Homerwork 1

Qi Ren

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

# A tibble: 10,200 × 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 1 811 630 101 1047 830  
 2 2013 1 1 848 1835 853 1001 1950  
 3 2013 1 1 957 733 144 1056 853  
 4 2013 1 1 1114 900 134 1447 1222  
 5 2013 1 1 1505 1310 115 1638 1431  
 6 2013 1 1 1525 1340 105 1831 1626  
 7 2013 1 1 1549 1445 64 1912 1656  
 8 2013 1 1 1558 1359 119 1718 1515  
 9 2013 1 1 1732 1630 62 2028 1825  
10 2013 1 1 1803 1620 103 2008 1750  
# ℹ 10,190 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>

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

# 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`)  
  
flights %>%   
 filter(carrier == "UA" | carrier == "AA" | carrier == "DL")

# 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)  
  
flights %>%   
 filter(month %in% c(7:9))

# 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  
  
flights %>%   
 filter(arr\_delay >= 120&dep\_delay == 0)

# A tibble: 3 × 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 10 7 1350 1350 0 1736 1526  
2 2013 5 23 1810 1810 0 2208 2000  
3 2013 7 1 905 905 0 1443 1223  
# 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 delayed by at least an hour, but made up over 30 minutes in flight  
  
flights %>%   
 filter(dep\_delay >= 60 & arr\_delay <= 30)

# A tibble: 239 × 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 3 1850 1745 65 2148 2120  
 2 2013 1 3 1950 1845 65 2228 2227  
 3 2013 1 3 2015 1915 60 2135 2111  
 4 2013 1 6 1019 900 79 1558 1530  
 5 2013 1 7 1543 1430 73 1758 1735  
 6 2013 1 11 1020 920 60 1311 1245  
 7 2013 1 12 1706 1600 66 1949 1927  
 8 2013 1 12 1953 1845 68 2154 2137  
 9 2013 1 19 1456 1355 61 1636 1615  
10 2013 1 21 1531 1430 61 1843 1815  
# ℹ 229 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))

#Intuition: 1) find no. of cancelled flights per month;  
#2) find total no. of flights per month;  
#3) find %  
#4) find max and min  
  
  
cancelled <- flights %>%   
 mutate(cancel = is.na(dep\_time)) %>% #Find all cancelled flights  
 group\_by(month, cancel) %>% #find all cancelled flights per month   
 summarise(n = n()) %>% #Count the total num of cancelled flights  
 mutate(prop = n/sum(n)) %>% #find the % of cancellation  
 filter(cancel == "TRUE") %>%   
 arrange(desc(prop))

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

flights %>%   
 group\_by(month) %>%   
 mutate(total = n()) %>%   
 filter(is.na(dep\_time)) %>%   
 group\_by(month) %>%   
 mutate(number = n()) %>%   
 mutate(percentage = number/total)

# A tibble: 8,255 × 22  
# Groups: month [12]  
 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 NA 1630 NA NA 1815  
 2 2013 1 1 NA 1935 NA NA 2240  
 3 2013 1 1 NA 1500 NA NA 1825  
 4 2013 1 1 NA 600 NA NA 901  
 5 2013 1 2 NA 1540 NA NA 1747  
 6 2013 1 2 NA 1620 NA NA 1746  
 7 2013 1 2 NA 1355 NA NA 1459  
 8 2013 1 2 NA 1420 NA NA 1644  
 9 2013 1 2 NA 1321 NA NA 1536  
10 2013 1 2 NA 1545 NA NA 1910  
# ℹ 8,245 more rows  
# ℹ 14 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>, total <int>, number <int>,  
# percentage <dbl>

## 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.

#intuition: 1) tailnumber that appeared the most times  
#2) count the number of flights  
#3) the destination it flew to  
mostflights <- flights %>%  
 group\_by(tailnum) %>% #group by tailnumber  
 summarise(n=n(),dest) %>% #summarise with count of flights (n), and destination  
 arrange(desc(n)) #arrange according to flight numbers

Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in  
dplyr 1.1.0.  
ℹ Please use `reframe()` instead.  
ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`  
 always returns an ungrouped data frame and adjust accordingly.

