# Exam 1

Christopher Hainzl

## Academic Honesty Statement

I, Christopher Hainzl, hereby state that I have not communicated with or gained information in any way from my classmates or anyone other than the Professor during this exam, and that all work is my own.

## Load packages

```
# load required packages here
library(readxl)
library(readr)
library(dplyr)
library(tidyr)
library(tidyverse)
library(nycflights13)
```

## Logistics

Add your code and/or narrative in the spaces below each question. Add code chunks as needed. Use as many lines as you need, but keep your narrative concise.

## Packages

In addition to `tidyverse`, you will need the `nycflights13` package for the data. You will first need to install these packages and then load them. Feel free to use any other library.

## The Data

1. The `nycflights13` package contains information about all flights that departed from NYC (e.g. EWR, JFK and LGA) in 2013. The main data is in the `flights` data frame, but there are additional data sets which may help understand what causes delays, specifically:

- weather: hourly meteorological data for each airport
- planes: construction information about each plane
- airports: airport names and locations
- airlines: translation between two letter carrier codes and names

2. The dataset `oslo-biomarkers.xlsx` contains data from a medical study about patients with disc herniations, performed at the Oslo University Hospital, Ullevål. Blood samples were collected from a number of patients with disc herniations at three time points: 0 weeks (first visit at the hospital), 6 weeks and 12 months. The levels of some biomarkers related to inflammation were measured in each blood sample.

3. The dataset `oslo-covariates.xlsx` contains information about the patients, such as age, gender and smoking habits.

## Questions

### Question 1

a.What does it mean for data analysis to be "reproducible"?

For a data analysis to be "reproducible" means that all the required data, software source code, and tools are distributed in a way that enables the reproduction of results discussed in a research publication.

b. Discuss the toolkit for reproducibility.

- Automation: repetitive tasks can be automated with scripts (which can be written in R)
- Literate Programming: Combining code snippets with a language that can be read by humans (which can be done with R Markdown)
- Version Control: Keeping track of changes to your work documents (which can be done using Git)

### Question 2

a. What is Exploratory data analysis (EDA)?

Exploratory data analysis is when visualization, transformation, and modeling are used to explore data. (Cited from "R for Data Science" by Hadley Wickham and Garrett Grolemund)

b. Why do we visualize data? Explain using Anscombe's quartet as a case study.

We visualize data so that we can detect differences in how our data is distributed when summary statistics do not seem to be enough. For example, when considering Anscombe's quartet, the summary statistics for all four sets of data are identical to one another. However, once we visualize those sets, we notice that the distribution of their data is different from each other (despite the summary statistics suggesting otherwise).

### Question 3

a. Write a function called `fahr_to_celsius` that converts temperatures from Fahrenheit to Celsius. Print out the results when the temperature is $32^oF$.

In my function, I used the formula for converting from Fahrenheit to Celsius (which involves subtracting 32 from the temperature in Fahrenheit, and then multiplying that difference by 5/9).

```
fahr_to_celsius <- function(x){
  # convert from Fahrenheit to Celsius
  (x - 32) * (5/9)
}
# should be equal to 0
fahr_to_celsius(32)
```

```
## [1] 0
```

b. Write another function called `celsius_to_kelvin` that converts Celsius into Kelvin. Print out the results when the temperature is $0^oC$.

In my function, I used the formula for converting from Celsius to Kelvin (which involves adding 273.15 to the temperature in Celsius).

```
celsius_to_kelvin <- function(x){
  # convert from Celsius to Kelvin
  x + 273.15
}
# should be equal to 273.15
celsius_to_kelvin(0)
```

```
## [1] 273.15
```

c. Compose the two functions in Parts 3a. and 3b. into a new function called `fahr_to_kelvin` that converts Fahrenheit to Kelvin. Print out the results when the temperature is $32^oF$.

To simplify this part, I called up my previous two functions and stored the results of those calls in two different variables.

```
fahr_to_kelvin <- function(x){
  # convert from Fahrenheit to Celsius
  first_conversion <- fahr_to_celsius(x)
  # convert from Celsius to Kelvin
  final_result <- celsius_to_kelvin(first_conversion)
}
# should be equal to 273.15
print(fahr_to_kelvin(32))
```

```
## [1] 273.15
```

** For Questions 4,5, and 6, use the `nycflights13` package **

### Question 4

What are the ten most common destinations for flights from NYC airports in 2013? Make a table that lists these in descending order of frequency and shows the number of flight heading to each airport.

