

K20041606_CW2

Jude Popham, *K20041606*

2022-12-27

Section 1: Visualising data with R

For these sets of questions we will use the dataset “cell.measurements.csv”. This dataset contains measurements depicting cell shape and cell movements at different time points. The aim of the study is to understand if addition of treatment alter cell shape or movement in the context of injury. So there are:

- 11 measurement
- 6 animals: 3 treated and 3 not treated (control)
- several time points (time_frame)

For each animal and each time point, measurements have been collected with and without Injury. Second and third column contain info about injury (1 presence of injury; 0 absence of Injury) and treatment (1 presence of treatment; 0 absence of treatment).

```
#Load the libraries
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   1.0.0
## v tibble  3.1.8     v dplyr    1.0.10
## v tidyr   1.2.1     v stringr  1.5.0
## v readr   2.1.3     vforcats 0.5.2
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()   masks stats::lag()

library(dplyr)
library(hexbin)
library(ggplot2)
```

Exercise V1

Question:

Visualise the distribution of each measurement within each animal, showing behavior in presence or absence of Injury. Discuss which measurement has the most variability between animals, in presence and absence of Injury. Can you see any apparent difference between treated animals vs the control ones?

Code:

To create two distribution plots, one for healthy and one for injured animals. Within each plot, there are 11 different box plots, showing distribution of all 11 measurements. These box plots detail the 6 animal types across the x axis ('Animal') and their corresponding distribution of the measurement on the y axis ('value').

```
#Load the dataframe
cell.measurements <- read.csv('/Users/judepops/Documents/Bioinformatics_Module/CW_2/CW_2_Data/RNAseq/cel

#Subset dataframe into presence of injury
cell.measurements.injured <- subset(cell.measurements, Injury == '1') %>% #I subset the dataframe using
  select(-c(Animal, Treatment, time_frame)) #select - is the inverse. I used this to remove
#all of the columns that are within the brackets of the select function as we only
#want the cell measurements, not the predictor variables.

cell.measurements.healthy <- subset(cell.measurements, Injury == '0') %>% #I subset the dataframe using
  select(-c(Animal, Treatment, time_frame)) #select - is the inverse. I used this to remove
#all of the columns that are within the brackets of the select function as we only
#want the cell measurements, not the predictor variables.

#Visualising the distribution of healthy cell measurements
#We do not need a log_10 scale because everything is comparably measured in size

cell.measurements.injured.distribution <- cell.measurements.injured %>%
  pivot_longer(-Animal) %>%
  mutate(Animal = factor(Animal)) %>%
  ggplot(aes(Animal, value)) + #inputting the aes into ggplot function
  geom_jitter(alpha = 0.3, color = 'tomato') + #alpha 0.3 changes the size of the
#noise added to the plot by geom jitter and the color specifies them as red
  geom_boxplot(aes(color = Animal)) + #this specifies the ggplot function
#as a boxplot and splits the animals by colour
  ggtitle('Injured_Distribution') + #adding a title for the graph
  theme(axis.text.x=element_blank()) + #adding a theme for the graph
  facet_wrap(~name) #facet wrapping the graph by the name

cell.measurements.healthy.distribution <- cell.measurements.healthy %>%
  pivot_longer(-Animal) %>%
  mutate(Animal = factor(Animal)) %>%
  ggplot(aes(Animal, value)) +
  geom_jitter(alpha = 0.3, color = 'tomato') +
  geom_boxplot(aes(color = Animal)) +
  ggtitle('Control_Distribution') +
  theme(axis.text.x=element_blank()) +
  facet_wrap(~name)
```

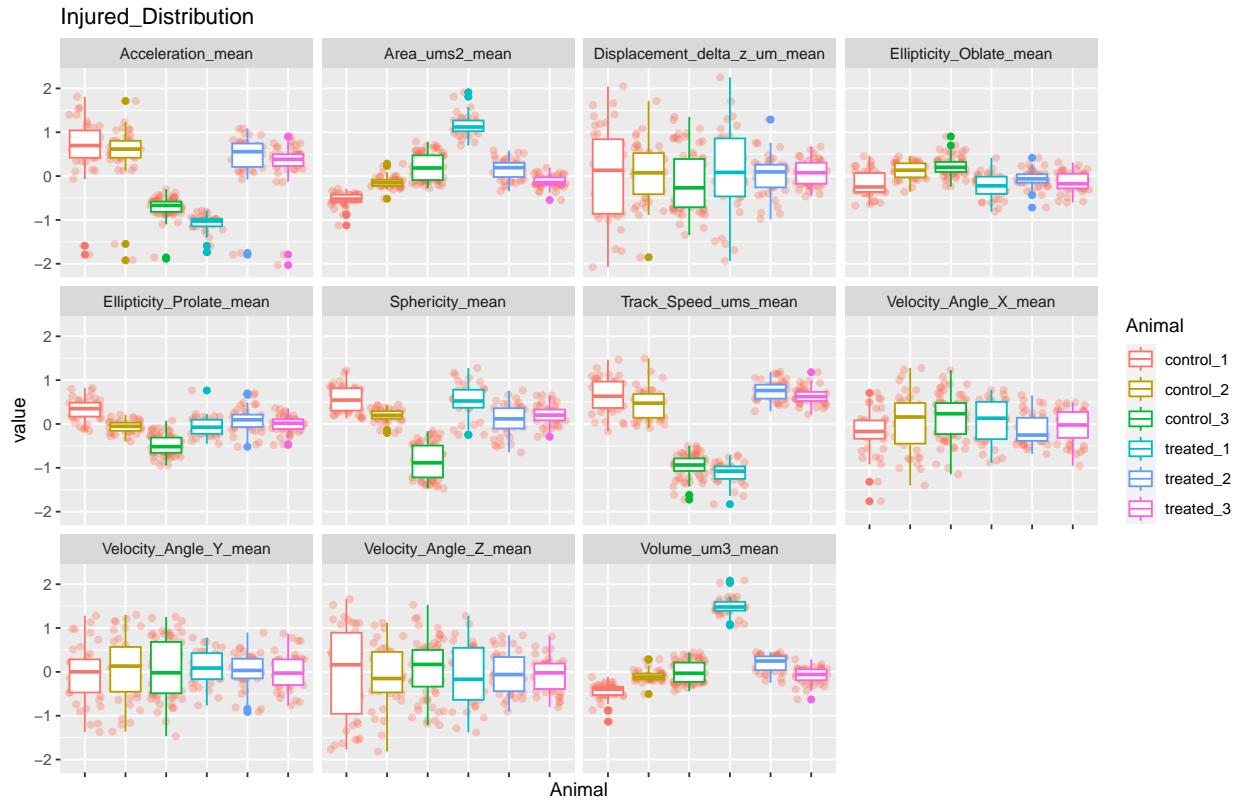
Plot:

Graphs showing distribution of each measurement within each animal for absence ('Control_Distribution') and presence ('Injured_Distribution') of injury:

```
cell.measurements.healthy.distribution
```



```
cell.measurements.injured.distribution
```



Discussion:

Which measurement has the most variability between animals in presence and absence of injury? Can you see any apparent difference between treated animals vs the control ones?

In the presence of injury (injured distribution), the measurement with the most variability is Displacement_delta_z_um_mean. This is because taller box plots imply more variable data: the medians are similar but there are very wide ranges (whisker length) which implies great variability.

In the absence of injury (control distribution), the measurement with the most variability is less obvious and is either Displacement_delta_z_um_mean or Velocity_Angle_Z_mean. They both have great variability in measurement with very wide ranges while the medians of each animal remain similar.

When comparing control and injury box plots, there tends to be an overall slight decrease in medians of the measurements and variability (box plot whisker length) appears to increase slightly. This means the injured animals tend to have more variability in measurement and slightly lower values in general.

Exercise V2

Question:

For each measurement, plot how the values change over time, discriminating between each animal and faceting based on the presence or absence of Injury. Discuss the final plot. Can you see similar behaviors across animals? Does time have a big effect on all or some of the measurements?

Code 1:

To create 11 individual graphs with color to visualise the data clearly

```
#Code to create individual graphs
Acceleration_mean_time_plot <- cell.measurements %>%
  mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
  group_by(Animal,time_frame) %>%
  ggplot(mapping = aes(x = time_frame, y = Acceleration_mean, color = Animal)) +
  geom_line() +
  facet_wrap(facets = vars(Injury)) +
  theme_minimal()

Velocity_Angle_X_mean_time_plot <- cell.measurements %>%
  mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
  group_by(Animal,time_frame) %>%
  ggplot(mapping = aes(x = time_frame, y = Velocity_Angle_X_mean, color = Animal)) +
  geom_line() +
  facet_wrap(facets = vars(Injury)) +
  theme_minimal()

Velocity_Angle_Y_mean_time_plot <- cell.measurements %>%
  mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
  group_by(Animal,time_frame) %>%
  ggplot(mapping = aes(x = time_frame, y = Velocity_Angle_Y_mean, color = Animal)) +
  geom_line() +
  facet_wrap(facets = vars(Injury)) +
  theme_minimal()

Velocity_Angle_Z_mean_time_plot <- cell.measurements %>%
  mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
  group_by(Animal,time_frame) %>%
  ggplot(mapping = aes(x = time_frame, y = Velocity_Angle_Z_mean, color = Animal)) +
  geom_line() +
  facet_wrap(facets = vars(Injury)) +
  theme_minimal()

Area_ums2_mean_time_plot <- cell.measurements %>%
  mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
  group_by(Animal,time_frame) %>%
  ggplot(mapping = aes(x = time_frame, y = Area_ums2_mean, color = Animal)) +
  geom_line() +
  facet_wrap(facets = vars(Injury)) +
  theme_minimal()

Displacement_delta_z_um_mean_time_plot <- cell.measurements %>%
  mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
  group_by(Animal,time_frame) %>%
  ggplot(mapping = aes(x = time_frame, y = Displacement_delta_z_um_mean, color = Animal)) +
  geom_line() +
  facet_wrap(facets = vars(Injury)) +
  theme_minimal()

Ellipticity_Oblate_mean_time_plot <- cell.measurements %>%
```

```

mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
group_by(Animal,time_frame) %>%
ggplot(mapping = aes(x = time_frame, y = Ellipticity_Oblate_mean, color = Animal)) +
geom_line() +
facet_wrap(facets = vars(Injury)) +
theme_minimal()

Ellipticity_Prolate_mean_time_plot <- cell.measurements %>%
mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
group_by(Animal,time_frame) %>%
ggplot(mapping = aes(x = time_frame, y = Ellipticity_Prolate_mean, color = Animal)) +
geom_line() +
facet_wrap(facets = vars(Injury)) +
theme_minimal()

Track_Speed_ums_mean_time_plot <- cell.measurements %>%
mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
group_by(Animal,time_frame) %>%
ggplot(mapping = aes(x = time_frame, y = Track_Speed_ums_mean, color = Animal)) +
geom_line() +
facet_wrap(facets = vars(Injury)) +
theme_minimal()

Sphericity_mean_time_plot <- cell.measurements %>%
mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
group_by(Animal,time_frame) %>%
ggplot(mapping = aes(x = time_frame, y = Sphericity_mean, color = Animal)) +
geom_line() +
facet_wrap(facets = vars(Injury)) +
theme_minimal()

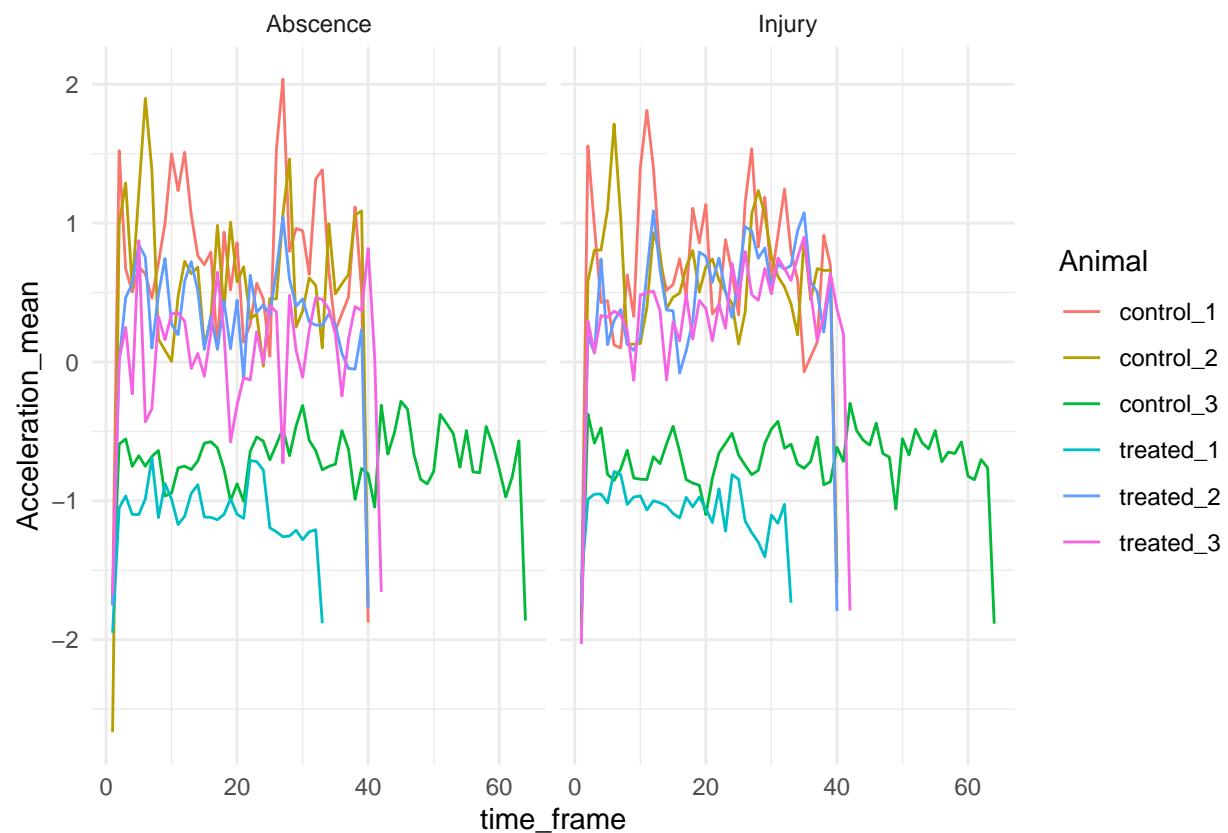
Volume_um3_mean_time_plot <- cell.measurements %>%
mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
group_by(Animal,time_frame) %>%
ggplot(mapping = aes(x = time_frame, y = Volume_um3_mean, color = Animal)) +
geom_line() +
facet_wrap(facets = vars(Injury)) +
theme_minimal()

```

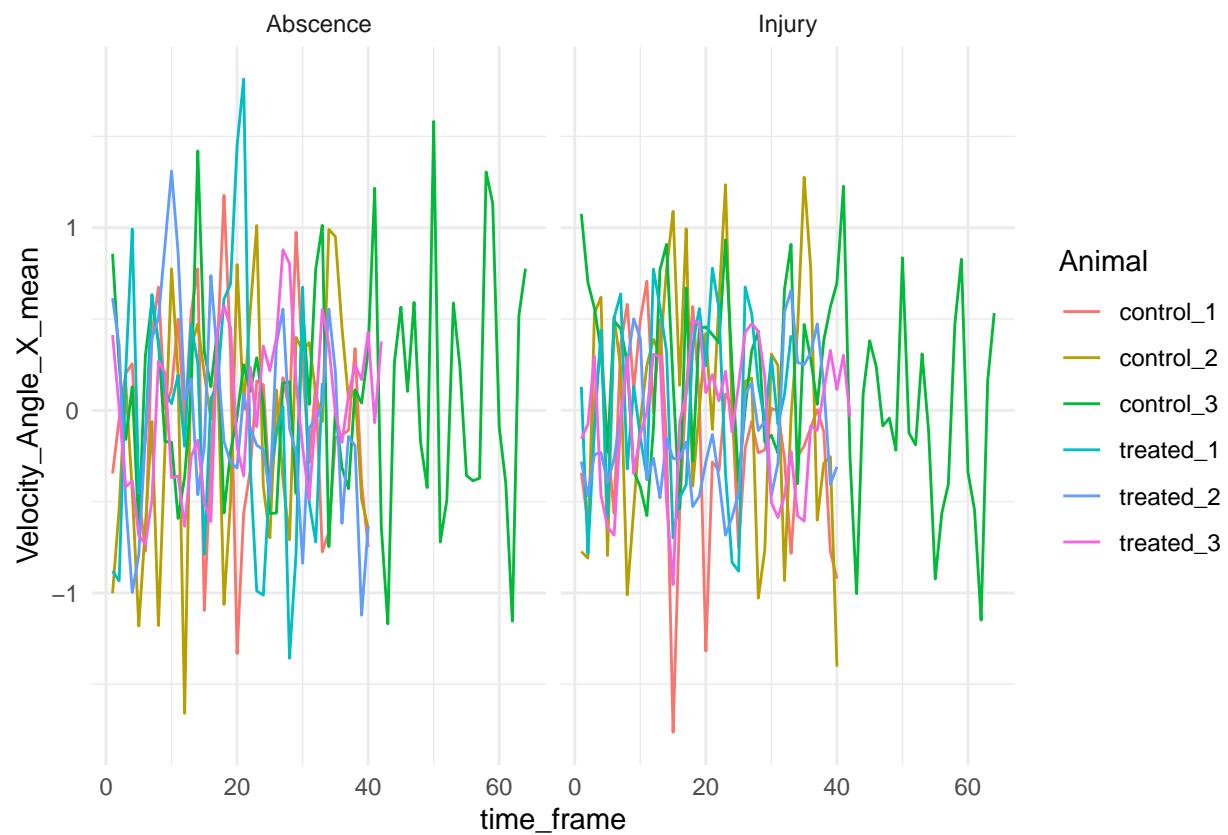
Plot 1:

11 individual graphs

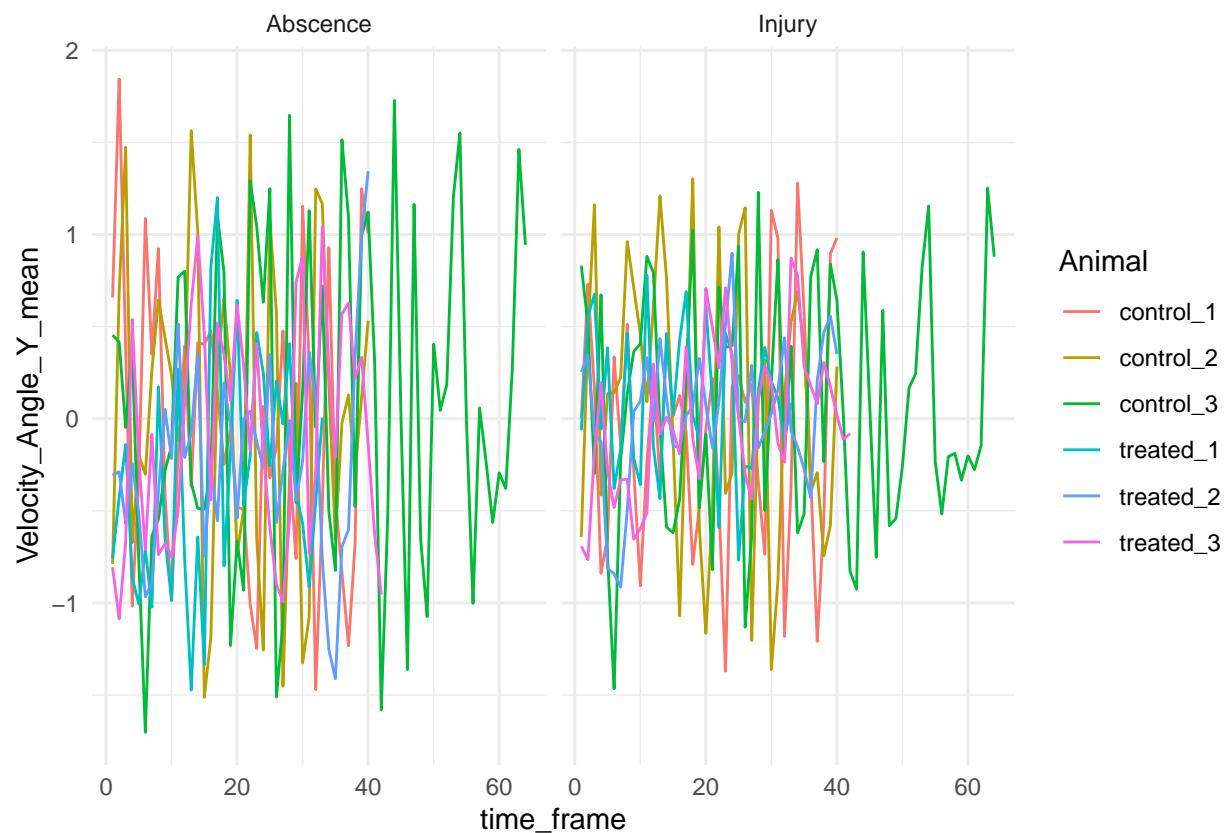
Acceleration_mean_time_plot



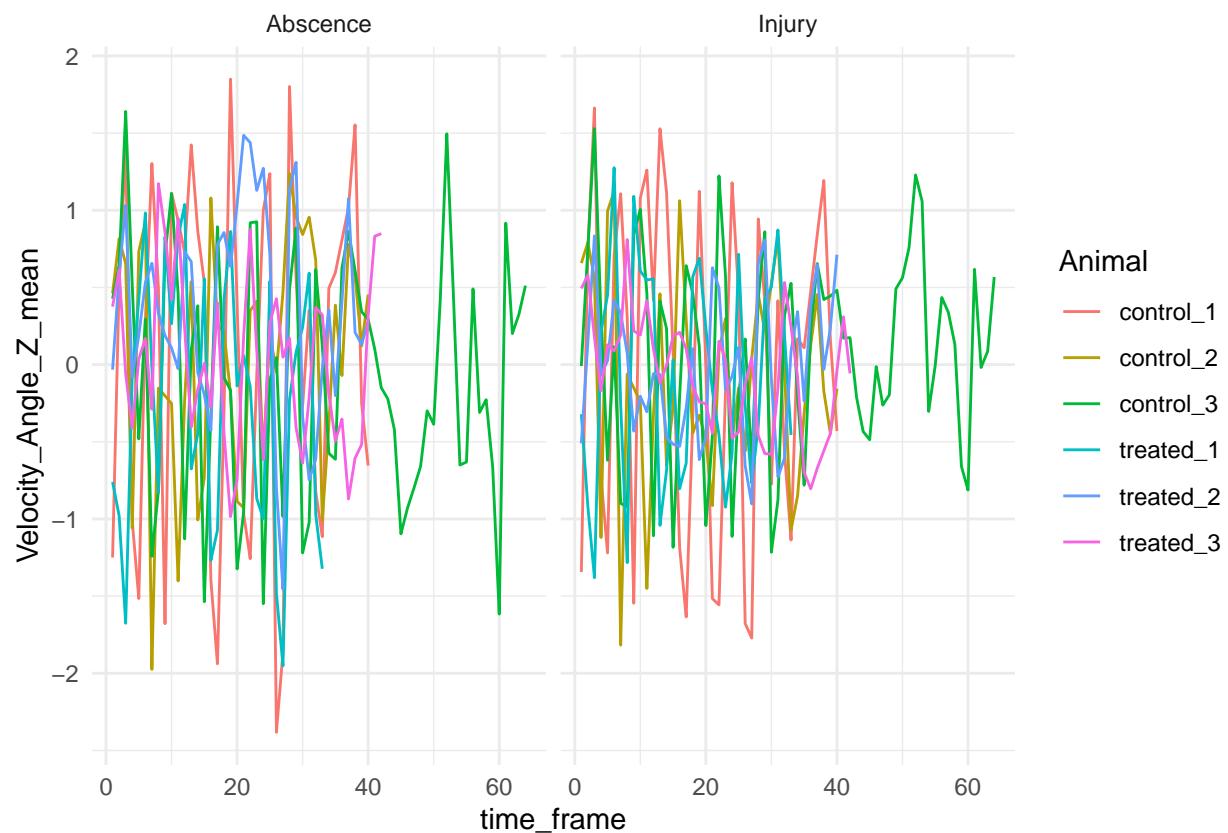
Velocity_Angle_X_mean_time_plot



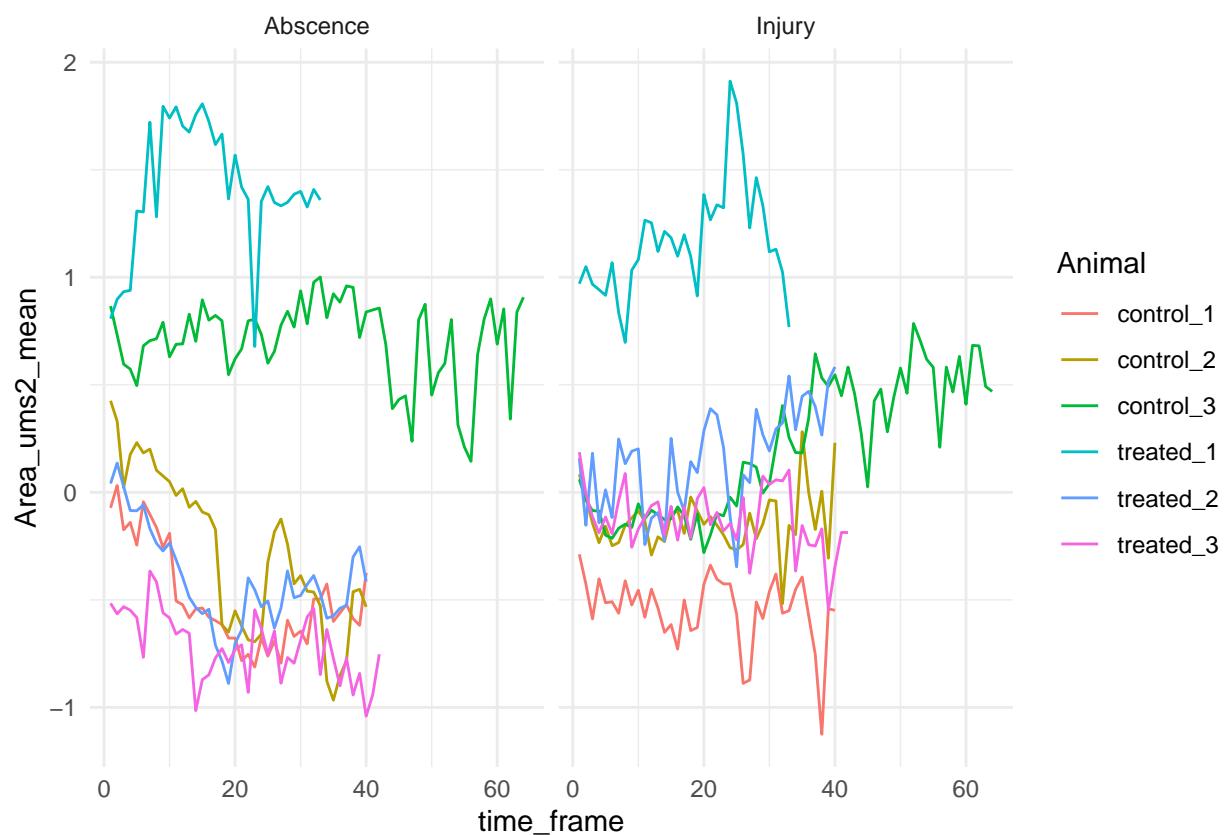
Velocity_Angle_Y_mean_time_plot



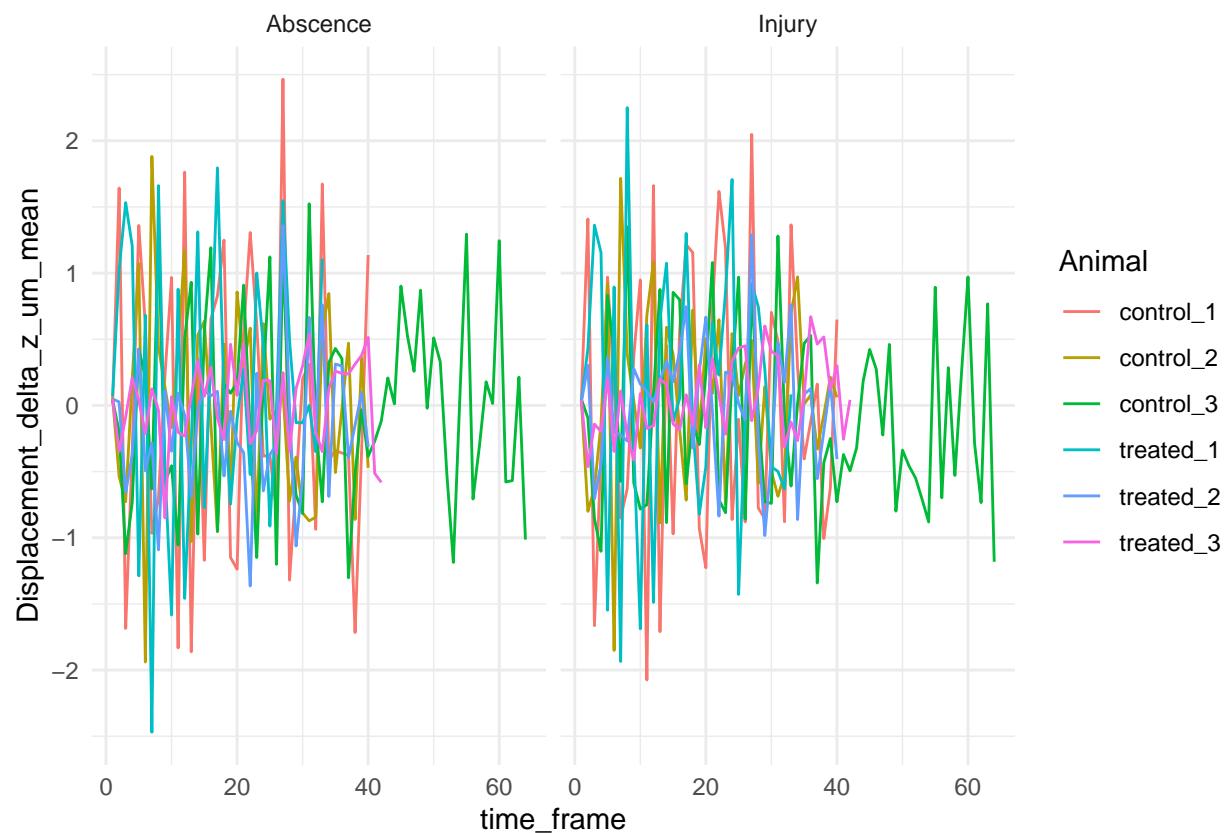
Velocity_Angle_Z_mean_time_plot



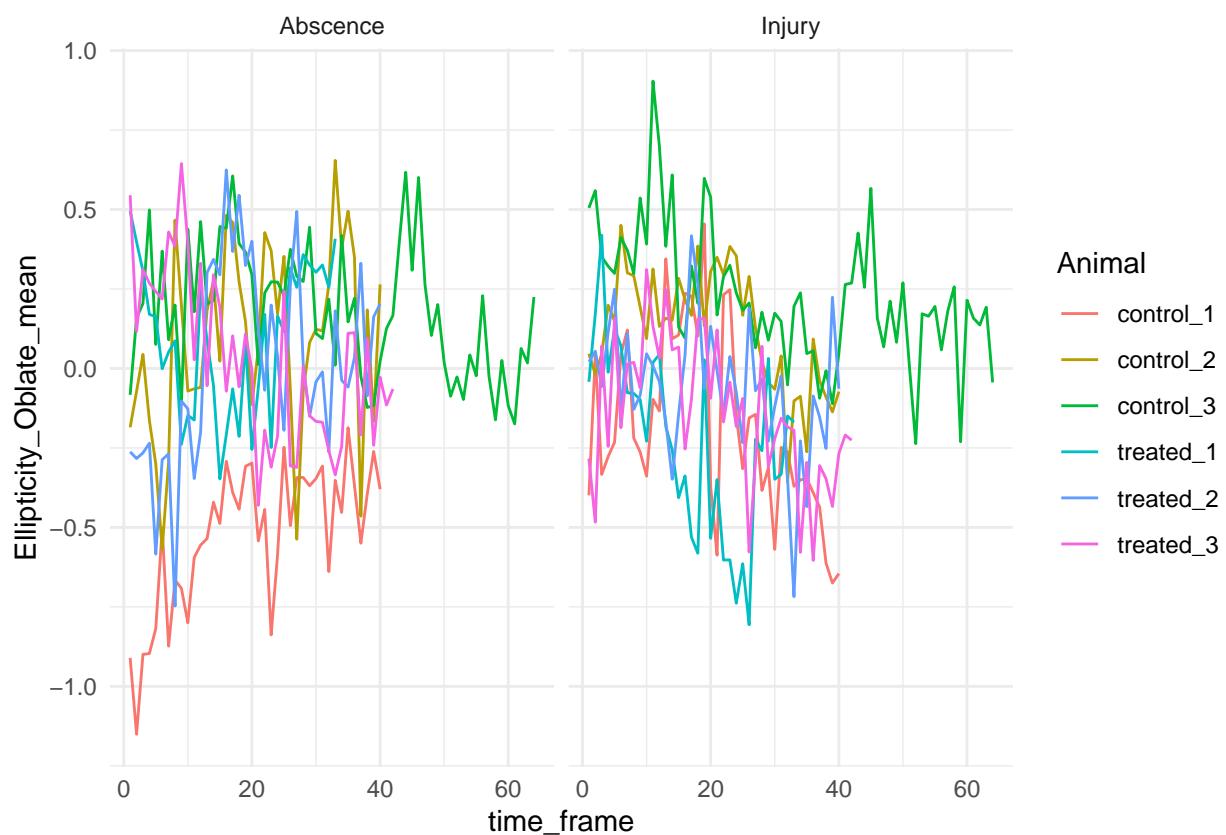
Area_um2_mean_time_plot



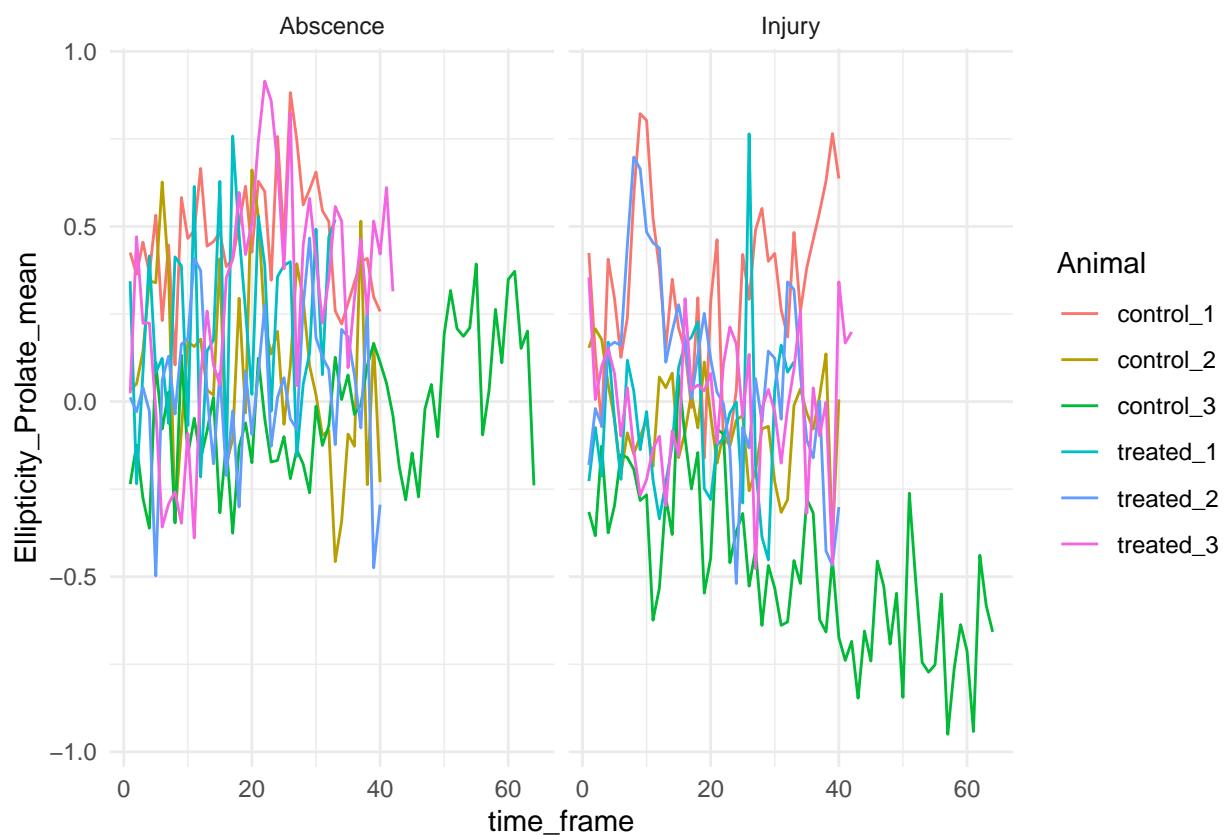
Displacement_delta_z_um_mean_time_plot



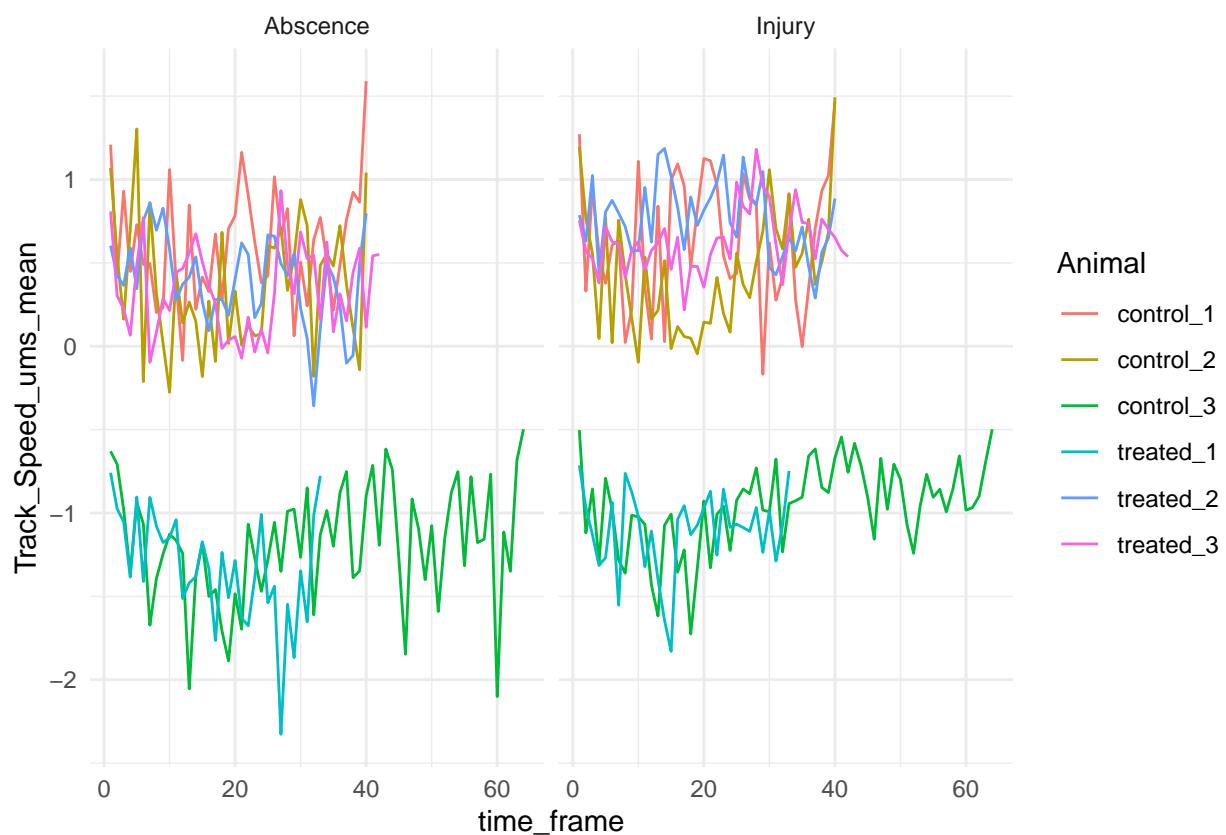
Ellipticity_Oblate_mean_time_plot



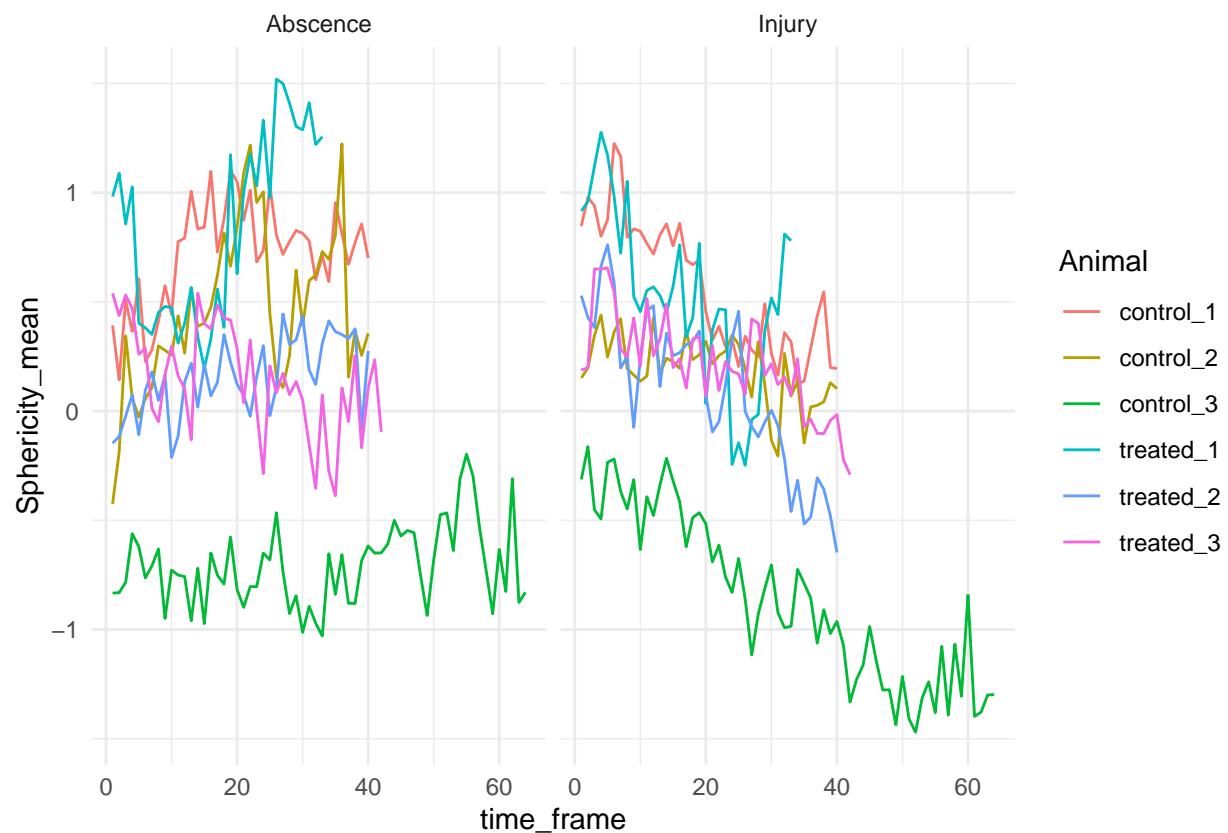
Ellipticity_Prolate_mean_time_plot



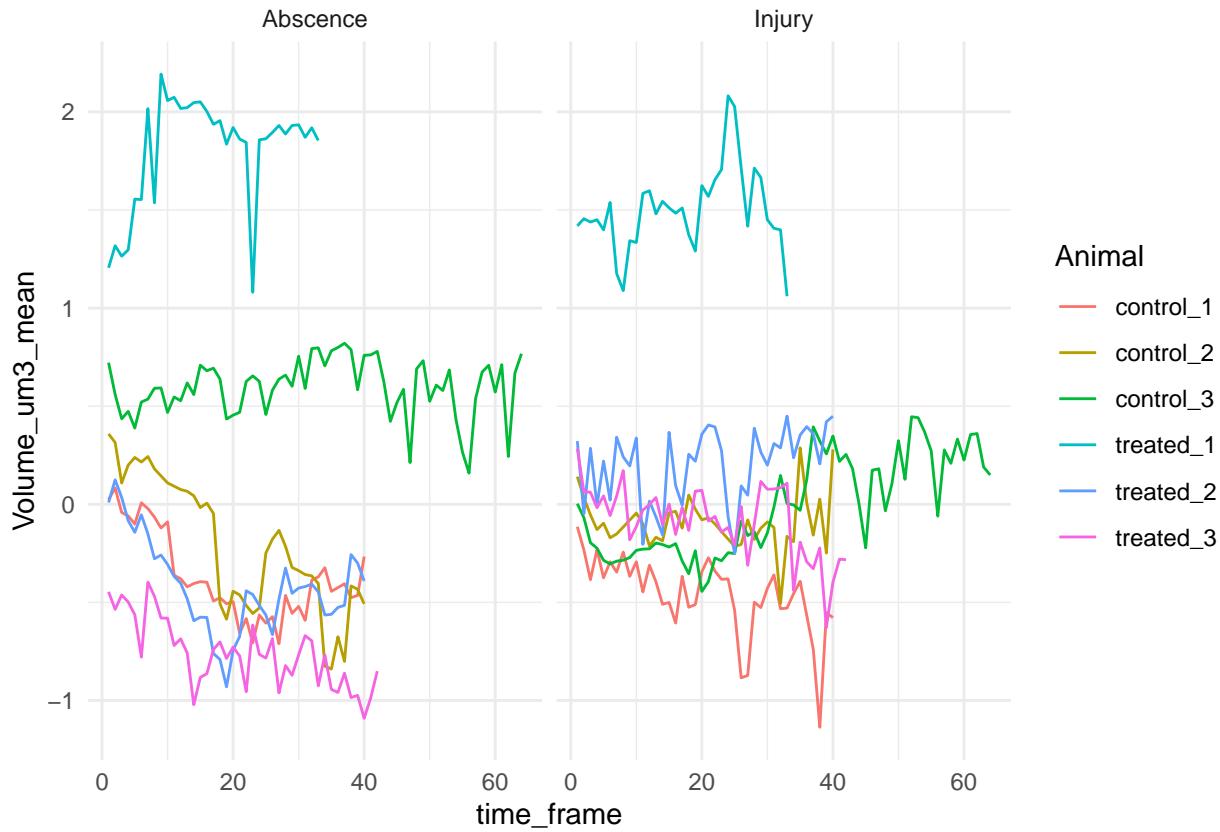
Track_Speed_um_s_mean_time_plot



Sphericity_mean_time_plot



Volume_um3_mean_time_plot



Code 2:

Running an lapply loop to create a wrap plot. This wrap plot contains 11 different graphs corresponding to the 11 different graph. Each graph is faceted into two: one for presence injury (injury) and one for absence of injury (abscence). These 11 graphs are line graphs which detail the change in measurement values (y axis) over the time frame (x axis). There are 6 lines on each graph, each corresponding to one of the 6 animal measurements (3 control and 3 treated). This big plot provides a better comparison for visual context.

```
#library
library(patchwork)

#Saving measurement column names as a string
colNames <- names(cell.measurements)[5:15]

