HW_4 Data-412

Ethan Pastel

2024-01-29

1. Load and review the data, Load the {tidyverse} and {nycflights 13} packages, Load the flights data frame, Use a function to identify the variables, Use a function to identify the number of observations (rows), Use a function to show only the first three rows.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(nycflights13)
```

```
## Warning: package 'nycflights13' was built under R version 4.3.2
```

```
flights_data <- flights
head(flights_data)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # 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>
```

```r
str(flights_data)
```

```
## tibble [336,776 x 19] (S3: tbl_df/tbl/data.frame)
##  $ year          : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int [1:336776] 517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int [1:336776] 515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int [1:336776] 830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int [1:336776] 819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr [1:336776] "UA" "UA" "AA" "B6" ...
##  $ flight        : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time      : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : num [1:336776] 1400 1416 1089 1576 762 ...
##  $ hour          : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

```r
paste("The number of observations in flights data is:", nrow(flights_data))
```

```
## [1] "The number of observations in flights data is: 336776"
```

2. Worst Plane to Fly

Which planes (tailnum) have the three worst (highest) average departure delay record?

```r
worst_planes <- flights %>%
  group_by(tailnum) %>%
  summarise(avg_dep_delay = mean(dep_delay, na.rm = TRUE)) %>%
  arrange(desc(avg_dep_delay)) %>%
  head(3)

print(worst_planes)
```

```
## # A tibble: 3 x 2
##   tailnum avg_dep_delay
##   <chr>           <dbl>
## 1 N844MH            297
## 2 N922EV            274
## 3 N587NW            272
```

How many trips did each make?

```
worst_planes_total_trips <- flights %>%
  filter(tailnum %in% worst_planes$tailnum) %>%
  group_by(tailnum) %>%
  summarise(num_trips = n())

print(worst_planes_total_trips)
```

```
## # A tibble: 3 x 2
##   tailnum num_trips
##   <chr>       <int>
## 1 N587NW          1
## 2 N844MH          1
## 3 N922EV          1
```

Now only look tailnums where each flew more than 15 trips and find the three tailnums with the highest average departure delay and show the tailnums with their average departure delay and number of trips in decreasing order of amount of delay.

```
filtered_planes <- flights %>%
  group_by(tailnum) %>%
  filter(n() > 15) %>%
  summarise(avg_dep_delay = mean(dep_delay, na.rm = TRUE), num_trips = n()) %>%
  arrange(desc(avg_dep_delay)) %>%
  head(3)

print(filtered_planes)
```

```
## # A tibble: 3 x 3
##   tailnum avg_dep_delay num_trips
##   <chr>           <dbl>     <int>
## 1 N184DN           54.7        16
## 2 N203FR           53.5        41
## 3 N645MQ           52.5        25
```

3. Best Time of Day to Fly

Calculate the average departure delay for each trip based on the hour the planes were scheduled to depart.

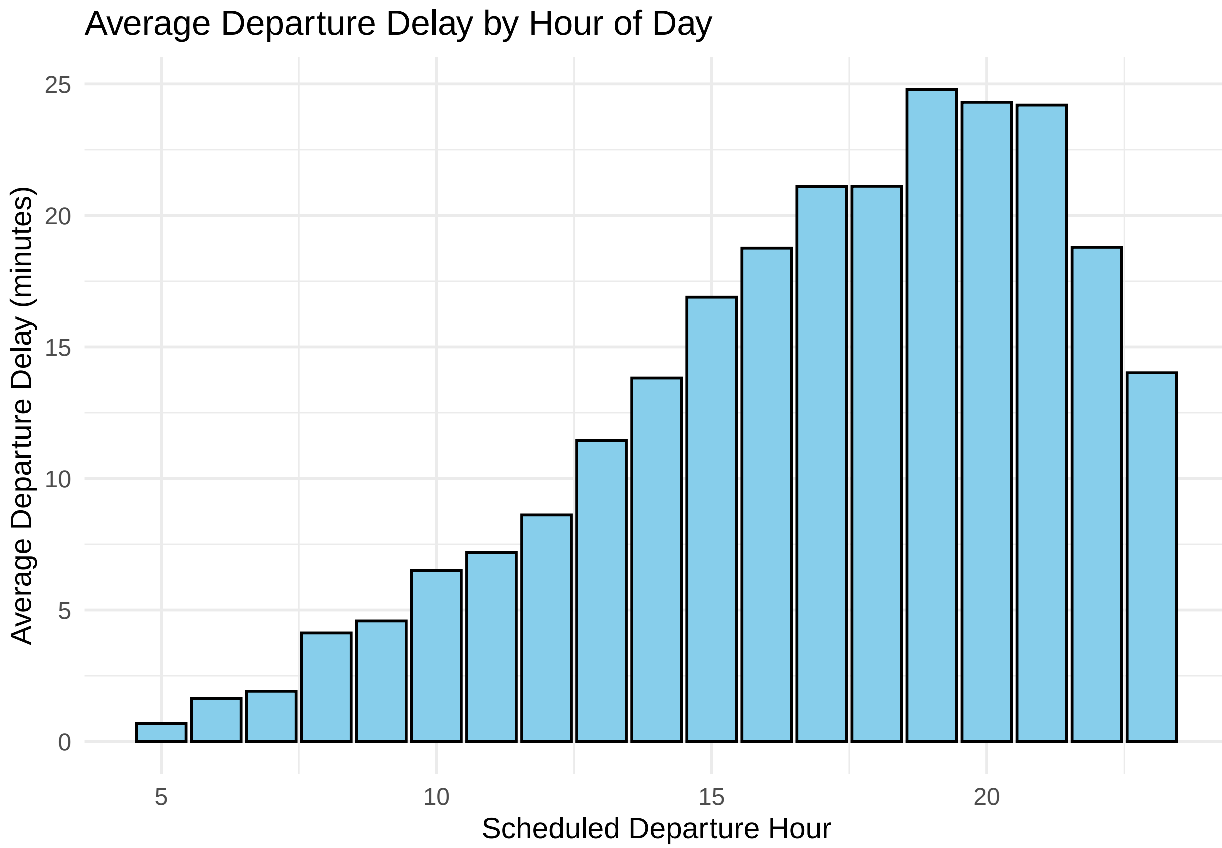
```r
avg_delay_by_hour <- flights %>%
  group_by(hour = sched_dep_time %/% 100) %>%
  summarise(avg_dep_delay = mean(dep_delay, na.rm = TRUE))


print(avg_delay_by_hour)
```

```
## # A tibble: 20 x 2
##     hour avg_dep_delay
##    <dbl>         <dbl>
##  1     1       NaN
##  2     5         0.688
##  3     6         1.64
##  4     7         1.91
##  5     8         4.13
##  6     9         4.58
##  7    10         6.50
##  8    11         7.19
##  9    12         8.61
## 10    13        11.4
## 11    14        13.8
## 12    15        16.9
## 13    16        18.8
## 14    17        21.1
## 15    18        21.1
## 16    19        24.8
## 17    20        24.3
## 18    21        24.2
## 19    22        18.8
## 20    23        14.0
```

