 1558 1359 119 1718 1515 123 EV   
 9 2013 1 1 1732 1630 62 2028 1825 123 EV   
10 2013 1 1 1803 1620 103 2008 1750 138 MQ   
# … with 10,190 more rows, 9 more variables: flight <int>, tailnum <chr>,  
# origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
# minute <dbl>, time\_hour <dttm>, and abbreviated variable names  
# ¹​sched\_dep\_time, ²​dep\_delay, ³​arr\_time, ⁴​sched\_arr\_time, ⁵​arr\_delay

# Flew to Houston (IAH or HOU)  
  
flights %>%   
 filter(dest == "TAH" | dest == "HOU")

# A tibble: 2,115 × 19  
 year month day dep\_time sched\_de…¹ dep\_d…² arr\_t…³ sched…⁴ arr\_d…⁵ carrier  
 <int> <int> <int> <int> <int> <dbl> <int> <int> <dbl> <chr>   
 1 2013 1 1 1208 1158 10 1540 1502 38 B6   
 2 2013 1 1 1306 1300 6 1622 1610 12 WN   
 3 2013 1 1 1708 1700 8 2037 2005 32 WN   
 4 2013 1 1 2030 2035 -5 2354 2342 12 B6   
 5 2013 1 2 734 700 34 1045 1025 20 WN   
 6 2013 1 2 1156 1158 -2 1517 1502 15 B6   
 7 2013 1 2 1319 1305 14 1633 1615 18 WN   
 8 2013 1 2 1810 1655 75 2146 2000 106 WN   
 9 2013 1 2 2031 2035 -4 2353 2342 11 B6   
10 2013 1 3 704 700 4 1036 1025 11 WN   
# … with 2,105 more rows, 9 more variables: flight <int>, tailnum <chr>,  
# origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
# minute <dbl>, time\_hour <dttm>, and abbreviated variable names  
# ¹​sched\_dep\_time, ²​dep\_delay, ³​arr\_time, ⁴​sched\_arr\_time, ⁵​arr\_delay

# Were operated by United (`UA`), American (`AA`), or Delta (`DL`)  
  
flights %>%   
 filter(carrier == "UA" | carrier == "AA" | carrier == "DL")

# A tibble: 139,504 × 19  
 year month day dep\_time sched\_de…¹ dep\_d…² arr\_t…³ sched…⁴ arr\_d…⁵ carrier  
 <int> <int> <int> <int> <int> <dbl> <int> <int> <dbl> <chr>   
 1 2013 1 1 517 515 2 830 819 11 UA   
 2 2013 1 1 533 529 4 850 830 20 UA   
 3 2013 1 1 542 540 2 923 850 33 AA   
 4 2013 1 1 554 600 -6 812 837 -25 DL   
 5 2013 1 1 554 558 -4 740 728 12 UA   
 6 2013 1 1 558 600 -2 753 745 8 AA   
 7 2013 1 1 558 600 -2 924 917 7 UA   
 8 2013 1 1 558 600 -2 923 937 -14 UA   
 9 2013 1 1 559 600 -1 941 910 31 AA   
10 2013 1 1 559 600 -1 854 902 -8 UA   
# … with 139,494 more rows, 9 more variables: flight <int>, tailnum <chr>,  
# origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
# minute <dbl>, time\_hour <dttm>, and abbreviated variable names  
# ¹​sched\_dep\_time, ²​dep\_delay, ³​arr\_time, ⁴​sched\_arr\_time, ⁵​arr\_delay

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

# A tibble: 86,326 × 19  
 year month day dep\_time sched\_de…¹ dep\_d…² arr\_t…³ sched…⁴ arr\_d…⁵ carrier  
 <int> <int> <int> <int> <int> <dbl> <int> <int> <dbl> <chr>   
 1 2013 7 1 1 2029 212 236 2359 157 B6   
 2 2013 7 1 2 2359 3 344 344 0 B6   
 3 2013 7 1 29 2245 104 151 1 110 B6   
 4 2013 7 1 43 2130 193 322 14 188 B6   
 5 2013 7 1 44 2150 174 300 100 120 AA   
 6 2013 7 1 46 2051 235 304 2358 186 B6   
 7 2013 7 1 48 2001 287 308 2305 243 VX   
 8 2013 7 1 58 2155 183 335 43 172 B6   
 9 2013 7 1 100 2146 194 327 30 177 B6   
10 2013 7 1 100 2245 135 337 135 122 B6   
# … with 86,316 more rows, 9 more variables: flight <int>, tailnum <chr>,  
# origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
# minute <dbl>, time\_hour <dttm>, and abbreviated variable names  
# ¹​sched\_dep\_time, ²​dep\_delay, ³​arr\_time, ⁴​sched\_arr\_time, ⁵​arr\_delay