To find the ten most common destinations, we need to determine how many times each destination appears in the dataframe, and arrange them in descending order. This can be done using the count(), arrange(), and desc() functions.

```
# Use count() function to keep track of how many times
# each destination appears in the dataframe
top_10 <- flights %>% count(dest) %>% arrange(desc(n)) %>% slice(1:10)
top_10
```

```
## # A tibble: 10 x 2
##    dest      n
##    <chr> <int>
##  1 ORD   17283
##  2 ATL   17215
##  3 LAX   16174
##  4 BOS   15508
##  5 MCO   14082
##  6 CLT   14064
##  7 SFO   13331
##  8 FLL   12055
##  9 MIA   11728
## 10 DCA    9705
```

The ten most common destinations are ORD, ATL, LAX, BOS, MCO, CLT, SFO, FLL, MIA, and DCA.

### Question 5

Which airlines have the most flights departing from NYC airports in 2013? Make a table that lists these in descending order of frequency and shows the number of flights for each airline. In your narrative mention the names of the airlines as well. Hint: You can use the `airlines` dataset to look up the airline name based on carrier code.

Just like with the top ten destinations, we can also use the count(), arrange(), and desc() functions.

```
# Same as before, except now we have to keep track of how many
# times each airline appears in the dataframe
airline_flights <- flights %>% count(carrier) %>% arrange(desc(n))
airline_flights
```

```
## # A tibble: 16 x 2
##    carrier     n
##    <chr>   <int>
##  1 UA      58665
##  2 B6      54635
##  3 EV      54173
##  4 DL      48110
##  5 AA      32729
##  6 MQ      26397
##  7 US      20536
##  8 9E      18460
##  9 WN      12275
## 10 VX       5162
## 11 FL       3260
## 12 AS        714
## 13 F9        685
## 14 YV        601
## 15 HA        342
## 16 OO         32
```

The top 5 airlines with the most flights departing from NYC are United Air Lines Inc., JetBlue Airways, ExpressJet Airlines Inc., Delta Air Lines Inc., and American Airlines Inc.

### Question 6

Consider only flights departing from New York on 1 January that year.

a. Are there missing data?

To determine if there is any missing data, I can use the summary() function once I have filtered my data.

```
# filter for flights departing on January 1
jan_first <- flights %>% filter(month == 1, day == 1)
# check for any missing data by using summary() function
summary(jan_first)
```

```
##       year          month        day       dep_time      sched_dep_time
##  Min.   :2013   Min.   :1   Min.   :1   Min.   : 517.0   Min.   : 515.0
##  1st Qu.:2013   1st Qu.:1   1st Qu.:1   1st Qu.: 940.2   1st Qu.: 944.2
##  Median :2013   Median :1   Median :1   Median :1439.5   Median :1430.0
##  Mean   :2013   Mean   :1   Mean   :1   Mean   :1385.0   Mean   :1372.4
##  3rd Qu.:2013   3rd Qu.:1   3rd Qu.:1   3rd Qu.:1756.8   3rd Qu.:1730.0
##  Max.   :2013   Max.   :1   Max.   :1   Max.   :2356.0   Max.   :2359.0
##                                         NA's   :4
##    dep_delay         arr_time    sched_arr_time   arr_delay
##  Min.   :-15.00   Min.   :   3   Min.   :   5   Min.   :-48.00
##  1st Qu.: -4.00   1st Qu.:1152   1st Qu.:1209   1st Qu.: -8.00
##  Median : -1.00   Median :1633   Median :1620   Median :  3.00
##  Mean   : 11.55   Mean   :1562   Mean   :1568   Mean   : 12.65
##  3rd Qu.:  9.00   3rd Qu.:2006   3rd Qu.:1953   3rd Qu.: 19.00
##  Max.   :853.00   Max.   :2400   Max.   :2359   Max.   :851.00
```

```
##  NA's   :4        NA's   :5                    NA's   :11
##    carrier              flight         tailnum             origin
##  Length:842         Min.   :   1.0   Length:842         Length:842
##  Class :character   1st Qu.: 505.5   Class :character   Class :character
##  Mode  :character   Median :1203.5   Mode  :character   Mode  :character
##                     Mean   :1821.5
##                     3rd Qu.:3367.8
##                     Max.   :5742.0
##
##      dest              air_time        distance         hour
##  Length:842         Min.   : 24.0   Min.   :  94   Min.   : 5.00
##  Class :character   1st Qu.:102.0   1st Qu.: 529   1st Qu.: 9.00
##  Mode  :character   Median :149.0   Median : 946   Median :14.00
##                     Mean   :169.7   Mean   :1077   Mean   :13.47
##                     3rd Qu.:229.0   3rd Qu.:1400   3rd Qu.:17.00
##                     Max.   :659.0   Max.   :4983   Max.   :23.00
##                     NA's   :11
##      minute        time_hour
##  Min.   : 0.00   Min.   :2013-01-01 05:00:00.00
##  1st Qu.: 8.00   1st Qu.:2013-01-01 09:00:00.00
##  Median :29.00   Median :2013-01-01 14:00:00.00
##  Mean   :25.81   Mean   :2013-01-01 13:27:56.00
##  3rd Qu.:44.00   3rd Qu.:2013-01-01 17:00:00.00
##  Max.   :59.00   Max.   :2013-01-01 23:00:00.00
##
```