#Visualising all the data in black and white
#Code to run this graphing across all of the columns
Value_Time_Plots <- do.call(wrap_plots, lapply(colNames, function(x){
  # plotting code
  cell.measurements %>%
    mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence')) %>%
    group_by(Animal,time_frame) %>%
    ggplot(aes_string(x="time_frame", y = x, color = 'Animal')) +
    geom_line() +
    theme(axis.text = element_text(size = 0.1)) +
    facet_wrap(facets = vars(Injury)) +
    scale_color_manual(values = c(control_1 = "#E69138", control_2 = "#C8A234", control_3 = "#99CC66", treated_1 = "#00BFFF", treated_2 = "#4682B4", treated_3 = "#FF00FF")) +
    theme(panel.grid = element_line())
}))
```

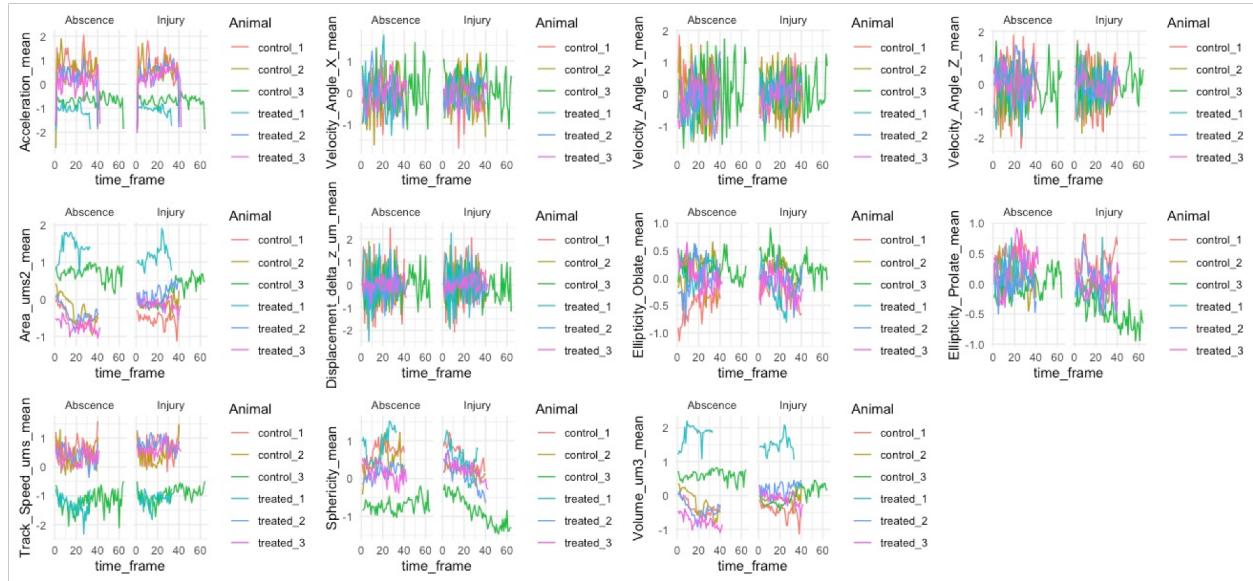
```

    theme_minimal()
})))

```

Plot 2:

Large plot detailing all measurements faceted into absence and presence of injury



Discussion:

Can you see similar behaviours across animals? The behaviours across animals largely depend on the measurement. For

Does time have a big effect on all or some of the measurements?

Time does not have a big effect on all of the measurements. It has a big effect on acceleration_mean as the values tend downwards over time. It does not have much effect on velocity_angle_x_mean which have no correlation with time.

You can see similar behaviours... Time has a great affect on some and all of the measurements..

Exercise V3

Question:

What are the most correlated measurements in the dataset? Create a correlation plot and discuss what are the most positive correlated and negative correlated variables.

Code 1 and Plot 1:

To create a corrplot detailing all 11 measurements on the x and y axis and there correlation within a grid where measurement axis meet. There is a colour code on the right of the plot, detailing the extent of positive

or negative correlation. A maximum value of correlation is 1 and a minimum value of correlation is -1. Where the same measurements meet in the grid, correlation is of course a maximum value of 1.

```
#Install necessary libraries for statistical analysis
library(corrplot)

## corrplot 0.92 loaded

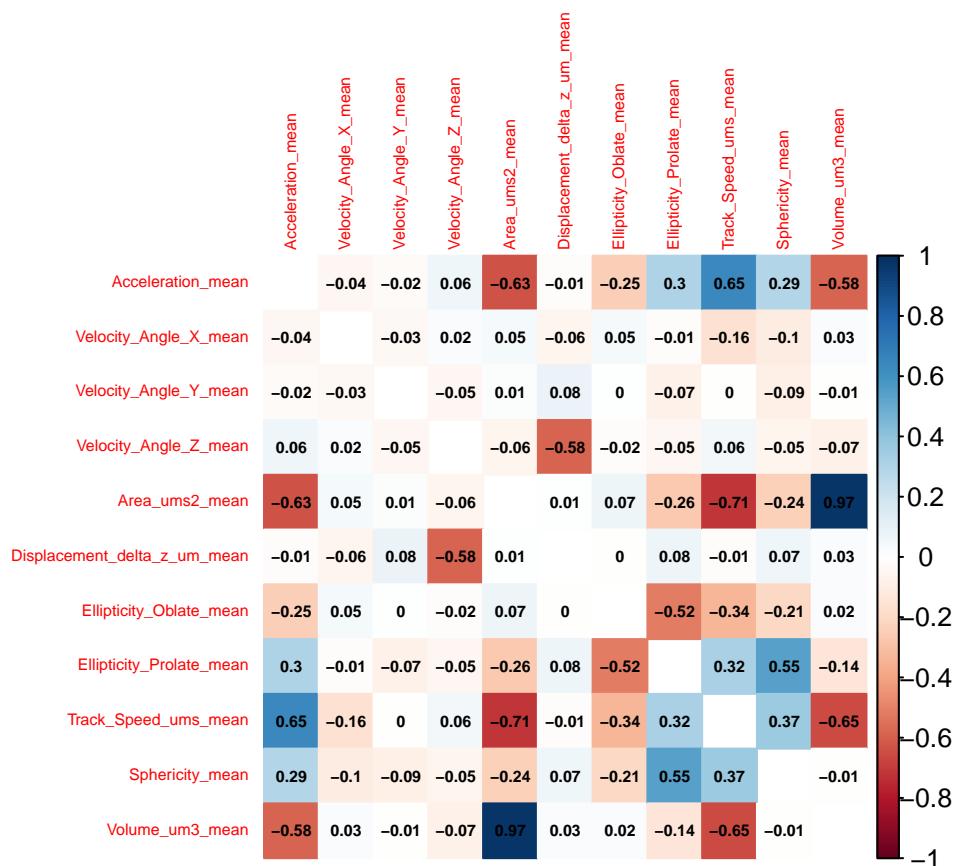
#Load the dataframe
cell.measurements <- read.csv('/Users/judepops/Documents/Bioinformatics_Module/CW_2/CW_2_Data/RNAseq/cell.

#extracting the 11 columns of quantitative data from the cell.measurements dataframe
cell.measurements.quantitative <- cell.measurements[,5:15]

#Using a spearman/pearson rank correlation: https://www.rdocumentation.org/packages/stats/versions/3.6.2

correlation_cell.measurements <- cor(cell.measurements.quantitative, method="pearson", use="pairwise.correlation")

# graph correlation specific columns
cell.measurements.corrplot <- corrplot(correlation_cell.measurements,
                                         method="color", addCoef.col = "black", tl.cex = 0.5, number.cex=0.5, diag = FALSE)
```