# Arrived more than two hours late, but didn't leave late  
  
flights %>%   
 filter(arr\_delay >= 120 & dep\_delay <= 0)

# A tibble: 29 × 19  
 year month day dep\_time sched\_de…¹ dep\_d…² arr\_t…³ sched…⁴ arr\_d…⁵ carrier  
 <int> <int> <int> <int> <int> <dbl> <int> <int> <dbl> <chr>   
 1 2013 1 27 1419 1420 -1 1754 1550 124 MQ   
 2 2013 10 7 1350 1350 0 1736 1526 130 EV   
 3 2013 10 7 1357 1359 -2 1858 1654 124 AA   
 4 2013 10 16 657 700 -3 1258 1056 122 B6   
 5 2013 11 1 658 700 -2 1329 1015 194 VX   
 6 2013 3 18 1844 1847 -3 39 2219 140 UA   
 7 2013 4 17 1635 1640 -5 2049 1845 124 MQ   
 8 2013 4 18 558 600 -2 1149 850 179 AA   
 9 2013 4 18 655 700 -5 1213 950 143 AA   
10 2013 5 22 1827 1830 -3 2217 2010 127 MQ   
# … with 19 more rows, 9 more variables: flight <int>, tailnum <chr>,  
# origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
# minute <dbl>, time\_hour <dttm>, and abbreviated variable names  
# ¹​sched\_dep\_time, ²​dep\_delay, ³​arr\_time, ⁴​sched\_arr\_time, ⁵​arr\_delay

# Were delayed by at least an hour, but made up over 30 minutes in flight  
  
flights %>%   
 filter(dep\_delay >= 60 & dep\_delay - arr\_delay >=30)

# A tibble: 2,074 × 19  
 year month day dep\_time sched\_de…¹ dep\_d…² arr\_t…³ sched…⁴ arr\_d…⁵ carrier  
 <int> <int> <int> <int> <int> <dbl> <int> <int> <dbl> <chr>   
 1 2013 1 1 1716 1545 91 2140 2039 61 B6   
 2 2013 1 1 2205 1720 285 46 2040 246 AA   
 3 2013 1 1 2326 2130 116 131 18 73 B6   
 4 2013 1 3 1503 1221 162 1803 1555 128 UA   
 5 2013 1 3 1821 1530 171 2131 1910 141 AA   
 6 2013 1 3 1839 1700 99 2056 1950 66 AA   
 7 2013 1 3 1850 1745 65 2148 2120 28 AA   
 8 2013 1 3 1923 1815 68 2036 1958 38 9E   
 9 2013 1 3 1941 1759 102 2246 2139 67 UA   
10 2013 1 3 1950 1845 65 2228 2227 1 B6   
# … with 2,064 more rows, 9 more variables: flight <int>, tailnum <chr>,  
# origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
# minute <dbl>, time\_hour <dttm>, and abbreviated variable names  
# ¹​sched\_dep\_time, ²​dep\_delay, ³​arr\_time, ⁴​sched\_arr\_time, ⁵​arr\_delay

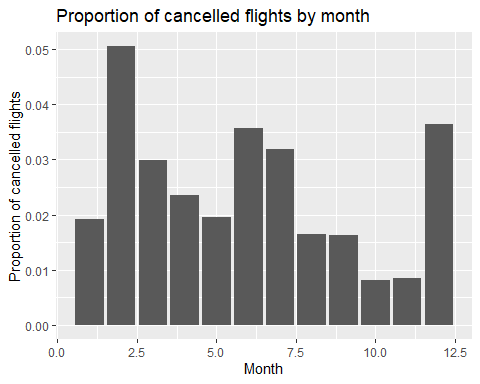
## Problem 2: What months had the highest and lowest proportion of cancelled flights? Interpret any seasonal patterns. To determine if a flight was cancelled use the following code

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

# What months had the highest and lowest % of cancelled flights?  
# Calculate the total number of flights and cancelled flights for each month  
flights\_summary <- flights %>%  
 group\_by(year, month) %>%  
 summarise(total\_flights = n(), cancelled\_flights = sum(is.na(dep\_time)))

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

# Calculate the proportion of cancelled flights for each month  
flights\_summary <- flights\_summary %>%  
 mutate(prop\_cancelled = cancelled\_flights / total\_flights)  
  