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

newtable <- left\_join(mostflights, planes, by = "tailnum") %>% #join table with "planes"  
 filter(seats >= 50) %>% #filter for seats >= 50   
 summarise(max\_flights = max(n), #summarise for max number of flights, destination, tail number, and destination  
 dest,  
 tailnum,  
 n) %>%   
 arrange(desc(n)) #arrange descending order of n

Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in  
dplyr 1.1.0.  
ℹ Please use `reframe()` instead.  
ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`  
 always returns an ungrouped data frame and adjust accordingly.

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

## 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`?

data(weather)  
prob\_4 <- weather %>%   
 filter(month == 7) %>%   
 select(temp, wind\_speed, dewp, humid, precip, visib) %>%  
 summary(prob\_4)

## 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.)

data(planes, flights)  
prob5a <- planes %>%   
 filter(is.na(year)) #70 planes have missing DoM  
  
prob5b <- planes %>%   
 group\_by(manufacturer) %>% #group by manufacturers  
 summarise(numplane = n()) %>% #count no. of planes in manufacturers  
 arrange(desc(numplane)) #arrange descending order  
  
prob5c1 <- planes %>%   
 mutate(across("manufacturer", str\_replace, "AIRBUS INDUSTRIE", "AIRBUS"), across("manufacturer", str\_replace, "MCDONNELL DOUGLAS AIRCRAFT CO|MCDONNELL DOUGLAS CORPORATION", "MCDONNELL DOUGLAS")) %>%   
 summarise(manufacturer, tailnum) %>%   
 group\_by(manufacturer) #group by manufacturers

Warning: There was 1 warning in `mutate()`.  
ℹ In argument: `across("manufacturer", str\_replace, "AIRBUS INDUSTRIE",  
 "AIRBUS")`.  
Caused by warning:  
! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.  
Supply arguments directly to `.fns` through an anonymous function instead.  
  
 # Previously  
 across(a:b, mean, na.rm = TRUE)  
  
 # Now  
 across(a:b, \(x) mean(x, na.rm = TRUE))

Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in  
dplyr 1.1.0.  
ℹ Please use `reframe()` instead.  
ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`  
 always returns an ungrouped data frame and adjust accordingly.

prob5c2 <- flights %>%   
 summarise(month, day, tailnum)

Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in  
dplyr 1.1.0.  
ℹ Please use `reframe()` instead.  
ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`  
 always returns an ungrouped data frame and adjust accordingly.

prob5c3 <- left\_join(prob5c1, prob5c2, by = "tailnum") %>%   
 group\_by(manufacturer) %>%   
 #intuition: should have 1 column for month, 1 column for manufacturer name, and 1 column for aggregated n() for each manufacturer  
 summarise(month, manufacturer, numplane = n()) %>% #numplane = number of rows for each manufacturer in each month  
 ungroup() %>%  
 mutate(manufacturer = case\_when(  
 numplane < 500 ~ "Others",  
 TRUE ~ manufacturer  
 )) %>% #when numplane < 500, replace manufacturer name with "Others"  
 group\_by(month, manufacturer) %>%   
 summarise(count = n()) %>% #creates a new column "count" that is the row of numplane for each group  
 arrange(desc(count))

Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in  
dplyr 1.1.0.  
ℹ Please use `reframe()` instead.  
ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`  
 always returns an ungrouped data frame and adjust accordingly.

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

## 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?

data(flights, planes)  
flights1 <- flights %>%   
 summarise(tailnum)

Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in  
dplyr 1.1.0.  
ℹ Please use `reframe()` instead.  
ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`  
 always returns an ungrouped data frame and adjust accordingly.

planes1 <- planes %>%   
 summarise(tailnum, year)

Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in  
dplyr 1.1.0.  
ℹ Please use `reframe()` instead.  
ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`  
 always returns an ungrouped data frame and adjust accordingly.

prob6 <- semi\_join(planes1, flights1, by = "tailnum") %>%   
 arrange(year)  
#oldest plane is N381AA, manufactured in 1956  
#3322 planes 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.