Create a plot to show the average departure delay for each hour of the day. What hour of the day you should schedule your trip to minimize your expected (average) delay time.

```r
ggplot(avg_delay_by_hour, aes(x = hour, y = avg_dep_delay)) + geom_bar(fill = "skyblue", color = "black
```

```
## Warning: Removed 1 rows containing missing values (‘position_stack()‘).
```




```
cat("Hour 5 and below is the best hour of the day to minimize the expected (average) delay time.")
```

```
## Hour 5 and below is the best hour of the day to minimize the expected (average) delay time.
```

4. Worst Trips for each Destination

a. In a single series of piped steps (no intermediate variables), complete the following to produce a single output.

-For each destination, compute the total minutes of arrival delay, then use that result to,

-Compute for each trip to a destination, its proportion of the total arrival delay for the destination; then use that result to . . .

-Sort by destination, alphabetically, and from the highest to lowest proportion of total delay for each

destination; then use that result to . . .

-Choose for each trip, the destination, total arrival delay, flight number, proportion of the delay, number of trips, and the year, month, and day of departure; then use that result to . . .

-Show the worst trip for each destination where the destination has total delay >= 0.

b. Repeat but instead of the worst trip for a destination find the worst flight number, replace the following: Step 2 Compute for each flight number to a destination. Step 4, choose for each flight number, the destination, total arrival delay, flight number, proportion of the delay, number of trips, Step 5 Show the worst flight number . . .

```
worst_trips <- flights %>%
  group_by(dest) %>%
  summarise(total_delay = sum(arr_delay, na.rm = TRUE)) %>%
  left_join(flights, by = "dest") %>%
  mutate(prop_delay = arr_delay / total_delay) %>%
  arrange(dest, desc(prop_delay)) %>%
  filter(total_delay >= 0) %>%
  group_by(dest) %>%
  slice_head(n = 1) %>%
  mutate(worst_flight_number = tailnum)


print(worst_trips)
```

```
## # A tibble: 96 x 22
## # Groups:   dest [96]
##    dest  total_delay  year month   day dep_time sched_dep_time dep_delay
##    <chr>       <dbl> <int> <int> <int>    <int>          <int>     <dbl>
##  1 ABQ          1113  2013     7    22     2145           2007        98
##  2 ACK          1281  2013     7    23     1139            800       219
##  3 ALB          6018  2013     1    25      123           2000       323
##  4 ATL        190260  2013     7    22     2257            759       898
##  5 AUS         14514  2013     7    10     2056           1505       351
##  6 AVL          2089  2013     8    13     1156            832       204
##  7 BDL          2904  2013     2    21     1728           1316       252
##  8 BGR          2874  2013    12     1     1504           1056       248
##  9 BHM          4540  2013     4    10       25           1900       325
## 10 BNA         71867  2013     1    25     2020           1527       293
## # i 86 more rows
## # i 14 more variables: arr_time <int>, sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>,
## #   prop_delay <dbl>, worst_flight_number <chr>
```

5. estinations with Multiple Carriers from NYC Area

Find all destinations that are flown by at least three carriers and show the destination airports ranked 35-40 for the most carriers out of the NYC airports. (hint: use n_distinct())

```
destinations_multiple_carriers <- flights %>%
  group_by(dest, origin, carrier) %>%
  summarise() %>%
  group_by(dest) %>%
  summarise(num_carriers = n_distinct(carrier)) %>%
  filter(num_carriers >= 3) %>%
  arrange(desc(num_carriers), dest) %>%
  slice(35:40)
```

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

```
print(destinations_multiple_carriers)
```

```
## # A tibble: 6 x 2
##   dest  num_carriers
##   <chr>        <int>
## 1 RSW              4
## 2 SAN              4
## 3 SJU              4
## 4 SRQ              4
## 5 BTV              3
## 6 CMH              3
```

6. Effect of the Delay in the Flight before Yours

Delays are typically temporally correlated: even once the problem causing the initial delay has been resolved, later flights are delayed to allow earlier flights to leave.