# Determine the months with the highest and lowest proportion of cancelled flights  
highest\_prop\_cancelled <- flights\_summary %>%   
 arrange(desc(prop\_cancelled)) %>%   
 slice(1)  
  
lowest\_prop\_cancelled <- flights\_summary %>%   
 arrange(prop\_cancelled) %>%   
 slice(1)  
  
# Interpret any seasonal patterns in the data  
# You can create a bar plot to visualize the proportion of cancelled flights across different months:  
  
library(ggplot2) # load the ggplot2 package  
  
ggplot(flights\_summary, aes(x = month, y = prop\_cancelled)) +  
 geom\_bar(stat = "identity") +  
 xlab("Month") +  
 ylab("Proportion of cancelled flights") +  
 ggtitle("Proportion of cancelled flights by month")



There appears to be no consistent pattern in the proportion of cancelled flights across the year, however, very qualitative assumptions can be made from the following graph. It must be remembered that additional analysis is required to confirm these assumptions.

The months that appear to have the highest proportion of cancellations are February, June, July and December.

December is likely to have increased cancellations due to winter weather. February is often a stormy month in many countries. For example, this is seen particularly in the US as this is when their own Winter storms occur, such as the recent Winter Storm Olive.

June and July are likely to have a higher proportion of cancellations and this is a very busy time for flying, with it being the start of summer. Airlines will often be at capacity during this period, and increase strain will likely increase the proportion of issues and therefore cancellations.

## 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.

top\_plane\_tailnum <- planes %>%  
 filter(seats > 50) %>%  
 select(tailnum) %>%  
 left\_join(flights, by = "tailnum") %>%  
 filter(year == 2013, origin %in% c("LGA", "JFK", "EWR")) %>%  
 group\_by(tailnum) %>%  
 summarise(num\_flights = n()) %>%  
 filter(num\_flights == max(num\_flights)) %>%  
 pull(tailnum)  
  
top\_plane\_tailnum

[1] "N328AA"

top\_plane\_routes <- flights %>%  
 filter(year == 2013, tailnum == top\_plane\_tailnum) %>%  
 group\_by(dest) %>%  
 summarise(num\_flights = n()) %>%  
 arrange(desc(num\_flights))  
  
top\_plane\_routes

# A tibble: 6 × 2  
 dest num\_flights  
 <chr> <int>  
1 LAX 313  
2 SFO 52  
3 MIA 25  
4 BOS 1  
5 MCO 1  
6 SJU 1

N328AA (an American Airlines Boeing 767-223ER) is the plane with the greatest number of flights from NYC airports in 2013.

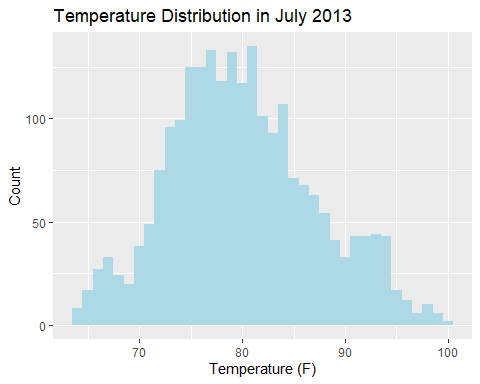
## Problem 4: The nycflights13 package includes a table (weather) that describes the weather during 2013. Use that table to answer the following questions:

**What is the distribution of temperature (`temp`) in July 2013? Identify any important outliers in terms of the `wind\_speed` variable.**

weather

# A tibble: 26,115 × 15  
 origin year month day hour temp dewp humid wind\_dir wind\_speed wind\_g…¹  
 <chr> <int> <int> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 EWR 2013 1 1 1 39.0 26.1 59.4 270 10.4 NA  
 2 EWR 2013 1 1 2 39.0 27.0 61.6 250 8.06 NA  
 3 EWR 2013 1 1 3 39.0 28.0 64.4 240 11.5 NA  
 4 EWR 2013 1 1 4 39.9 28.0 62.2 250 12.7 NA  
 5 EWR 2013 1 1 5 39.0 28.0 64.4 260 12.7 NA  
 6 EWR 2013 1 1 6 37.9 28.0 67.2 240 11.5 NA  
 7 EWR 2013 1 1 7 39.0 28.0 64.4 240 15.0 NA  
 8 EWR 2013 1 1 8 39.9 28.0 62.2 250 10.4 NA  
 9 EWR 2013 1 1 9 39.9 28.0 62.2 260 15.0 NA  
10 EWR 2013 1 1 10 41 28.0 59.6 260 13.8 NA  
# … with 26,105 more rows, 4 more variables: precip <dbl>, pressure <dbl>,  
# visib <dbl>, time\_hour <dttm>, and abbreviated variable name ¹​wind\_gust