Code 2 and Plot 2

Ranking the top differentially expressed genes using a GITHUB package: *lares*

```

#Loading the package
#devtools::install_github("laresbernardo/lares")
library(lares)

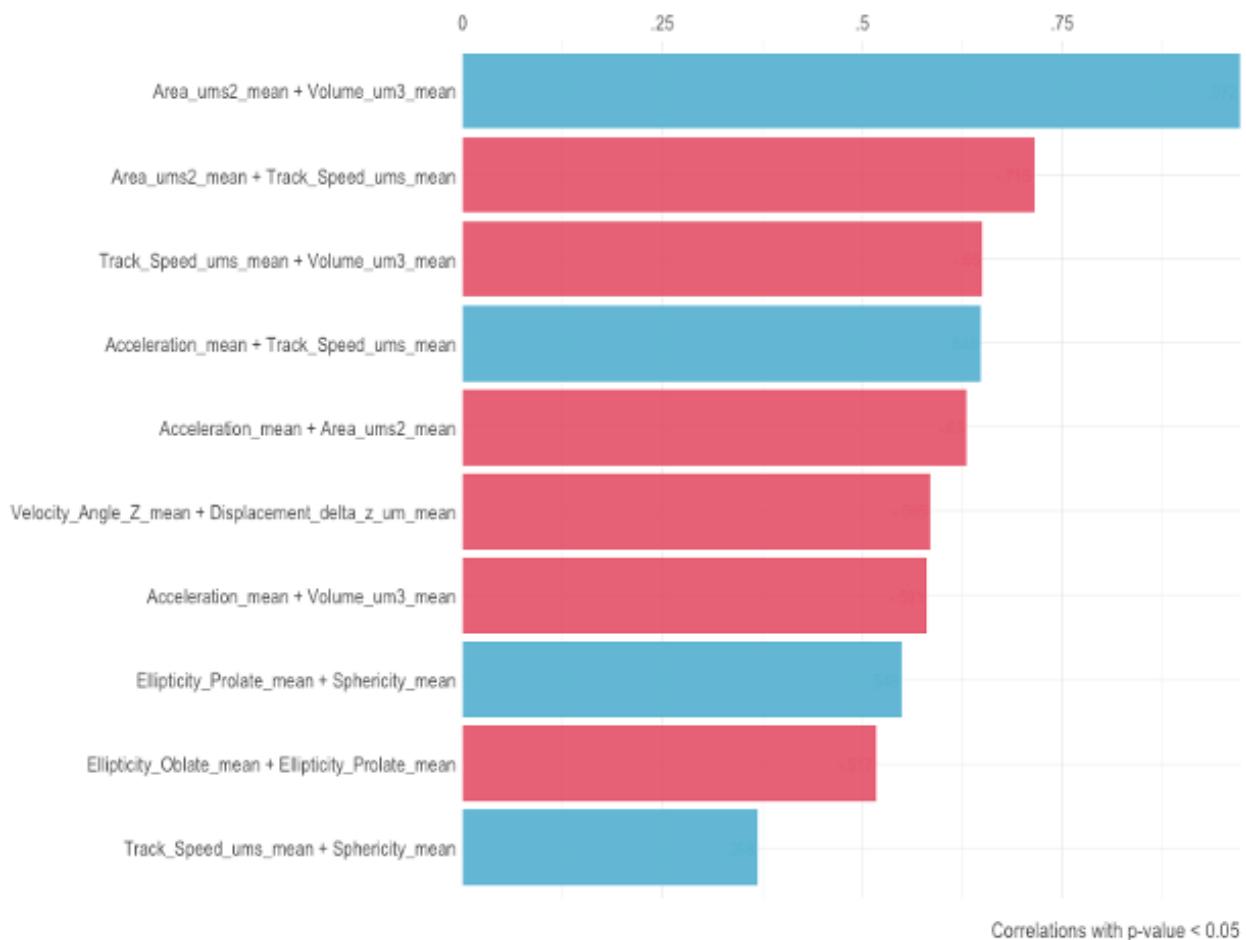
#Ranking the top 10 genes
top_10_corr <- corr_cross(cell.measurements.quantitative, # dataset
                           max_pvalue = 0.05, # show only sig. correlations at selected level
                           top = 10) # display top 10 correlations, any couples of variables

```

Returning only the top 10. You may override with the 'top' argument

Ranked Cross-Correlations

10 most relevant



Discussion:

What are the most correlated measurements in the dataset? What are the most positive correlated and negatively correlated variables?

The correlation between same measurements is a good control to check that the correlation worked as same measurements will always have a maximum correlation of 1.

We can gather that the most correlated cell measurements in the dataset are the ones with the highest values: the highest negative values and the highest positive values. The top 10 most correlated measurements are

shown in the ‘Ranked Cross Correlations’ graph. This displays the positive correlated measurements in blue and the most negative correlated measurements in red, just as in the corrplot. All correlations have a p-value less than 0.05.

The most positively correlated variables are between Area_ums2_mean and Volume_um3_mean, Acceleration_mean and Track_Speed_ums_mean, Ellipticity_Prolate_mean and Sphericity_mean and finally Track_Speed_ums_mean and Sphericity_mean in descending positive correlation. These all have high values of positive correlation ranging from 0.97 down to 0.37.

The most negatively correlated variables are between Area_ums2_mean and Track_Speed_ums_mean, Track_Speed_ums_mean and Volume_um3_mean, Acceleration_mean and Area_ums2_mean, and finally Velocity_Angle_Z_mean and Displacement_delta_z_um_mean in descending negative correlation. These have high values of negative correlation ranging from -0.71 to -0.58.

Overall there are more negatively correlated variables in the top 10 most correlated variables than positive correlated variables. This suggests that the values are mainly negatively correlated. Interestingly, the range in the top 4 most positive correlated means (from 0.97 to 0.37) is far greater than the range in the 4 most negatively correlated means (-0.71 to -0.58). This further supports the assumption that the measurements are more negatively correlated than positively correlated on average.

Section 2: Stats with R

Exercise SR1

Question:

Consider the Velocity_Angle_X_mean measurement. Is there a statistical significant difference between Injury and not Injury in control animals? Is there a statistical significant difference between Injury and not Injury in treated animals? Explain which test you used and why it is the appropriate one. Discuss and visualise the results

- **Normality:** The data have to be normally distributed (normal shape, bell shape, Gaussian shape).
- **Homogeneity in variance:** The variance should not change systematically throughout the data.
- **Interval data:** The distance between points of the scale should be equal at all parts along the scale
- **Independence:** Data from different subjects are independent so that values corresponding to one subject do not influence the values corresponding to another subject. Basically, it means one measure per subject. For more details, check the presentation of the course.

Code and 3 graphs (Visualising the data distribution):

To subset the dataframe into control and treated, selecting specific columns and mutating columns.

```
#Subsetting the dataframe into Velocity_Angle_X_mean and injury
cell.measurements.t_test <- cell.measurements %>%
  select(Animal, Injury, Velocity_Angle_X_mean) %>%
  mutate(Injury = ifelse(Injury == "1", "Prescence of Injury", 'Abscence of Injury'))
```

```
#Subsetting into control and treated
```

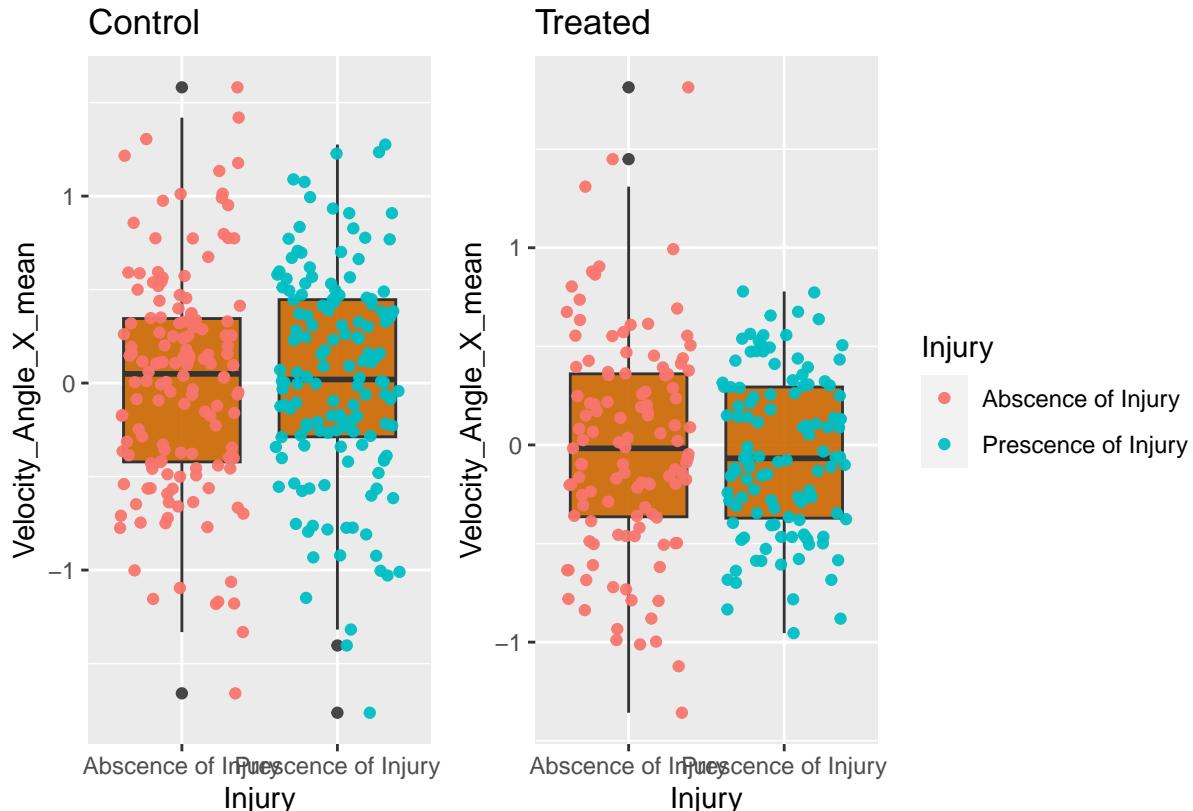
```
cell.measurements.t_test.control <- cell.measurements.t_test[cell.measurements.t_test$Animal %in% c('control', 'treated')]  
cell.measurements.t_test.treated <- cell.measurements.t_test[cell.measurements.t_test$Animal %in% c('treated')]
```

To create graphs that visualise the spread of Velocity_Angle_X means between control and treated animals.

```
#Using a boxplot as they are the most informative for the first 2 assumptions and they also show you outliers
```

```
##Boxplots
```