Yes, there is some missing data.

b. Are there any differences between the different carriers? Focusing on delays and the amount of time spent in the air.

For this exercise, I chose to compare the carriers based on their average delays and average amount of time spent in the air.

```
# summary statistics involving average departure delay,
# average arrival delay, and average time spent in the air
differences <- jan_first %>% group_by(carrier) %>% summarise(average_dep_delay = mean(dep_delay, na.rm =
                                                             average_arr_delay = mean(arr_delay, na.rm =
                                                            average_air_time = mean(air_time, na.rm = TRU

differences
```

```
## # A tibble: 14 x 4
##    carrier average_dep_delay average_arr_delay average_air_time
##    <chr>               <dbl>             <dbl>            <dbl>
##  1 9E                 17.6               12.5             95.3
##  2 AA                  7.96              11.4            210.
##  3 AS                 -4                -14.5            337
##  4 B6                 10.5                8.64           164.
##  5 DL                 -0.0625            -7.58           185.
##  6 EV                 33.3               41.4             94.0
##  7 F9                 -8                 13              250.
##  8 FL                 -5.1                5.3            120.
##  9 HA                 -3                -14              659
## 10 MQ                 22.2               33.3            108.
## 11 UA                  7.65               6.27           225.
## 12 US                 -2.09               1.16           140.
```

```
## 13 VX                -0.75             -12.2             354.
## 14 WN                 2.96              16.7             163.
```

- On average, ExpressJet Airlines Inc. has the highest departure delay, while Frontier Airlines Inc. has the lowest departure delay.
- On average, ExpressJet Airlines Inc. has the highest arrival delay, and Alaska Airlines Inc. has the lowest arrival delay.
- On average, Hawaiian Airlines Inc. has the highest air time, while ExpressJet Airlines Inc. has the lowest air time.

c. Are there any outliers? Focusing on delays and the amount of time spent in the air.