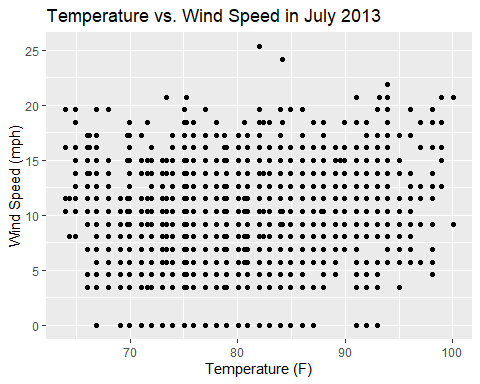
weather %>%  
 filter(month == 7, year == 2013) %>%  
 ggplot(aes(x = temp)) +  
 geom\_histogram(binwidth = 1, fill = "lightblue") +  
 labs(title = "Temperature Distribution in July 2013",  
 x = "Temperature (F)", y = "Count")



From the plot, we can see that the distribution of temperature is roughly normal, with a peak around 80 degrees Fahrenheit. There are a few outliers with very high wind speeds, which we can identify using a scatter plot of temperature against wind speed:

weather %>%  
 filter(month == 7, year == 2013) %>%  
 ggplot(aes(x = temp, y = wind\_speed)) +  
 geom\_point() +  
 labs(title = "Temperature vs. Wind Speed in July 2013",  
 x = "Temperature (F)", y = "Wind Speed (mph)")

Warning: Removed 2 rows containing missing values (geom\_point).

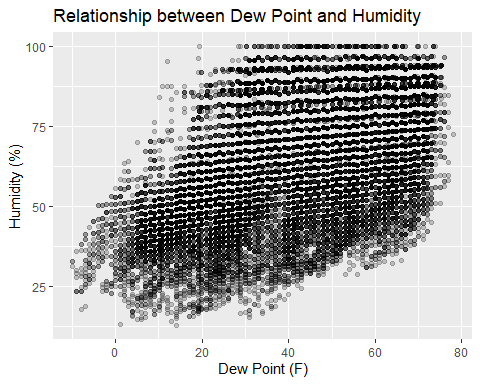


We can see that there are a few points with very high wind speeds (> 22.5 mph). These points could be considered outliers in terms of the wind speed variable.

**- What is the relationship between `dewp` and `humid`?**

weather %>%  
 ggplot(aes(x = dewp, y = humid)) +  
 geom\_point(alpha = 0.2) +  
 labs(title = "Relationship between Dew Point and Humidity",  
 x = "Dew Point (F)", y = "Humidity (%)")

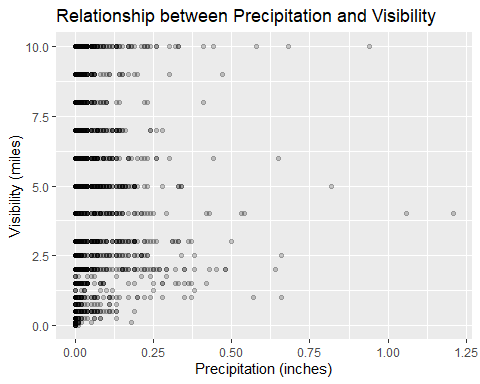
Warning: Removed 1 rows containing missing values (geom\_point).



We can see that there is a strong positive relationship between **dewp** and **humid**. As the dew point temperature increases, the humidity also tends to increase.

**- What is the relationship between `precip` and `visib`?**

weather %>%  
 ggplot(aes(x = precip, y = visib)) +  
 geom\_point(alpha = 0.2) +  
 labs(title = "Relationship between Precipitation and Visibility",  
 x = "Precipitation (inches)", y = "Visibility (miles)")



Although it would be plausible to think that higher precipitation will lead to lower visibility, there does not seem to be a strong relationship between the two, as seen through the large variety in visibility at similar levels of precipitation.