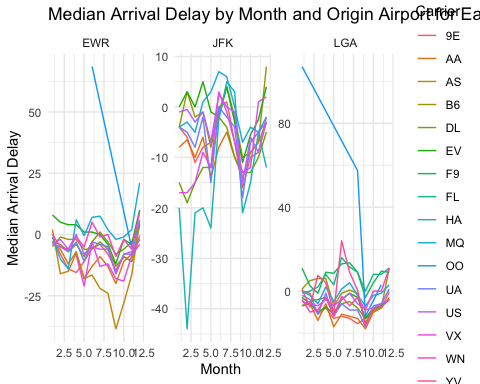
#intuition:  
#1) data needed: arrival delay, airport code, month, day  
#2) output: 1 column for month (1-12), 1 column for each airport, data: median arrival delay  
  
prob7a <- flights %>%  
 group\_by(month, origin) %>%   
 summarise(median\_arr\_delay = median(arr\_delay, na.rm = TRUE))

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

library(ggplot2)  
prob7b <- flights %>%   
 group\_by(month, origin, carrier) %>%  
 summarise(median\_arr\_delay = median(arr\_delay, na.rm = TRUE))

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

# Now plot the data  
ggplot(prob7b, aes(x = month, y = median\_arr\_delay, color = carrier)) +  
 geom\_line() +  
 facet\_wrap(~origin, scales = "free\_y") +  
 labs(x = "Month",   
 y = "Median Arrival Delay",   
 color = "Carrier",   
 title = "Median Arrival Delay by Month and Origin Airport for Each Airline") +  
 theme\_minimal()

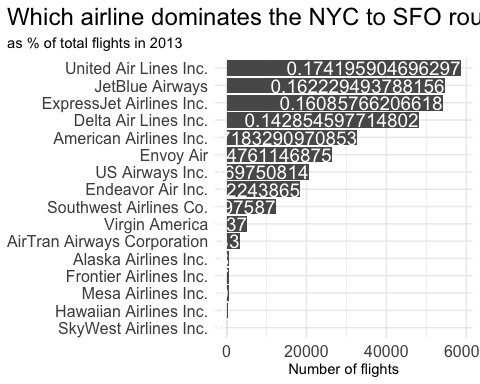


## Problem 8: Let’s take a closer look at what carriers service the route to San Francisco International (SFO). Join the flights and airlines tables and count which airlines flew the most to SFO. Produce a new dataframe, fly\_into\_sfo that contains three variables: the name of the airline, e.g., United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

data(flights)  
data(airlines)  
fly\_into\_sfo <- left\_join(flights, airlines, by = "carrier") %>%   
 group\_by(name) %>%   
 summarise(count = n()) %>%  
 mutate(percent = (count/sum(count))) %>%   
 arrange(desc(count))

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))

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.



# Plot the number of cancellations with month as x-axis, counts of cancellation as y-axis, and each individual graph is the carrier

## 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 |

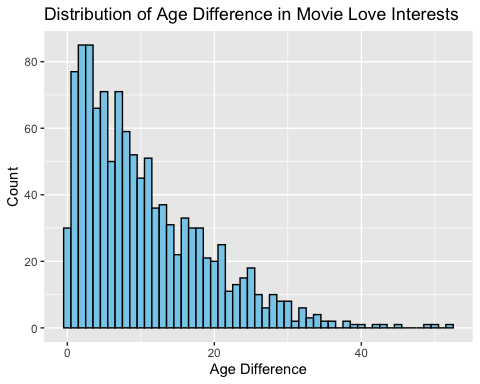
age\_gaps <- readr::read\_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2023/2023-02-14/age\_gaps.csv')

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.