6.1 Using lag() and cor(), explore how the departure delay of a flight is related to the delay of the

immediately preceding flight. Hint. Think through how the data needs to be grouped, filtered, and organized so the lag makes sense from both physical/temporal perspectives. Calculate the lag and save to a new data frame. Use the new data frame to calculate the daily correlation for each airport - Hint- there should be 1095 rows in the answer. Hint. Look at the help for cor() to select the correct arguments to handle the NAs created by lag(). Calculate the mean and median of the daily correlation for each airport along with the number of days considered and arrange in decreasing order by mean. Use an appropriate plot to show the distribution of the daily correlation for each airport. Interpret the numerical summary and plot to answer the question which airport appears to have the highest average daily correlation between subsequent flight delays.

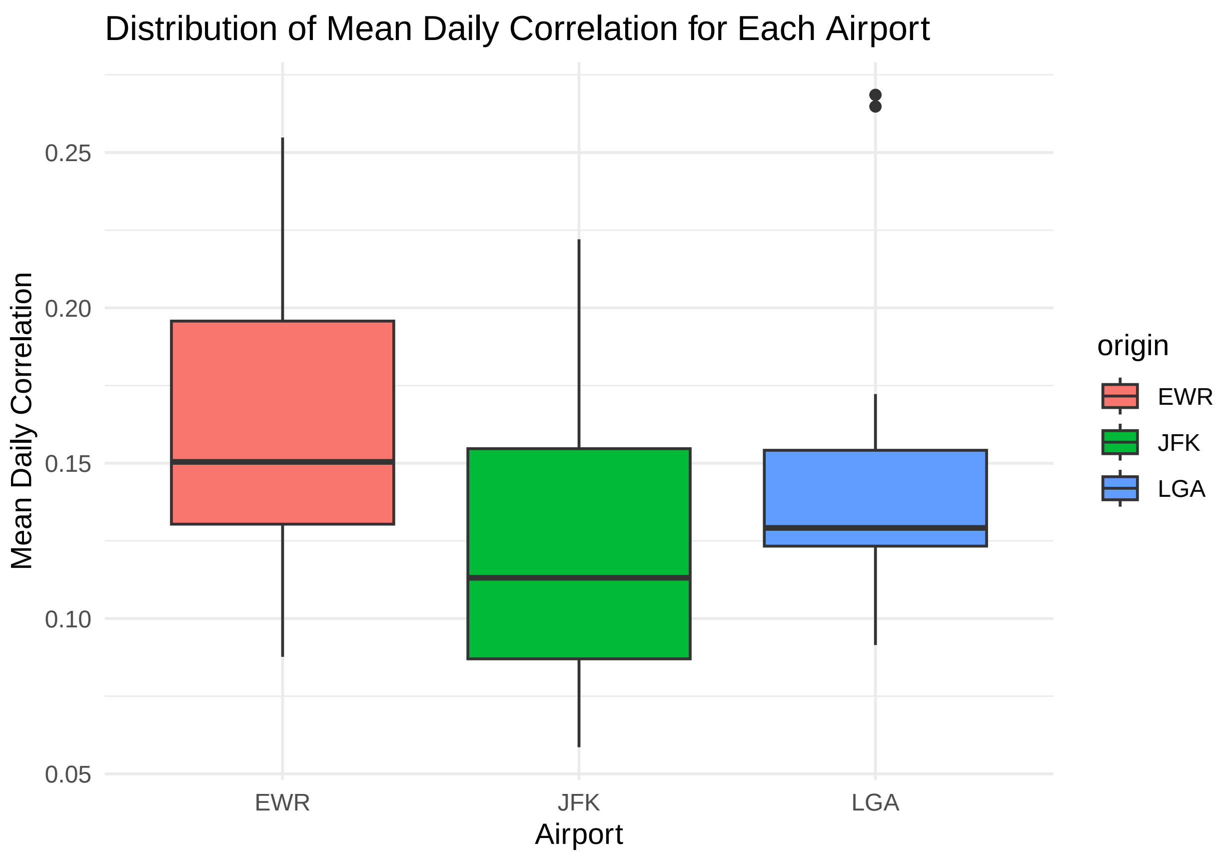
```r
delay_data <- flights %>%
  select(origin, dep_delay, year, month, day, dep_time) %>%
  filter(!is.na(dep_delay))


lagged_data <- delay_data %>%
  group_by(origin) %>%
  arrange(origin, year, month, day, dep_time) %>%
  mutate(prev_dep_delay = lag(dep_delay))

daily_correlation <- lagged_data %>%
  group_by(origin, year, month, day) %>%
  summarise(correlation = cor(dep_delay, prev_dep_delay, use = "complete.obs")) %>%
  summarise(mean_correlation = mean(correlation, na.rm = TRUE), median_correlation = median(correlation
```