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

**- How many planes have a missing date of manufacture?**

planes

# A tibble: 3,322 × 9  
 tailnum year type manuf…¹ model engines seats speed engine  
 <chr> <int> <chr> <chr> <chr> <int> <int> <int> <chr>   
 1 N10156 2004 Fixed wing multi engi… EMBRAER EMB-… 2 55 NA Turbo…  
 2 N102UW 1998 Fixed wing multi engi… AIRBUS… A320… 2 182 NA Turbo…  
 3 N103US 1999 Fixed wing multi engi… AIRBUS… A320… 2 182 NA Turbo…  
 4 N104UW 1999 Fixed wing multi engi… AIRBUS… A320… 2 182 NA Turbo…  
 5 N10575 2002 Fixed wing multi engi… EMBRAER EMB-… 2 55 NA Turbo…  
 6 N105UW 1999 Fixed wing multi engi… AIRBUS… A320… 2 182 NA Turbo…  
 7 N107US 1999 Fixed wing multi engi… AIRBUS… A320… 2 182 NA Turbo…  
 8 N108UW 1999 Fixed wing multi engi… AIRBUS… A320… 2 182 NA Turbo…  
 9 N109UW 1999 Fixed wing multi engi… AIRBUS… A320… 2 182 NA Turbo…  
10 N110UW 1999 Fixed wing multi engi… AIRBUS… A320… 2 182 NA Turbo…  
# … with 3,312 more rows, and abbreviated variable name ¹​manufacturer

missingmanufacturedate <- planes %>%   
 filter(is.na(year)) %>%   
 count()  
  
missingmanufacturedate

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

There are 70 planes with a missing date of manufacture.

**- What are the five most common manufacturers?**

planes %>%   
 mutate(manufacturer = recode(manufacturer, `AIRBUS INDUSTRIE` = 'AIRBUS')) %>%  
 count(manufacturer, sort = TRUE) %>%   
 head(5)

# A tibble: 5 × 2  
 manufacturer n  
 <chr> <int>  
1 BOEING 1630  
2 AIRBUS 736  
3 BOMBARDIER INC 368  
4 EMBRAER 299  
5 MCDONNELL DOUGLAS 120

The five most common manufacturers are Boeing, McDonnell Douglas, Bombardier, Airbus and Embraer.

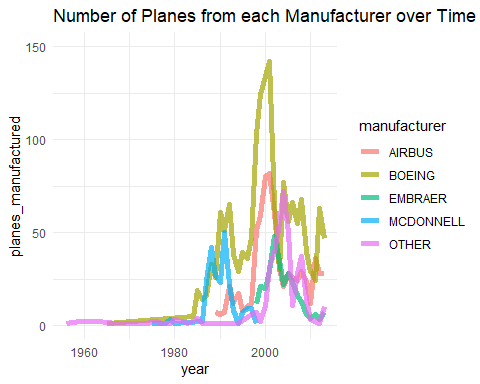
**- Has the distribution of manufacturer changed over time as reflected by the airplanes flying from NYC in 2013? (Hint: you may need to use case\_when() to recode the manufacturer name and collapse rare vendors into a category called Other.)**

tailnums\_from\_NYC <- flights %>%  
 filter(origin %in% c("LGA", "JFK", "EWR")) %>%  
 distinct(tailnum) %>%  
 rename(tailnum = tailnum)  
  
planes\_dept\_nyc <- left\_join(tailnums\_from\_NYC, planes, by = "tailnum") %>%  
 select(tailnum, manufacturer, year)  
  
planes\_dept\_nyc <- planes\_dept\_nyc %>%  
 mutate(manufacturer = case\_when(  
 manufacturer %in% c("AIRBUS INDUSTRIE", "AIRBUS") ~ "AIRBUS",  
 manufacturer %in% c("MCDONNELL DOUGLAS", "MCDONNELL DOUGLAS AIRCRAFT CO", "MCDONNELL DOUGLAS CORPORATION") ~ "MCDONNELL",  
 manufacturer == "BOEING" ~ "BOEING",  
 manufacturer == "EMBRAER" ~ "EMBRAER",  
 manufacturer == "BOMBARDIER" ~ "BOMBARDIER",  
 TRUE ~ "OTHER"  
 )) %>%  
  
 group\_by(year, manufacturer) %>%  
 summarise(planes\_manufactured = n()) %>%  
 arrange(desc(planes\_manufactured))

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

ggplot(planes\_dept\_nyc, aes(x = year, y = planes\_manufactured, color = manufacturer)) +  
 geom\_line(size = 2, alpha = 0.7) +  
 scale\_y\_continuous(limits = c(0, 150)) +  
 theme\_minimal() +  
 labs(title = "Number of Planes from each Manufacturer over Time")

Warning: Removed 5 row(s) containing missing values (geom\_path).