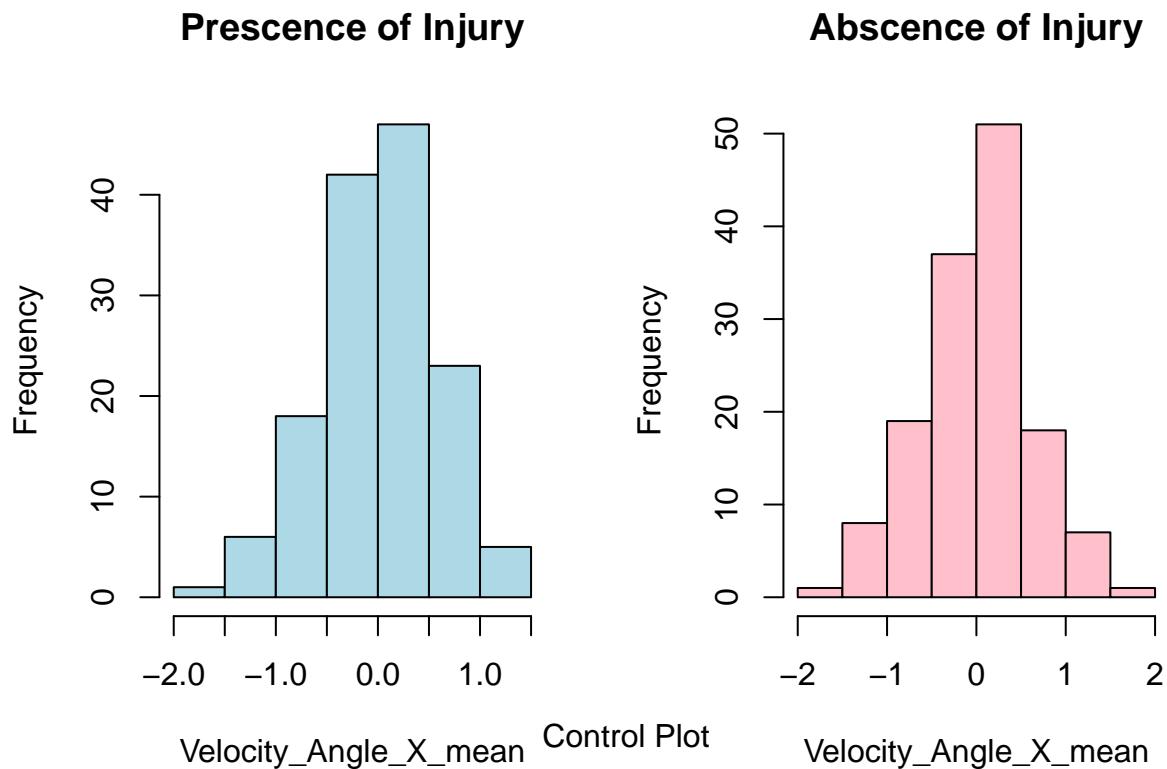
```
control_boxplot <- ggplot(cell.measurements.t_test.control, aes(Injury, Velocity_Angle_X_mean)) + geom_boxplot()  
treated_boxplot <- ggplot(cell.measurements.t_test.treated, aes(Injury, Velocity_Angle_X_mean)) + geom_boxplot()  
  
control_boxplot + treated_boxplot
```



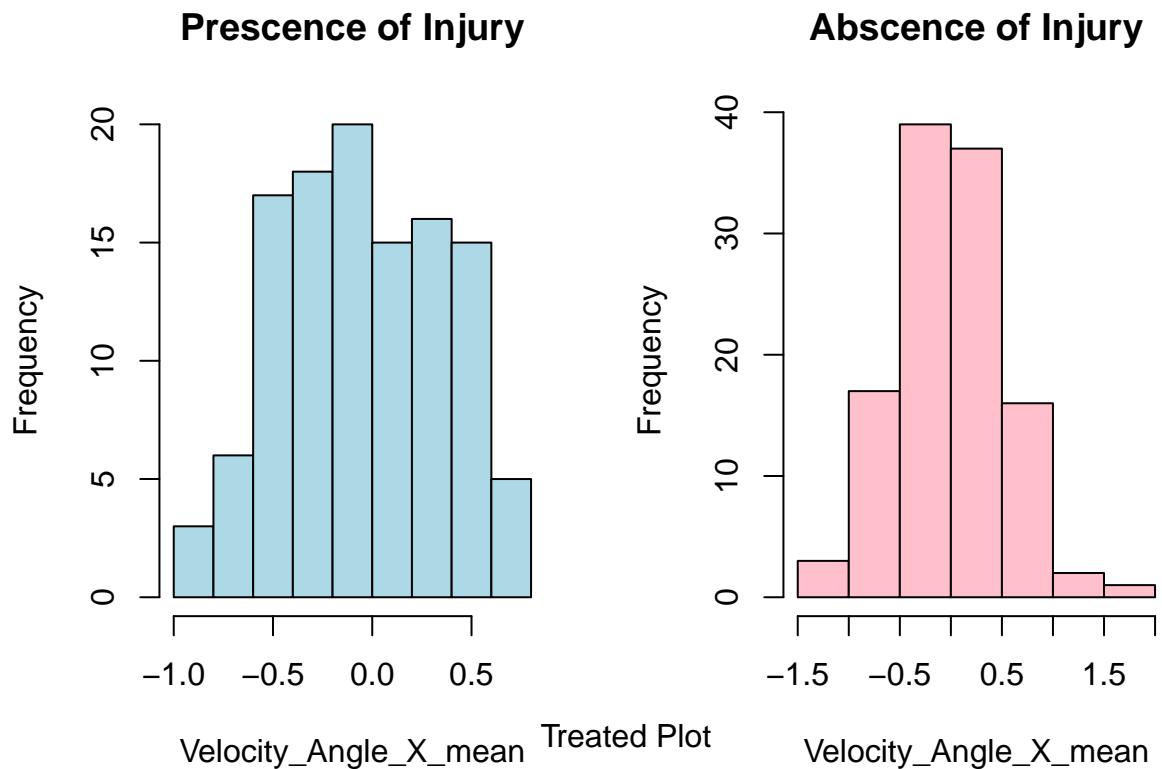
```
#Using a histogram to visualize distribution
```

```
##Control
```

```
par(mfrow=c(1,2))  
hist(cell.measurements.t_test.control[cell.measurements.t_test.control$Injury=="Prescence of Injury",]$VelocityAngleX, main="Control Plot", xlab="Velocity Angle X",  
hist(cell.measurements.t_test.control[cell.measurements.t_test.control$Injury=="Abscence of Injury",]$VelocityAngleX, main="Control Plot", xlab="Velocity Angle X",  
mtext("Control Plot", side = 3, line = -21, outer = TRUE)
```



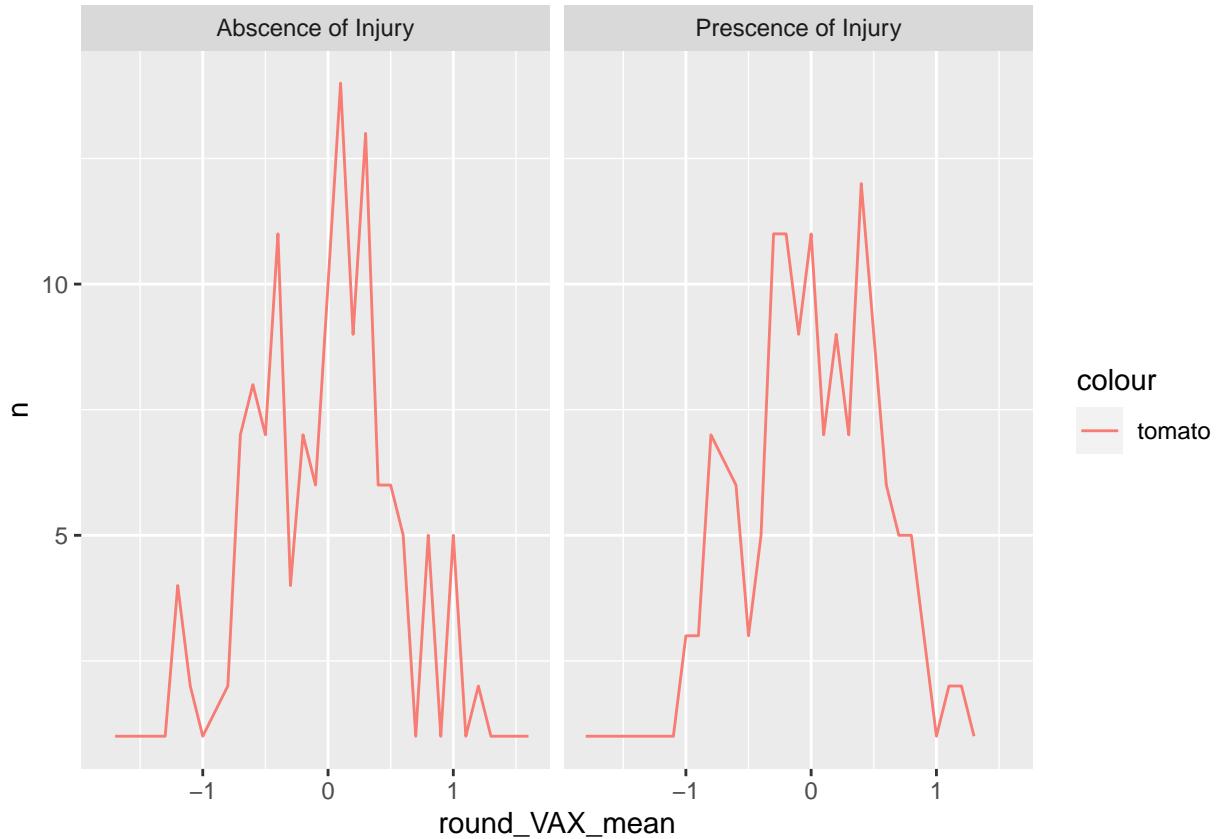
```
##Treated
par(mfrow=c(1,2))
hist(cell.measurements.t_test.treated[cell.measurements.t_test.treated$Injury=="Prescence of Injury",]$Velocity_Angle_X_mean)
hist(cell.measurements.t_test.treated[cell.measurements.t_test.treated$Injury=="Abscence of Injury",]$Velocity_Angle_X_mean)
mtext("Treated Plot", side = 3, line = -21, outer = TRUE)
```



```
#Visualising the spread of Velocity_Angle_X_mean means between control and injured
cell.measurements.t_test.control$round_VAX_mean <- round(cell.measurements.t_test.control$Velocity_Angle_X_mean)

t_test.control_dist <- cell.measurements.t_test.control %>%
  group_by(Injury, round_VAX_mean) %>%
  count(Injury, round_VAX_mean)

t_test.control_dist %>%
  ggplot(mapping = aes(x = round_VAX_mean, y=n, color = 'tomato')) +
  geom_line() +
  facet_wrap(facets = vars(Injury))
```