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

```r
daily_correlation_sorted <- daily_correlation %>%
  arrange(desc(mean_correlation))

print(daily_correlation_sorted)
```

```
## # A tibble: 36 x 6
## # Groups:   origin, year [3]
##    origin  year month mean_correlation median_correlation num_days
##    <chr>  <int> <int>            <dbl>              <dbl>    <int>
##  1 LGA     2013     6            0.269              0.210       30
##  2 LGA     2013     7            0.265              0.213       31
##  3 EWR     2013     6            0.255              0.247       30
##  4 EWR     2013     7            0.235              0.182       31
##  5 JFK     2013     7            0.222              0.245       31
##  6 EWR     2013     4            0.204              0.200       30
##  7 JFK     2013     6            0.199              0.160       30
##  8 EWR     2013     5            0.193              0.221       31
##  9 EWR     2013     3            0.190              0.176       31
## 10 JFK     2013     5            0.175              0.161       31
## # i 26 more rows
```

```r
ggplot(daily_correlation, aes(x = origin, y = mean_correlation, fill = origin)) + geom_boxplot() + labs
```




```
paste("The aiports that appear to have the highest average daily correlation between subsequent flight
```

```
## [1] "The aiports that appear to have the highest average daily correlation between subsequent flight
```

2 Star Wars Charracters. The starwars data frame in the dplyr package contains demographic characteristics of various characters from the hit franchise Star Wars.

1. Load the data into R and load any necessary packages.

```
library(dplyr)

data("starwars")
head(starwars)
```

```
## # A tibble: 6 x 14
##   name      height  mass hair_color skin_color eye_color birth_year sex   gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
## 1 Luke Sky~    172    77 blond      fair       blue              19 male  mascu~
## 2 C-3PO        167    75 <NA>       gold       yellow           112 none  mascu~
```

```
## 3 R2-D2          96    32 <NA>       white, bl~ red             33    none   mascu~
## 4 Darth Va~     202   136 none       white      yellow          41.9 male   mascu~
## 5 Leia Org~     150    49 brown      light      brown           19    fema~ femin~
## 6 Owen Lars     178   120 brown, gr~ light      blue            52    male   mascu~
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

2. Use appropriate {dplyr} code to determine which individuals have missing gender. Make sure to only

print out their names and heights, arranged in ascending order of height.

```r
missing_gender_data <- starwars %>%
  filter(is.na(gender)) %>%
  select(name, height) %>%
  arrange(height)

print(missing_gender_data)
```

```
## # A tibble: 4 x 2
##   name           height
##   <chr>           <int>
## 1 Sly Moore         178
## 2 Ric Olié          183
## 3 Quarsh Panaka     183
## 4 Captain Phasma     NA
```

3. Use a {dplyr} function to change their gender to "nonbinary" and save to the dataframe.

```r
new_starwars <- starwars %>%
  mutate(gender = if_else(is.na(gender), "nonbinary", gender))

print(new_starwars)
```

```
## # A tibble: 87 x 14
##    name     height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Luke Sk~    172    77 blond      fair       blue            19   male  mascu~
##  2 C-3PO       167    75 <NA>       gold       yellow         112   none  mascu~
##  3 R2-D2        96    32 <NA>       white, bl~ red             33   none  mascu~
##  4 Darth V~    202   136 none       white      yellow          41.9 male  mascu~
##  5 Leia Or~    150    49 brown      light      brown           19   fema~ femin~
##  6 Owen La~    178   120 brown, gr~ light      blue            52   male  mascu~
##  7 Beru Wh~    165    75 brown      light      blue            47   fema~ femin~
##  8 R5-D4        97    32 <NA>       white, red red             NA   none  mascu~
##  9 Biggs D~    183    84 black      light      brown           24   male  mascu~
## 10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57   male  mascu~
```

```
## # i 77 more rows
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

4. The body mass index (BMI) is defined as masskg

height2m

That is, the BMI is the weight of the individual (in kg) divided by the height of the individual (in m)

squared. There are 100 cm in each m. Use a {dplyr} function to add the BMI for each individual in the starwars data frame and save it to the starwars data frame. Sanity check: The median BMI should be in the 20-30 range.

```
new_starwars <- new_starwars %>%
  mutate(height_m = height / 100)


new_starwars <- new_starwars %>%
  mutate(BMI = mass / height_m^2)

median_BMI <- median(new_starwars$BMI, na.rm = TRUE)

print(median_BMI)
```

```
## [1] 24.67038
```

```
print(new_starwars)
```

```
## # A tibble: 87 x 16
##    name     height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Luke Sk~    172    77 blond      fair       blue            19   male  mascu~
##  2 C-3PO       167    75 <NA>       gold       yellow         112   none  mascu~
##  3 R2-D2        96    32 <NA>       white, bl~ red             33   none  mascu~
##  4 Darth V~    202   136 none       white      yellow          41.9 male  mascu~
##  5 Leia Or~    150    49 brown      light      brown           19   fema~ femin~
##  6 Owen La~    178   120 brown, gr~ light      blue            52   male  mascu~
##  7 Beru Wh~    165    75 brown      light      blue            47   fema~ femin~
##  8 R5-D4        97    32 <NA>       white, red red             NA   none  mascu~
##  9 Biggs D~    183    84 black      light      brown           24   male  mascu~
## 10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57   male  mascu~
## # i 77 more rows
## # i 7 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>, height_m <dbl>, BMI <dbl>
```

5. Use a single block of code using {dplyr} functions to calculate the median and mean height for each gender as well as the number of individuals and show the results.