Over time, less popular plane manufacters such as Embraer and McDonnell have fallen into obscurity. Although the market became more fragmented prior to 2010, it is clear now that the market is mainly consolidated between Airbus and Boeing.

## 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?

N381AA is the oldest plane that flew from NYC airports in 2013

# What is the oldest plane (specified by the tailnum variable) that flew from New York City airports in 2013?  
flights %>%   
 filter(year == 2013, origin %in% c("JFK", "LGA", "EWR")) %>%   
 select(tailnum) %>%   
 distinct() %>%   
 left\_join(planes %>%   
 select(tailnum, year) %>%   
 group\_by(tailnum) %>%   
 summarise(min\_year = min(year, na.rm = TRUE)),   
 by = "tailnum") %>%   
 arrange(min\_year) %>%   
 slice\_head(n = 1)

Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf  
  
Warning in min(year, na.rm = TRUE): no non-missing arguments to min; returning  
Inf

# A tibble: 1 × 2  
 tailnum min\_year  
 <chr> <dbl>  
1 N381AA 1956

# How many airplanes that flew from New York City are included in the planes table?  
flights %>%   
 filter(origin %in% c("JFK", "LGA", "EWR")) %>%   
 select(tailnum) %>%   
 distinct() %>%   
 left\_join(planes %>%   
 select(tailnum) %>%   
 distinct(),   
 by = "tailnum") %>%   
 count()

# A tibble: 1 × 1  
 n  
 <int>  
1 4044

4044 planes that flew from NYC are included in the planes table.

## 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.

**- What is the median arrival delay on a month-by-month basis in each airport?**

flights %>%  
 group\_by(month, origin) %>%  
 summarise(median\_arr\_delay = median(arr\_delay, na.rm = TRUE))

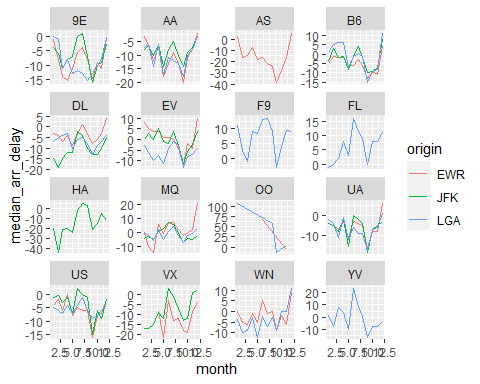
`summarise()` has grouped output by 'month'. You can override using the  
`.groups` argument.

# A tibble: 36 × 3  
# Groups: month [12]  
 month origin median\_arr\_delay  
 <int> <chr> <dbl>  
 1 1 EWR 0  
 2 1 JFK -7  
 3 1 LGA -4  
 4 2 EWR -2  
 5 2 JFK -5  
 6 2 LGA -4  
 7 3 EWR -4  
 8 3 JFK -7  
 9 3 LGA -7  
10 4 EWR -1  
# … with 26 more rows

**- For each airline, plot the median arrival delay for each month and origin airport.**

flights %>%  
 group\_by(carrier, month, origin) %>%  
 summarise(median\_arr\_delay = median(arr\_delay, na.rm = TRUE)) %>%  
 ggplot(aes(x = month, y = median\_arr\_delay, color = origin)) +  
 geom\_line() +  
 facet\_wrap(~ carrier, scales = "free\_y")

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



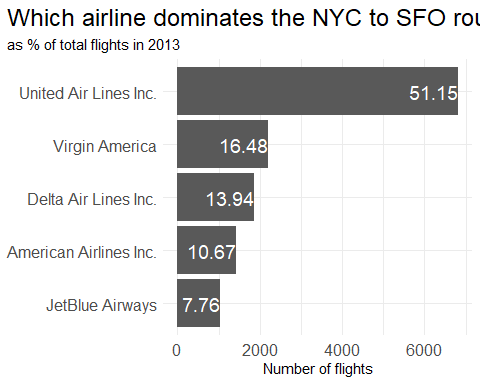
## Problem 8: Let’s take a closer look at what carriers service the route to San Francisco International (SFO). Join the flights and airlines tables and count which airlines flew the most to SFO. Produce a new dataframe, fly\_into\_sfo that contains three variables: the name of the airline, e.g., United 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.