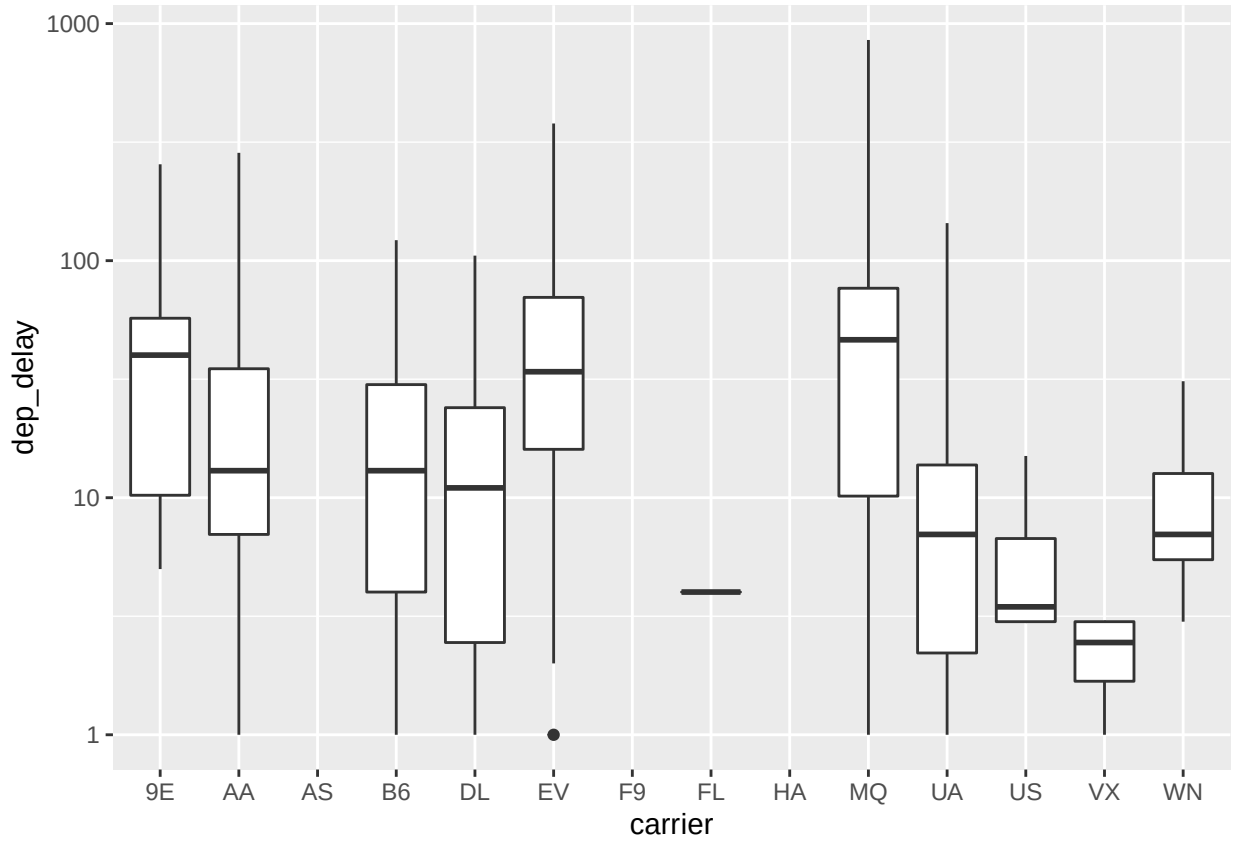
To determine if there are any outliers, we have to use a boxplot, with the carriers on the x-axis, and the delays / amount of time spent in the air on the y-axis.

```
# Use boxplots to detect any outliers in data
ggplot(data = jan_first, aes(x=carrier, y=dep_delay, group = carrier)) +
geom_boxplot() + scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 490 rows containing non-finite values (stat_boxplot).
```