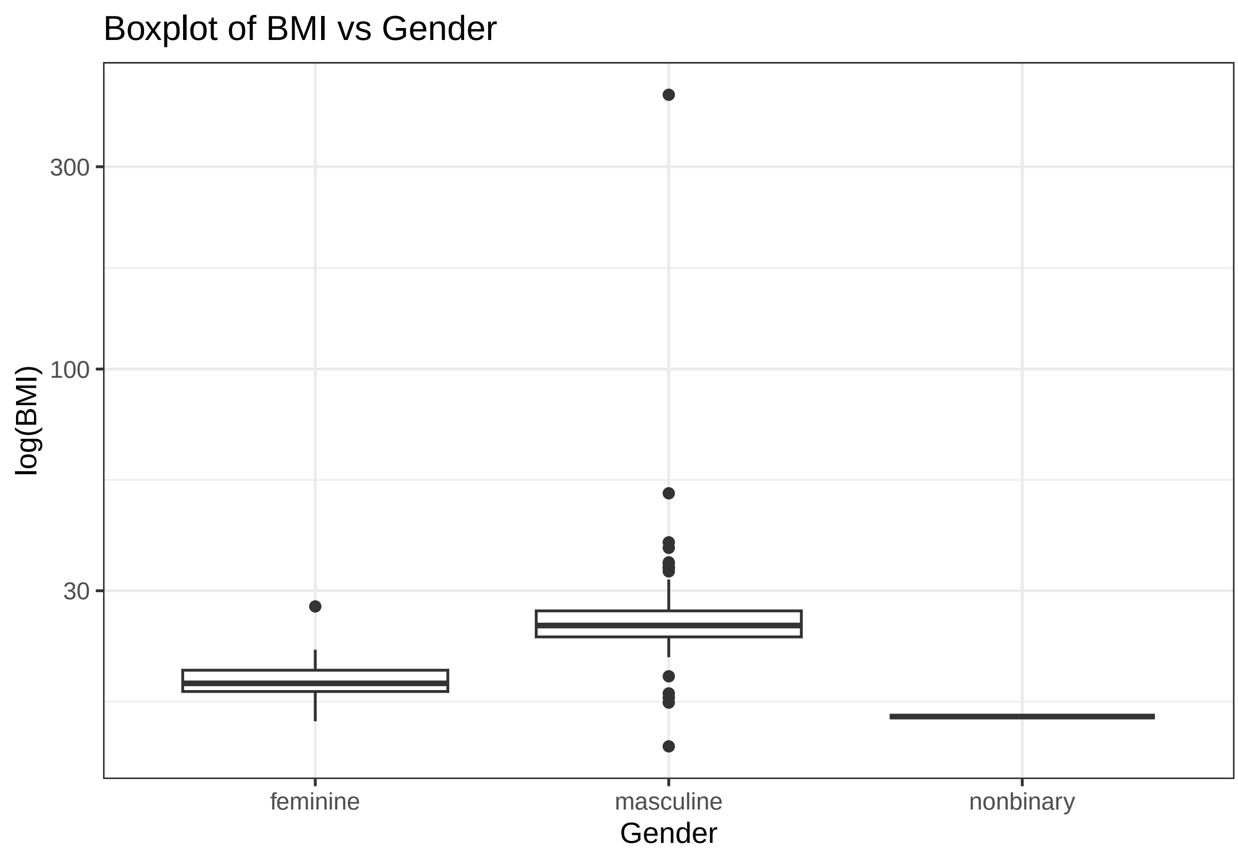
```
breakdown <- new_starwars %>%
  group_by(gender) %>%
  summarize(
    median_height = median(height, na.rm = TRUE),
    mean_height = mean(height, na.rm = TRUE),
    num_individuals = n()
  )
print(breakdown)
```

```
## # A tibble: 3 x 4
##   gender    median_height mean_height num_individuals
##   <chr>             <dbl>       <dbl>           <int>
## 1 feminine           166.        165.              17
## 2 masculine          183         177.              66
## 3 nonbinary          183         181.               4
```

6. Use {ggplot2} to create a boxplot of BMI vs gender. Include only masculine, feminine, and nonbinary. Use a log 10 scale for Y axis. Use the black and white theme.

```
ggplot(new_starwars, aes(x = gender, y = BMI)) + geom_boxplot() + scale_y_log10() + labs(title = "Boxpl
```