# Join flights and airlines tables  
fly\_sfo <- flights %>%  
 filter(dest == "SFO") %>%  
 left\_join(airlines, by = "carrier")   
  
# Count flights by airline  
fly\_into\_sfo <- fly\_sfo %>%  
 group\_by(name) %>%  
 summarise(count = n()) %>%  
 arrange(desc(count)) %>%  
 mutate(percent = round((count/sum(count))\*100, 2))  
  
# View fly\_into\_sfo dataframe  
fly\_into\_sfo

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

And here is some bonus ggplot code to plot the 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))

To create this plot, I would need to group the cancellations data by month, carrier, and airport origin, and then count the number of cancelled flights for each combination of these variables. I could then create separate histograms for each combination of airport origin (EWR and JFK), with each histogram showing the distribution of cancelled flights for each month. The histograms could be arranged in a 2-column, 5-row grid. I could add appropriate axis labels, a title, and a legend to the plot. To achieve this, we can use **dplyr** and **ggplot2** packages in R. I will then use **facet\_grid()** to split the histograms by airport origin and use **geom\_histogram()** to plot the histograms.

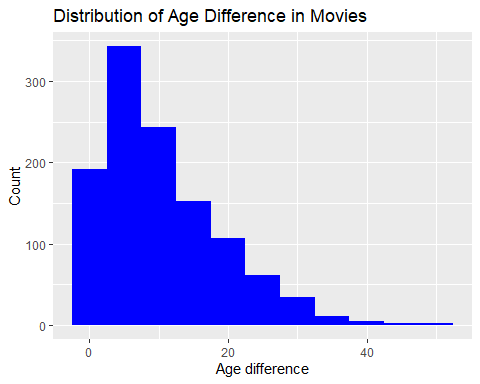
## Problem 10: On your own – Hollywood Age Gap

Utilising the Hollywood Age Gap dataset, we can discover interesting facts and trends about how we view romantic relationships in movies, particularly over time. Firstly, we must import the dataset.

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.

1. **To explore the distribution of age differences between movie love interests, we can create a histogram or density plot of the age\_difference variable. We can calculate summary statistics like mean, median, and standard deviation to get an idea of the typical age difference**

* 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.
* # Histogram of age difference  
  ggplot(age\_gaps, aes(x = age\_difference)) +  
   geom\_histogram(binwidth = 5, fill = 'blue') +  
   labs(x = 'Age difference', y = 'Count', title = 'Distribution of Age Difference in Movies')
* 
* # Summary statistics of age difference  
  summary(age\_gaps$age\_difference)
* Min. 1st Qu. Median Mean 3rd Qu. Max.   
   0.00 4.00 8.00 10.42 15.00 52.00
* When observing this data, it is clear that the most common age difference within movies is around 5-10 years. Although we see see some age differences ranging up to 50 years old, however, this is very uncommon.

1. **We can apply the half plus seven rule to the dataset by calculating the age limits based on each character’s age and see how frequently the age difference falls within the acceptable range.**

* 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.
* # Compute minimum and maximum partner age according to the half plus seven rule  
  min\_partner\_age <- age\_gaps$actor\_1\_age / 2 + 7  
  max\_partner\_age <- (age\_gaps$actor\_1\_age - 7) \* 2  
    
  # Count how many times the rule applies  
  sum(age\_gaps$actor\_2\_age > min\_partner\_age & age\_gaps$actor\_2\_age < max\_partner\_age)
* [1] 795
* # Count how many times the rule does not apply  
  sum(age\_gaps$actor\_2\_age < min\_partner\_age & age\_gaps$actor\_2\_age < max\_partner\_age)
* [1] 326
* sum(age\_gaps$actor\_2\_age < min\_partner\_age & age\_gaps$actor\_2\_age > max\_partner\_age)
* [1] 0
* This occurs 795 times, however, there are many times in which the rule does not apply, for example, there are 326 occasions when the rule does not apply,.

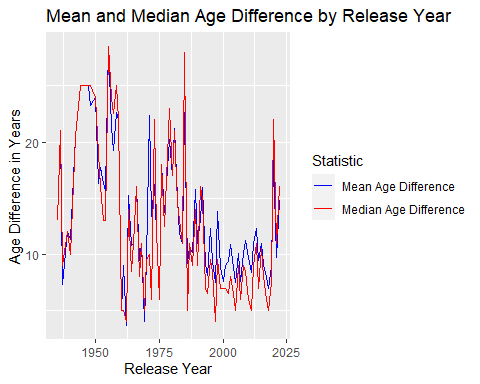
1. **We can identify the movie with the greatest number of love interests by grouping the data by movie\_name and counting the number of unique couple\_numbers.**

* 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.
* # Summing of number of interests - using slice to get the answer from the descend order list.  
    
  most\_interests <- age\_gaps %>%   
   group\_by(movie\_name) %>%   
   summarise(num\_interests = n\_distinct(couple\_number)) %>%   
   arrange(desc(num\_interests)) %>%   
   slice(1)  
    
  most\_interests
* # A tibble: 1 × 2  
   movie\_name num\_interests  
   <chr> <int>  
  1 Love Actually 7
* The answer to this is Love Actually with 7 love interests. This is expected as this film follows various different characters and storylines, unlike typical movies which have a more focused, singular storyline.