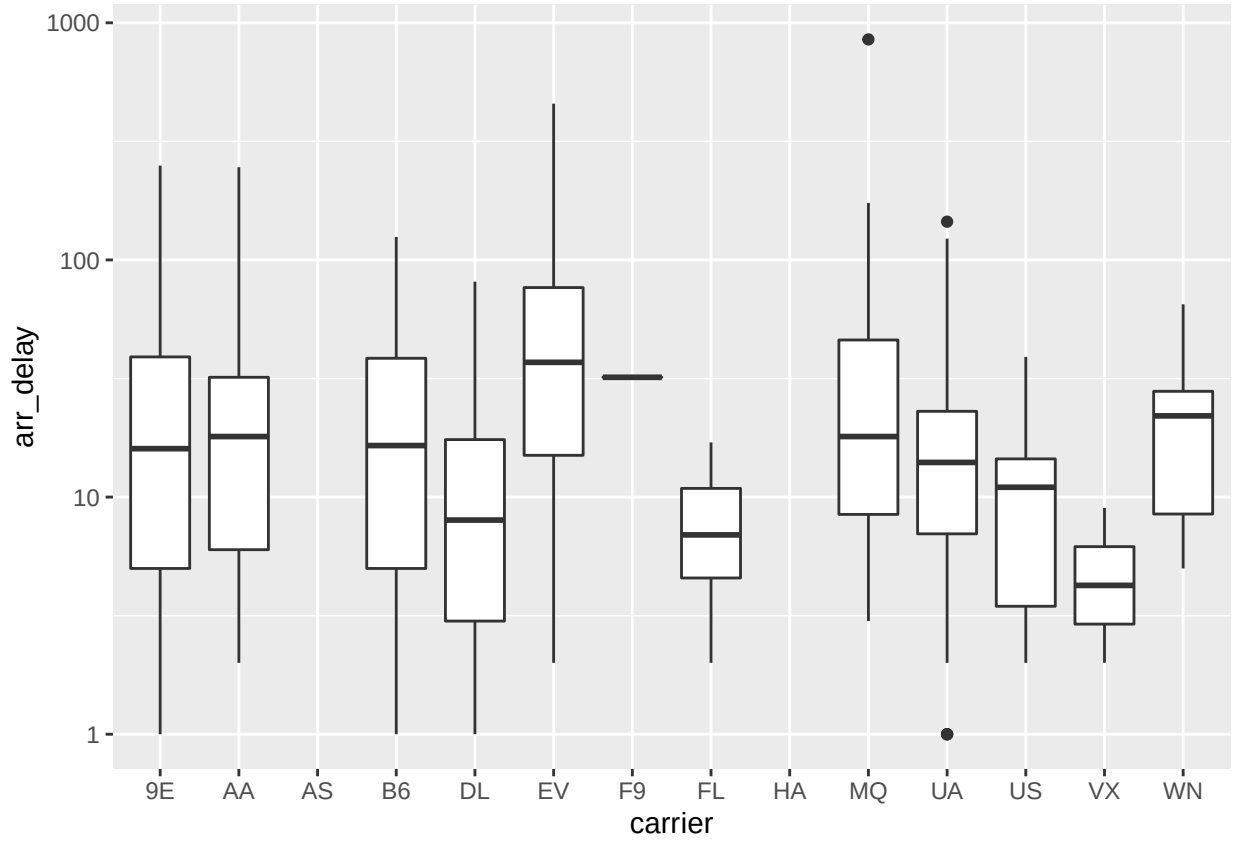



```
ggplot(data = jan_first, aes(x=carrier, y=arr_delay, group = carrier)) +
geom_boxplot() + scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced
```

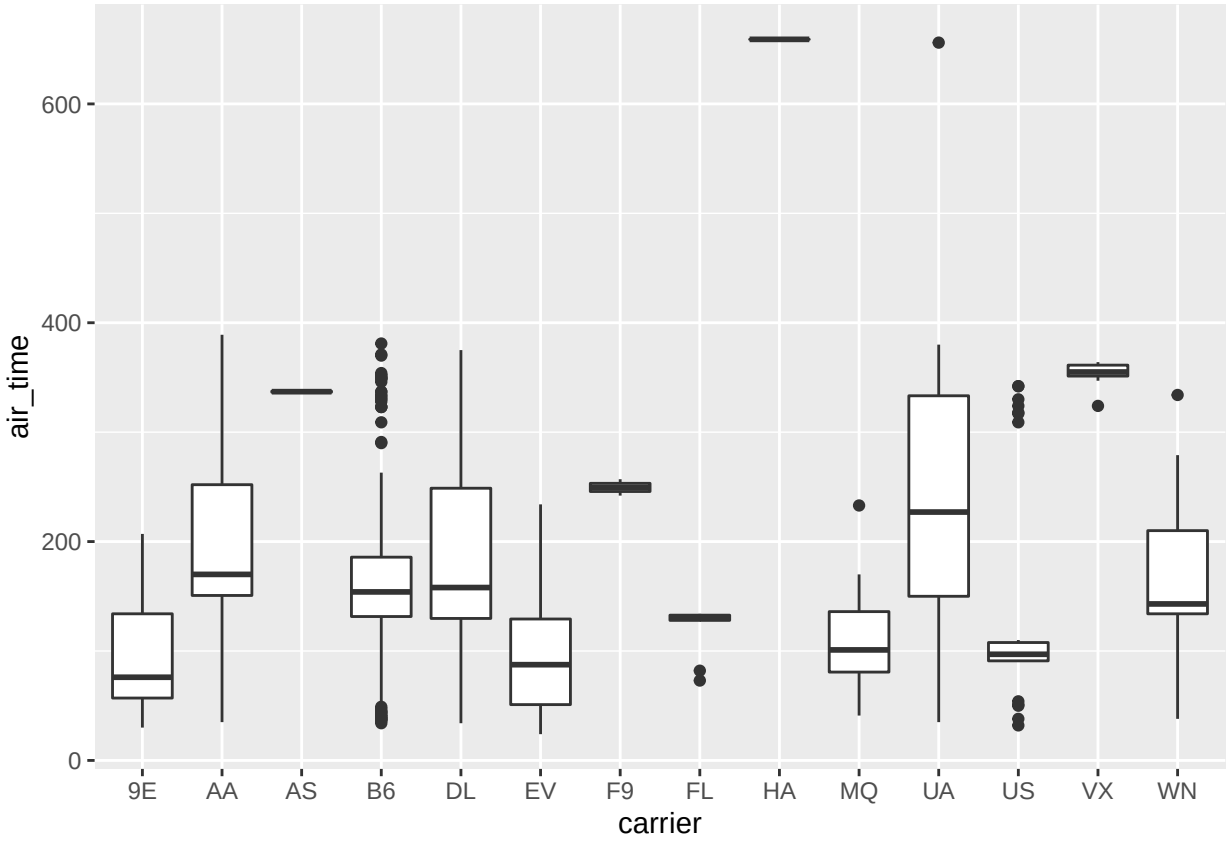
```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 381 rows containing non-finite values (stat_boxplot).
```




```
ggplot(data = jan_first, aes(x=carrier, y=air_time, group = carrier)) +
geom_boxplot()
```

```
## Warning: Removed 11 rows containing non-finite values (stat_boxplot).
```