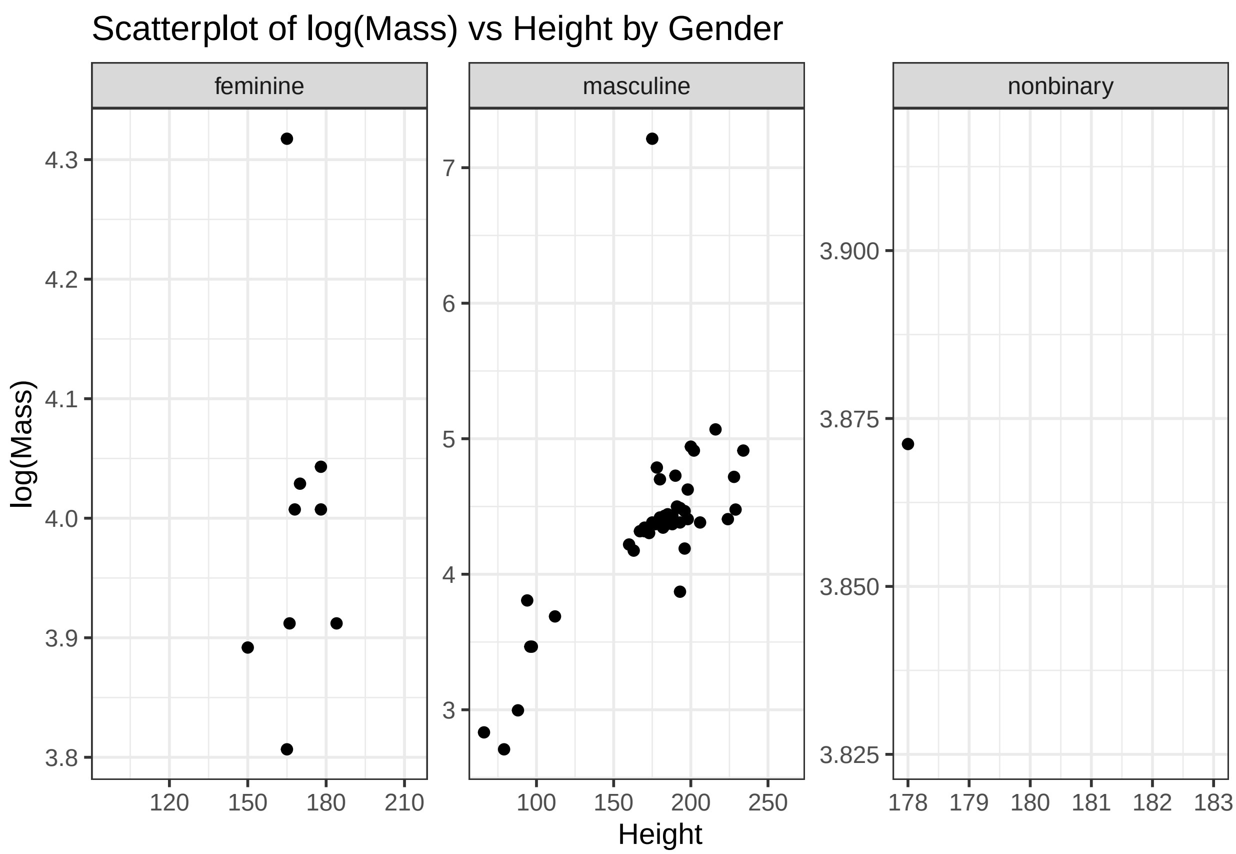
```
## Warning: Removed 28 rows containing non-finite values (`stat_boxplot()`).
```



7. Use {ggplot2} to create a scatterplot of log(mass) vs height, faceting by gender.

```
ggplot(new_starwars, aes(x = height, y = log(mass))) +
  geom_point() + facet_wrap(~ gender, scales = "free") + labs(title = "Scatterplot of log(Mass) vs Heig
```

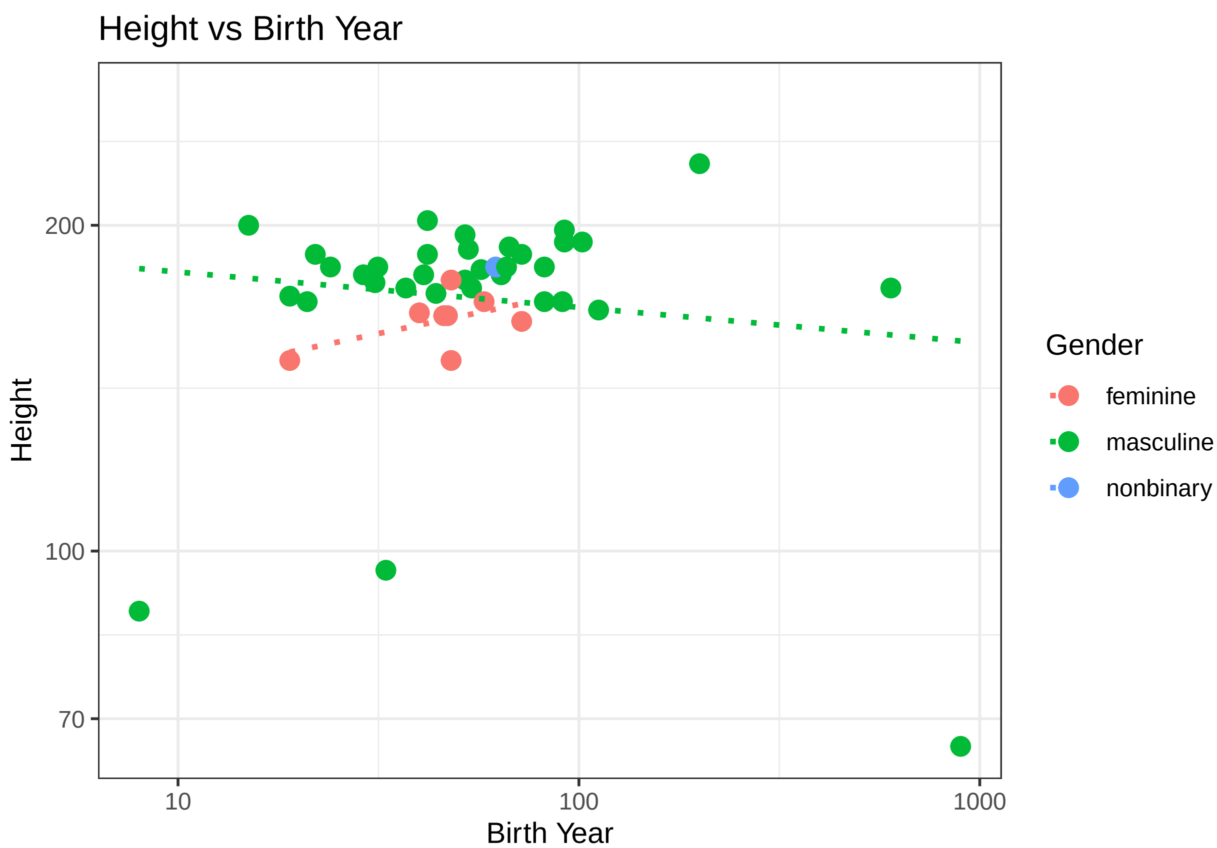
```
## Warning: Removed 28 rows containing missing values (`geom_point()`).
```