1. **We can identify the actors/actresses with the greatest number of love interests by grouping the data by actor\_1\_name and actor\_2\_name and counting the number of unique movie\_names they appear in.**

* 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.
* age\_gaps %>%  
   group\_by(actor\_1\_name) %>%  
   summarise(n\_love\_interests = n\_distinct(movie\_name)) %>%  
   arrange(desc(n\_love\_interests)) %>%  
   head(10)
* # A tibble: 10 × 2  
   actor\_1\_name n\_love\_interests  
   <chr> <int>  
   1 Keanu Reeves 20  
   2 Adam Sandler 17  
   3 Harrison Ford 12  
   4 Johnny Depp 11  
   5 Leonardo DiCaprio 10  
   6 Richard Gere 10  
   7 Brad Pitt 9  
   8 Humphrey Bogart 9  
   9 Sean Connery 9  
  10 Tom Cruise 9
* age\_gaps %>%  
   group\_by(actor\_2\_name) %>%  
   summarise(n\_love\_interests = n\_distinct(movie\_name)) %>%  
   arrange(desc(n\_love\_interests)) %>%  
   head(10)
* # A tibble: 10 × 2  
   actor\_2\_name n\_love\_interests  
   <chr> <int>  
   1 Scarlett Johansson 12  
   2 Emma Stone 11  
   3 Keira Knightley 10  
   4 Drew Barrymore 9  
   5 Julia Roberts 9  
   6 Amanda Seyfried 8  
   7 Renee Zellweger 8  
   8 Audrey Hepburn 7  
   9 Emily Blunt 7  
  10 Jennifer Aniston 7

1. **We can create a scatter plot of release\_year against age\_difference to see if the mean/median age difference stays constant over the years.**

* # calculate mean and median age difference by release year  
  age\_diff\_by\_year <- age\_gaps %>%  
   group\_by(release\_year) %>%  
   summarise(mean\_age\_diff = mean(age\_difference),   
   median\_age\_diff = median(age\_difference))  
    
  # create line plots to visualize the trend over time  
  ggplot(age\_diff\_by\_year, aes(x = release\_year)) +  
   geom\_line(aes(y = mean\_age\_diff, color = "Mean Age Difference")) +  
   geom\_line(aes(y = median\_age\_diff, color = "Median Age Difference")) +  
   labs(title = "Mean and Median Age Difference by Release Year",  
   x = "Release Year", y = "Age Difference in Years",  
   color = "Statistic") +  
   scale\_color\_manual(values = c("Mean Age Difference" = "blue", "Median Age Difference" = "red"))
* 
* It certainly does not stay constant over the years, and has even seen a high increase in both mean and median age difference in recent years.

1. **We can filter the data by same-gender love interests and calculate the frequency of occurrence to see how often Hollywood depicts same-gender relationships.**

* 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.
* age\_gaps %>%  
   filter(!is.na(character\_1\_gender), !is.na(character\_2\_gender)) %>%  
   group\_by(character\_1\_gender, character\_2\_gender) %>%  
   summarise(n\_movies = n\_distinct(movie\_name)) %>%  
   mutate(total\_movies = sum(n\_movies)) %>%  
   mutate(percentage = n\_movies/total\_movies\*100) %>%  
   select(character\_1\_gender, character\_2\_gender, n\_movies, percentage) %>%  
   arrange(desc(n\_movies))
* `summarise()` has grouped output by 'character\_1\_gender'. You can override  
  using the `.groups` argument.
* # A tibble: 4 × 4  
  # Groups: character\_1\_gender [2]  
   character\_1\_gender character\_2\_gender n\_movies percentage  
   <chr> <chr> <int> <dbl>  
  1 man woman 717 98.5   
  2 woman man 186 94.9   
  3 man man 11 1.51  
  4 woman woman 10 5.10

It portrays same-sex love interests 6.61% of the time. This is very uncommon, however, when observing the data is appears it has become more common in recent years.

# 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.