Yes, there are outliers when considering the data involving departure / arrival delays and amount of time in the air.

**Question 7**

Load the `oslo-biomarkers.xlsx data`. Then do the following using dplyr/tidyr:

```
oslo_biomarkers <- read_excel("oslo-biomarkers.xlsx")
oslo_covariates <- read_excel("oslo-covariates.xlsx")
```

a. Split the `PatientID.timepoint` column in two parts: one with the patient ID and one with the timepoint.

We need to use the separate() function to split the 'PatientID.timepoint' column into two separate columns, with the '-' symbol as our separator. Once that is done, we then have to convert the 'PatientID' column to a column of type 'numeric' (since it was originally part of a column of type 'character'). This can be done using the as.numeric() function.

```
# Use separate() function to split one column into
# two separate columns
oslo_biomarkers_new <- oslo_biomarkers %>%
separate(col = "PatientID.timepoint", into = c("PatientID", "Timepoint"), sep = "-")
# Using the as.numeric() function will benefit us later on
oslo_biomarkers_new$PatientID <- as.numeric(oslo_biomarkers_new$PatientID)
# Inspiration taken from: #https://stackoverflow.com/questions/7069076/split-column-at-delimter-in-data
oslo_biomarkers_new
```

```
## # A tibble: 347 x 11
##    PatientID Timepoint `IL-8` `VEGF-A`   OPG TGF-be~1 `IL-6` CXCL9 CXCL1 `IL-18`
```

```
##        <dbl> <chr>      <dbl>    <dbl> <dbl>    <dbl>  <dbl> <dbl> <dbl>   <dbl>
##  1       126 0weeks      7.64     11.5  10.2     8.85   3.53  6.19  9.5     7.91
##  2       126 6weeks      7.09     11.6  10.4     8.86   3.89  6.1   9.07    7.92
##  3       127 0weeks      6.92     10.9  10.3     6.61   2.69  6.17  7.28    7.98
##  4       127 6weeks      7.18     11.6  10.4     8.57   2.59  6.35  8.6     7.93
##  5       127 12months    6.89     11.2  10.2     7.46   3.96  6.17  8.8     7.97
##  6       128 0weeks      8.57     12.4  10.6     8.49   3.71  7.34  9.9     8.72
##  7       128 6weeks      6.94     11.6  10.6     7.44   3.86  7.16  8.57    8.63
##  8       128 12months    6.49     11.0  10.1     6.41   4.62  7.98  8.17    8.68
##  9       129 0weeks      8.16     11.1  10.6     8.81   3.84  5.82  9.17    7.51
## 10       129 6weeks      6.52     10.7  10.2     6.8    2.96  6.16  6.71    7.28
## # ... with 337 more rows, 1 more variable: `CSF-1` <dbl>, and abbreviated
## #   variable name 1: `TGF-beta-1`
```

b. Sort the table by patient ID, in numeric order.