8. Reproduce the following plot (hint: filter based on birth year):

```
ggplot(new_starwars, aes(x = birth_year, y = height, color = gender)) + geom_point(size = 3) + geom_smo
  labs(title = "Height vs Birth Year",
       x = "Birth Year",
       y = "Height",
       color = "Gender") +
  theme_bw()
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 44 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 44 rows containing missing values (`geom_point()`).
```



3 3. The Palmer Penguins Package.

1. Install the {palmerpenguins} package in the console, load the package and the penguins data set, and look at the first 6 rows

```
library(palmerpenguins)
```

```
## Warning: package 'palmerpenguins' was built under R version 4.3.2
```

```
data("penguins")

head(penguins)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
## 1 Adelie  Torgersen           39.1          18.7               181        3750
## 2 Adelie  Torgersen           39.5          17.4               186        3800
## 3 Adelie  Torgersen           40.3          18                 195        3250
## 4 Adelie  Torgersen           NA            NA                  NA          NA
```

```
## 5 Adelie  Torgersen           36.7          19.3                 193        3450
## 6 Adelie  Torgersen           39.3          20.6                 190        3650
## # i 2 more variables: sex <fct>, year <int>
```

2. Bill Ratio. For each penguin calculate the ratio of the flipper length to the maximum of the bill length or the bill depth. Save to the data frame as the variable fb_ratio.

```r
penguins <- penguins %>%
  mutate(fb_ratio = flipper_length_mm / pmax(bill_length_mm, bill_depth_mm, na.rm = TRUE))

print(head(penguins))
```

```
## # A tibble: 6 x 9
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
## 1 Adelie  Torgersen           39.1          18.7               181        3750
## 2 Adelie  Torgersen           39.5          17.4               186        3800
## 3 Adelie  Torgersen           40.3          18                 195        3250
## 4 Adelie  Torgersen           NA            NA                  NA          NA
## 5 Adelie  Torgersen           36.7          19.3               193        3450
## 6 Adelie  Torgersen           39.3          20.6               190        3650
## # i 3 more variables: sex <fct>, year <int>, fb_ratio <dbl>
```

3. Using the data frame from 2., eliminate the penguins with an fb_ration of NA and show the highest

four penguins of each sex with only the factor variables and fb_ratio. Your code should not include any variables names other than sex and fb_ratio.

```r
new_penguins <- penguins %>% filter(!is.na(fb_ratio))

top_four_fb_penguins_each_sex <- new_penguins %>%
  group_by(sex) %>%
  arrange(desc(fb_ratio)) %>%
  select(sex, fb_ratio) %>%
  slice_head(n = 4)


print(top_four_fb_penguins_each_sex)
```

```
## # A tibble: 12 x 2
## # Groups:   sex [3]
##    sex    fb_ratio
##    <fct>     <dbl>
##  1 female     5.86
##  2 female     5.67
```

```
##   3 female     5.66
##   4 female     5.49
##   5 male       5.72
##   6 male       5.50
##   7 male       5.34
##   8 male       5.31
##   9 <NA>       5.66
## 10 <NA>       4.92
## 11 <NA>       4.88
## 12 <NA>       4.85
```