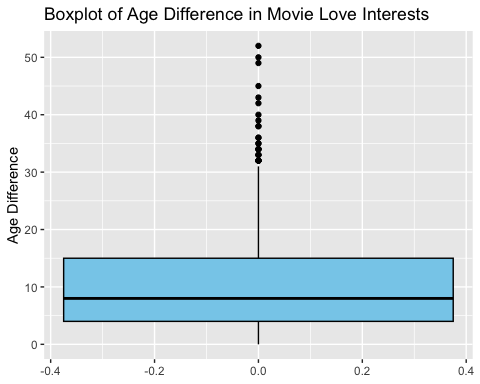
summary(age\_gaps$age\_difference)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 0.00 4.00 8.00 10.42 15.00 52.00

# Histogram  
ggplot(age\_gaps, aes(x = age\_difference)) +  
 geom\_histogram(binwidth = 1, fill = "skyblue", color = "black") +  
 labs(x = "Age Difference", y = "Count", title = "Distribution of Age Difference in Movie Love Interests")



# Boxplot  
ggplot(age\_gaps, aes(y = age\_difference)) +  
 geom\_boxplot(fill = "skyblue", color = "black") +  
 labs(y = "Age Difference", title = "Boxplot of Age Difference in Movie Love Interests")



# Typical age difference (mean)  
mean\_age\_difference <- mean(age\_gaps$age\_difference)  
mean\_age\_difference

[1] 10.42424

#Half+7 rule  
  
# Calculate the lower and upper age boundaries based on the rule  
lower\_bound <- (age\_gaps$your\_age / 2) + 7

Warning: Unknown or uninitialised column: `your\_age`.

upper\_bound <- (age\_gaps$your\_age - 7) \* 2

Warning: Unknown or uninitialised column: `your\_age`.

# Count the number of relationships that meet the rule's criteria  
rule\_applicable <- sum(age\_gaps$partner\_age > lower\_bound & age\_gaps$partner\_age < upper\_bound)

Warning: Unknown or uninitialised column: `partner\_age`.

Warning: Unknown or uninitialised column: `partner\_age`.

# Calculate the proportion of relationships that meet the rule  
rule\_applicable\_proportion <- rule\_applicable / nrow(age\_gaps)  
  
rule\_applicable\_proportion

[1] 0

#Which movie has the greatest number of love interests?  
  
love\_interest\_counts <- table(age\_gaps$movie\_name)  
  
# Find the movie with the highest count of love interests  
movie\_with\_most\_love\_interests <- names(love\_interest\_counts)[which.max(love\_interest\_counts)]  
  
movie\_with\_most\_love\_interests

[1] "Love Actually"

#Which actors/ actresses have the greatest number of love interests in this dataset?  
  
love\_interest\_counts <- table(age\_gaps$actor\_actress\_name)

Warning: Unknown or uninitialised column: `actor\_actress\_name`.

# Find the actor/actress with the highest count of love interests  
actors\_with\_most\_love\_interests <- names(love\_interest\_counts)[which.max(love\_interest\_counts)]  
  
actors\_with\_most\_love\_interests

NULL

#Is the mean/median age difference staying constant over the years (1935 - 2022)?  
  
# Group the data by year and calculate mean and median age difference  
age\_difference\_stats <- age\_gaps %>%  
 group\_by(release\_year) %>%  
 summarise(mean\_age\_difference = mean(age\_difference),  
 median\_age\_difference = median(age\_difference))  
  
age\_difference\_stats

# A tibble: 82 × 3  
 release\_year mean\_age\_difference median\_age\_difference  
 <dbl> <dbl> <dbl>  
 1 1935 13 13   
 2 1936 21 21   
 3 1937 7.33 9   
 4 1939 12 12   
 5 1940 11.3 10   
 6 1942 20.5 20.5  
 7 1944 25 25   
 8 1946 25 25   
 9 1947 25 25   
10 1948 23.2 25   
# ℹ 72 more rows

#How frequently does Hollywood depict same-gender love interests?  
  
same\_gender\_love\_interests <- age\_gaps %>%  
 filter(character\_1\_gender == character\_2\_gender) # comare gender of 2 characters  
  
# Calculate the proportion of same-gender love interests  
proportion\_same\_gender <- nrow(same\_gender\_love\_interests) / nrow(age\_gaps)  
  
proportion\_same\_gender

[1] 0.01991342

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:

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.