Now that the 'PatientID' column has been converted into a column of type 'numeric' we can sort the table in numeric order using the arrange() function.

```
# sort using arrange() function
oslo_biomarkers_ordered <- oslo_biomarkers_new %>% arrange(PatientID)
oslo_biomarkers_ordered
```

```
## # A tibble: 347 x 11
##    PatientID Timepoint `IL-8` `VEGF-A`   OPG TGF-be~1 `IL-6` CXCL9 CXCL1 `IL-18`
##        <dbl> <chr>      <dbl>    <dbl> <dbl>    <dbl>  <dbl> <dbl> <dbl>   <dbl>
##  1         1 0weeks      8.16     12.3  10.5     8.68   2.6   6.55  9.61    8.52
##  2         1 6weeks      7.11     11.6  10.3     7.57   2.28  7.83  9.47    8.39
##  3         1 12months    8.6      12.5  10.7     8.5    2.57  6.68  9.62    8.79
##  4         3 0weeks      6.56     11.2  10.5     6.85   2.55  5.37  6.69    7.7
##  5         3 6weeks      6.33     11.2  10.6     6.63   2.26  5.51  6.67    7.82
##  6         3 12months    7.44     11.7  10.7     7.82   2.73  5.53  8.78    7.77
##  7         4 0weeks      6.42     11.1  10.8     6.95   5.64  5.44  7.74    8
##  8         4 6weeks      7.71     11.5  11.0     7.74   5.14  5.63  8.32    8.15
##  9         4 12months    7.24     11.5  10.9     7.37   4.33  5.72  8.28    8.21
## 10         5 0weeks      6.42     11.2  10.7     7.26   1.92  5.29  7.14    7.68
## # ... with 337 more rows, 1 more variable: `CSF-1` <dbl>, and abbreviated
## #   variable name 1: `TGF-beta-1`
```

c. Reformat the data from long to wide, keeping only the `IL-8` and `VEGF-A` measurements.

To reformat our data, we need to use 'PatientID' as our id column, 'Timepoint' to get the names of the new columns, and 'IL-8' & 'VEGF-A' to get our values. Luckily, R's pivot_wider() function makes it possible for us to do this.

```
oslo_biomarkers_reformatted <- oslo_biomarkers_ordered %>%
  pivot_wider(id_cols = PatientID, names_from = Timepoint, values_from = c('IL-8','VEGF-A'))
oslo_biomarkers_reformatted
```

```
## # A tibble: 118 x 7
##    PatientID `IL-8_0weeks` `IL-8_6weeks` `IL-8_12months` VEGF-~1 VEGF-~2 VEGF-~3
##        <dbl>         <dbl>         <dbl>           <dbl>   <dbl>   <dbl>   <dbl>
##  1         1          8.16          7.11            8.6     12.3    11.6    12.5
##  2         3          6.56          6.33            7.44    11.2    11.2    11.7
##  3         4          6.42          7.71            7.24    11.1    11.5    11.5
##  4         5          6.42          6.29            7.5     11.2    11.2    11.8
##  5         6          6.54          6.57            6.66    11.5    11.4    11.2
```

```
##  6         7          6.93          6.4            6.65    11.0    11.0    11.0
##  7         8          8.28          7.29           8.82    12.8    11.5    13.3
##  8         9          9.02          7.62           6.67    12.6    11.8    11.6
##  9        13          8.66          8.68           7.83    12.2    12.5    11.8
## 10        14          7.79          6.99           6.74    11.3    10.9    10.9
## # ... with 108 more rows, and abbreviated variable names 1: `VEGF-A_0weeks`,
## #   2: `VEGF-A_6weeks`, 3: `VEGF-A_12months`
```

d. Merge the wide data frame from part c. with the `oslo-covariates.xlsx` data, using patient ID as key.