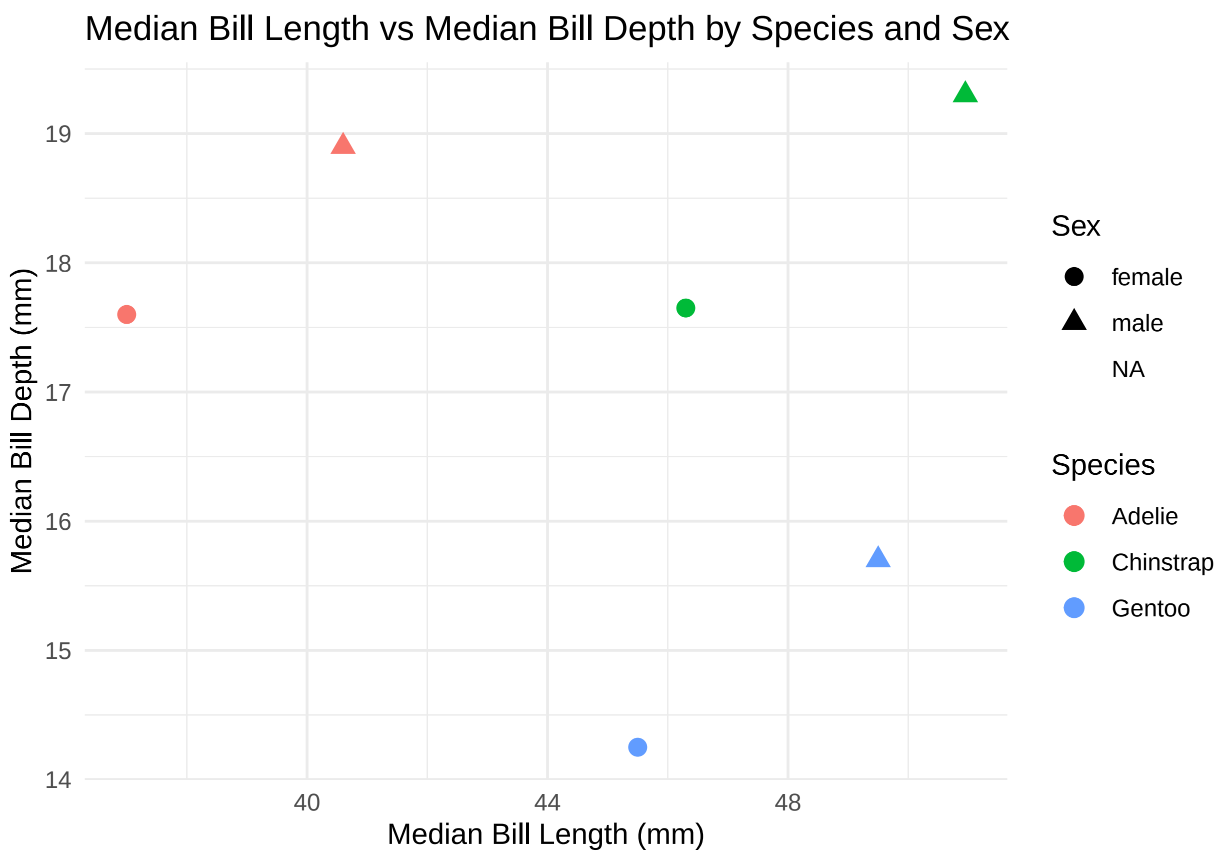
4. For each species and sex, calculate the median of the numeric variables. Then use an appropriate plot to show median bill length vs median bill depth by species and sex. Interpret the plot in one sentence

```
numeric_median_data <- penguins %>%
  group_by(species, sex) %>%
  summarize(median_bill_length = median(bill_length_mm, na.rm = TRUE), median_bill_depth = median(bill_
```

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

```
ggplot(numeric_median_data, aes(x = median_bill_length, y = median_bill_depth, color = species, shape =
```

```
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```




```r
paste("Across all of the diffrent species of peguins, Females have both a greater median bill length and
```

```
## [1] "Across all of the diffrent species of peguins, Females have both a greater median bill length an
```

```r
print(numeric_median_data)
```

```
## # A tibble: 8 x 6
## # Groups:   species [3]
##   species   sex    median_bill_length median_bill_depth median_flipper_length
##   <fct>     <fct>               <dbl>             <dbl>                 <dbl>
## 1 Adelie    female               37                17.6                  188
## 2 Adelie    male                 40.6              18.9                  193
## 3 Adelie    <NA>                 37.8              18.1                  186
## 4 Chinstrap female               46.3              17.6                  192
## 5 Chinstrap male                 51.0              19.3                  200.
## 6 Gentoo    female               45.5              14.2                  212
## 7 Gentoo    male                 49.5              15.7                  221
## 8 Gentoo    <NA>                 45.4              14.4                  216
## # i 1 more variable: median_body_mass <dbl>
```

5. How many rows have no missing values?

```r
non_missing_rows <- penguins %>% filter_all(all_vars(!is.na(.)))


count_non_missing_rows <- nrow(non_missing_rows)


paste("The number of rows with no missing values is:", count_non_missing_rows)
```

```
## [1] "The number of rows with no missing values is: 333"
```

6. How many unique values are there for each of the columns that end in "_mm" for each sex.

```r
unique_values <- penguins %>%
  group_by(sex) %>%
  summarize(across(ends_with("_mm"), ~ length(unique(.))))

print(unique_values)
```

```
## # A tibble: 3 x 4
##   sex    bill_length_mm bill_depth_mm flipper_length_mm
##   <fct>           <int>         <int>             <int>
## 1 female             97            56                41
## 2 male              110            58                49
## 3 <NA>                8            10                 9
```