We can use left join the dataframe from part c with 'oslo-covariates.xlsx' to combine them together and keep all the observations from our wide dataframe.

```
# use the left_join() function
oslo_merged <- oslo_biomarkers_reformatted %>% left_join(oslo_covariates)
```

```
## Joining, by = "PatientID"
```

```
oslo_merged
```

```
## # A tibble: 118 x 12
##    Patie~1 IL-8_~2 IL-8_~3 IL-8_~4 VEGF-~5 VEGF-~6 VEGF-~7   Age Sex (~8 Smoke~9
##      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>   <dbl>
##  1       1    8.16    7.11    8.6     12.3    11.6    12.5    55       2       2
##  2       3    6.56    6.33    7.44    11.2    11.2    11.7    32       1       2
##  3       4    6.42    7.71    7.24    11.1    11.5    11.5    33       2       2
##  4       5    6.42    6.29    7.5     11.2    11.2    11.8    25       2       1
##  5       6    6.54    6.57    6.66    11.5    11.4    11.2    39       1       2
##  6       7    6.93    6.4     6.65    11.0    11.0    11.0    38       2       2
##  7       8    8.28    7.29    8.82    12.8    11.5    13.3    48       2       2
##  8       9    9.02    7.62    6.67    12.6    11.8    11.6    43       2       1
##  9      13    8.66    8.68    7.83    12.2    12.5    11.8    54       2       1
## 10      14    7.79    6.99    6.74    11.3    10.9    10.9    41       1       2
## # ... with 108 more rows, 2 more variables: `VAS-at-inclusion` <dbl>,
## #   `Vas-12months` <dbl>, and abbreviated variable names 1: PatientID,
## #   2: `IL-8_0weeks`, 3: `IL-8_6weeks`, 4: `IL-8_12months`, 5: `VEGF-A_0weeks`,
## #   6: `VEGF-A_6weeks`, 7: `VEGF-A_12months`, 8: `Sex (1=male, 2=female)`,
## #   9: `Smoker (1=yes, 2=no)`
```

e. Use the `oslo-covariates.xlsx` data to select data for smokers from the wide data frame in part c. First, we have to filter out the data in 'oslo-covariates.xlsx' so it only shows the patients who are smokers. Then, we join the wide data frame in part c with the new, filtered one created in this part.

```
covariate_smoker_data <- oslo_covariates %>%
  # 1 means that the patient is a smoker
filter(oslo_covariates$'Smoker (1=yes, 2=no)' == 1)
  # join both dataframes using inner_join()
oslo_biomarkers_reformatted %>% inner_join(covariate_smoker_data)
```

```
## Joining, by = "PatientID"
```

```
## # A tibble: 39 x 12
##    Patie~1 IL-8_~2 IL-8_~3 IL-8_~4 VEGF-~5 VEGF-~6 VEGF-~7   Age Sex (~8 Smoke~9
##      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>   <dbl>
##  1       5    6.42    6.29    7.5     11.2    11.2    11.8    25       2       1
##  2       9    9.02    7.62    6.67    12.6    11.8    11.6    43       2       1
##  3      13    8.66    8.68    7.83    12.2    12.5    11.8    54       2       1
##  4      18    7.31    7.26    6.27    11.4    11.3    11.1    27       1       1
```

```
##  5      19   11.3     6.67    7.09    12.1    11.4    11.3    39       2       1
##  6      23    8.85    8.43    8.66    12.7    12.9    12.6    52       2       1
##  7      26    7.12    8.31    6.47    11.1    12.6    11.1    36       1       1
##  8      31    7.92    8.21    7.52    12.5    12.0    11.8    52       1       1
##  9      32    8.25    6.8     7.11    11.8    11.2    11.0    31       1       1
## 10      37    8.72    6.57    7.09    12.2    11.2    11.1    39       1       1
## # ... with 29 more rows, 2 more variables: `VAS-at-inclusion` <dbl>,
## #   `Vas-12months` <dbl>, and abbreviated variable names 1: PatientID,
## #   2: `IL-8_0weeks`, 3: `IL-8_6weeks`, 4: `IL-8_12months`, 5: `VEGF-A_0weeks`,
## #   6: `VEGF-A_6weeks`, 7: `VEGF-A_12months`, 8: `Sex (1=male, 2=female)`,
## #   9: `Smoker (1=yes, 2=no)`
```