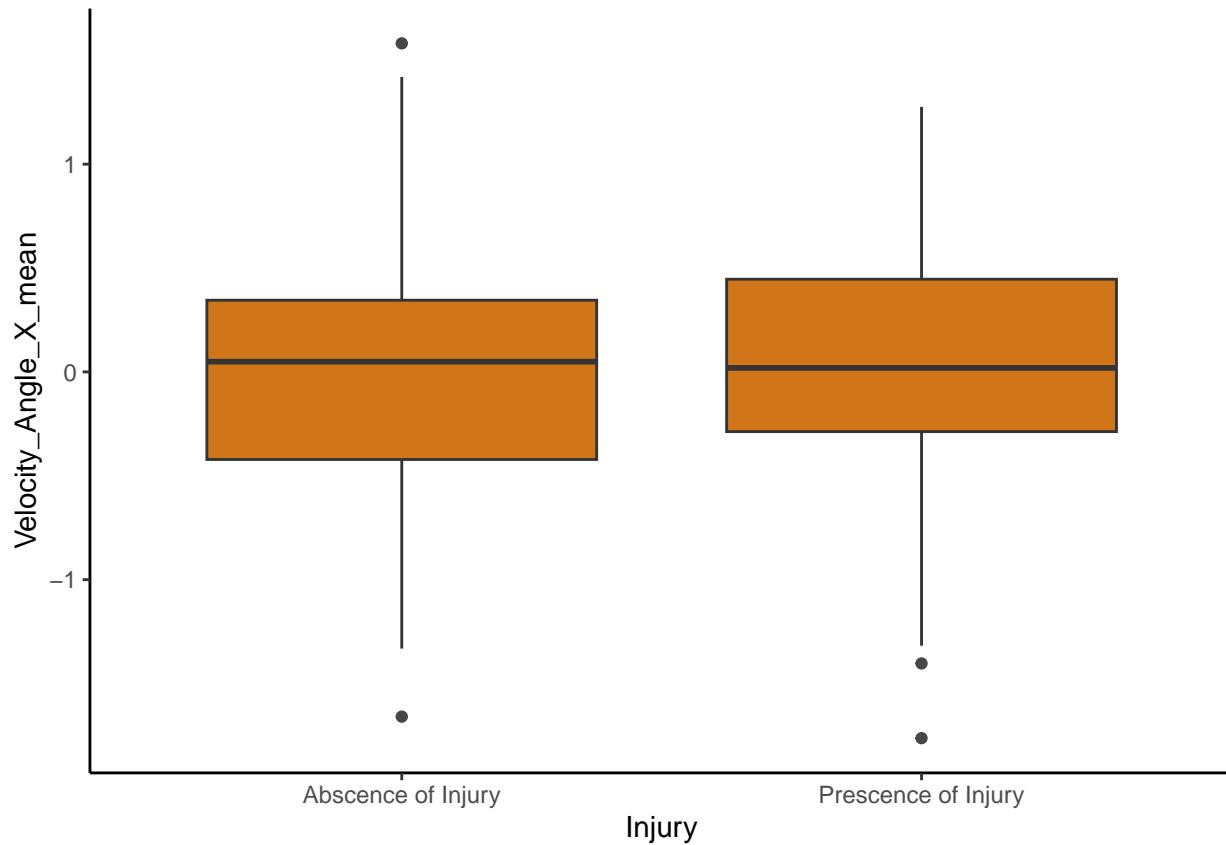
Code and results (T-Tests):

Control T test:

```
#Removing the Animal column
cell.measurements.t_test.control <- cell.measurements.t_test.control %>%
  select(Injury, Velocity_Angle_X_mean)

##T test for control measurements

#1) Plotting boxplot
ggplot(cell.measurements.t_test.control, aes(x = Injury, y = Velocity_Angle_X_mean)) + geom_boxplot() +
```



#2) Calculating test statistic

```
t.test(Velocity_Angle_X_mean ~ Injury, data = cell.measurements.t_test.control, var.equal = TRUE, paired = FALSE)
```

```
##
```

```
## Two Sample t-test
```

```
##
```

```
## data: Velocity_Angle_X_mean by Injury
```

```
## t = -0.34856, df = 282, p-value = 0.7277
```

```
## alternative hypothesis: true difference in means between group Abscence of Injury and group Prescence of Injury is not equal to 0
```

```
## 95 percent confidence interval:
```

```
## -0.1611001 0.1126287
```

```
## sample estimates:
```

```
## mean in group Abscence of Injury mean in group Prescence of Injury
```

```
## 0.004853499 0.029089197
```

#paired is false because we are running a two sampled t test

#we assume variation is equal so var.equal is true

#TUS, T value = -0.34856

#3) Stating the p value

#P value = 0.7277

#4) Conclusions (in discussion below)

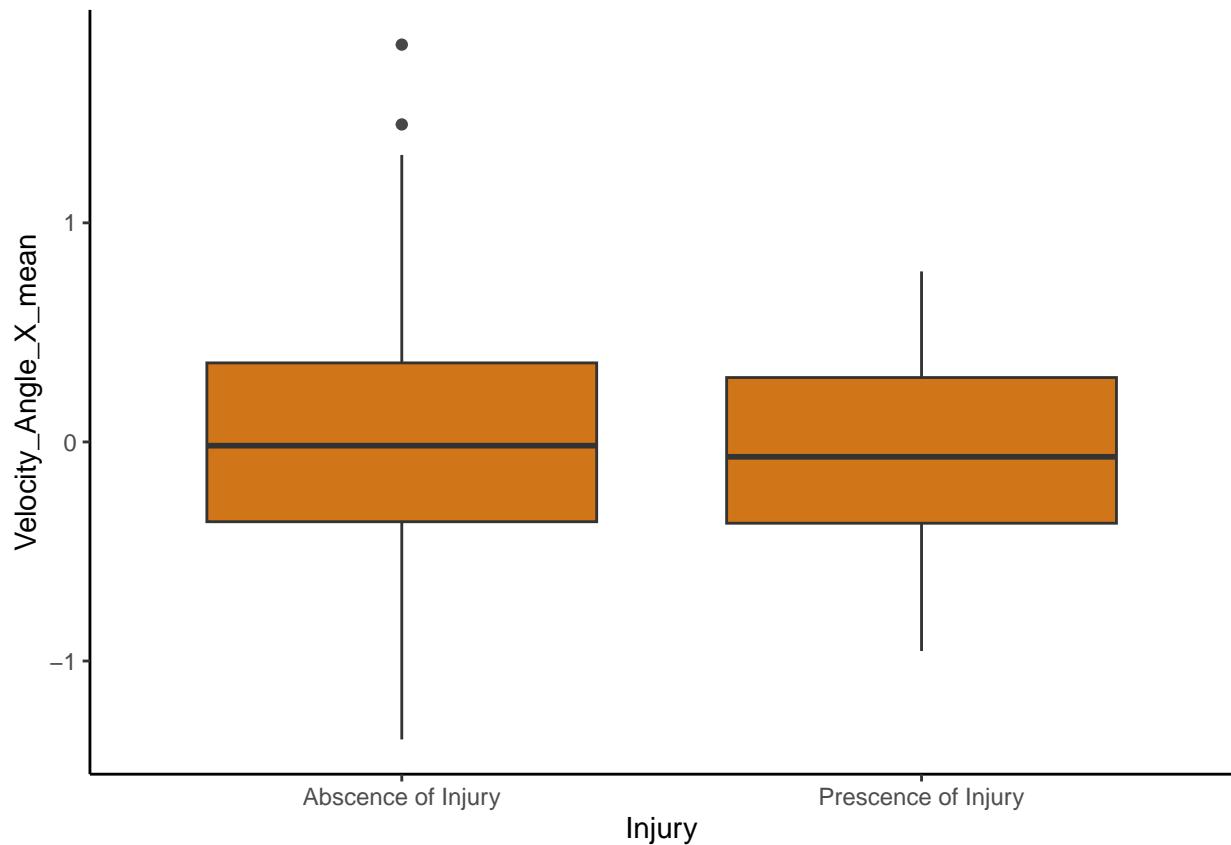
Treated T test

```
#Removing the Animal column

cell.measurements.t_test.treated <- cell.measurements.t_test.treated %>%
  select(-Injury, -Velocity_Angle_X_mean)

##T test for treated measurements

#1) Plotting boxplot
ggplot(cell.measurements.t_test.treated, aes(x = Injury, y = Velocity_Angle_X_mean)) + geom_boxplot() +
```



```
#2) Calculating test statistic
t.test(Velocity_Angle_X_mean ~ Injury, data = cell.measurements.t_test.treated, var.equal = TRUE, paired = FALSE)

##
##  Two Sample t-test
##
## data: Velocity_Angle_X_mean by Injury
## t = 0.53046, df = 228, p-value = 0.5963
## alternative hypothesis: true difference in means between group Abscence of Injury and group Prescence of Injury is not equal to 0
## 95 percent confidence interval:
## -0.09235362  0.16039631
## sample estimates:
## mean in group Abscence of Injury mean in group Prescence of Injury
## -0.01016812                      -0.04418947
```

```

#paired is false because we are running a two sampled t test
#we assume variation is equal so var.equal is true
#THUS, T value = -0.53046

#3) Stating the p value
#P value = 0.5963

#4) Conclusions (in discussion below)

```

Discussion:

Deciphering whether there is a statistically significant difference between Injury and not Injury in control animals and treated animals. Why I chose to use a t-test.

Our data meets the four key criteria that are required to perform a *t*-test: the measurement is continuous (Velocity_Angle_X_mean), the sample data have been randomly sampled from a population, there is homogeneity of variance (the variability in each group is similar - this can be seen in the **boxplot** generated above where both injury and control have values with similar ranges within -2 and 2), and finally the distribution is approximately normal (this can be seen in the **histogram** and **line curve of means** generated above which generates a data that qualitatively follows the pattern of a normal distribution).

I chose to use an Two-Sample *t*-test. This is because we have a continuous measurement (Velocity_Angle_X_mean) which is split into two categorical groups: injury and control. The purpose of this test is to decide if the population means of the two different groups (injury and control) are equal or not at a 5% significance level.

Discussing Results

Null hypothesis: The means of Velocity_Angle_X_mean are the same between injury and control groups

Alternative hypothesis: The means of Velocity_Angle_X_mean are unequal between injury and control groups

Two sided test: So we can see if difference is above or below. No directionality is implied.

Two sample: The animals included in the presence of injury trials are independent of those in the absence of injury trials

For the control animals:

There is no statistical significant difference between injury and not injury. This is because the p value (0.7277) is well above 0.05 threshold and we thus have to accept the null hypothesis and reject the alternative hypothesis.

For the treated animals:

There is also no statistical significant difference between injury and non injury. This is because the p value (0.5963) is also well above 0.05 threshold and thus we have to accept the null hypothesis and reject the alternative hypothesis.

Exercise SR2

Question:

Is there a statistically significant relationship between each cell measurement and the following predictor variables: Injury, Treatment, time? Explain which test you used and why it is the appropriate one. Discuss and present the results.

Here we have to use analysis of variance because we are comparing more than 2 groups (means): one-way ANOVA which is an extension of 2 groups comparison of a t-test but with slightly different logic. We are comparing 11 means (each measurement) with 3 predictor variables: Injury, Treatment and time.

Code:

Subsetting the cell measurements dataframe into each predictor variable so there is only one independent variable among the 11 cell measurements.

```
#Predictor variable is the name given to an independent variable used in regression analysis. It provides information about the independent variable in the model.

#Before calculating the correlation coefficient for a pair of variables, must check whether association exists between them.
##This can be done for the whole dataframe at once in a tabular format known as a scatter plot matrix.

#A) Subsetting the data into the predictor variables
cell.measurements.injury = subset(cell.measurements, select = -c(Animal,Treatment,time_frame)) #subsetting
cell.measurements.treatment = subset(cell.measurements, select = -c(Animal,Injury,time_frame))
cell.measurements.time_frame = subset(cell.measurements, select = -c(Animal,Treatment,Injury))

cell.measurements.time_frame %>%
  group_by(time_frame) %>%
  view()
```

Starting with the Pearson correlation code and results between the continuous predictor variable (time_frame) and continuous dependent variables (cell measurements)

Visualising the correlation

```
#B) Time_Frame Scatter

# Time frame is a continuous variable. The cell measurements are also continuous variables. This means
#we can perform a simple correlation test between one continuous independent variable and multiple
#continuous dependent variables.
# The pearson correlation test is suitable to compute correlation between a continuous dependent variable
#and continuous independent (predictor) variables (measurements)

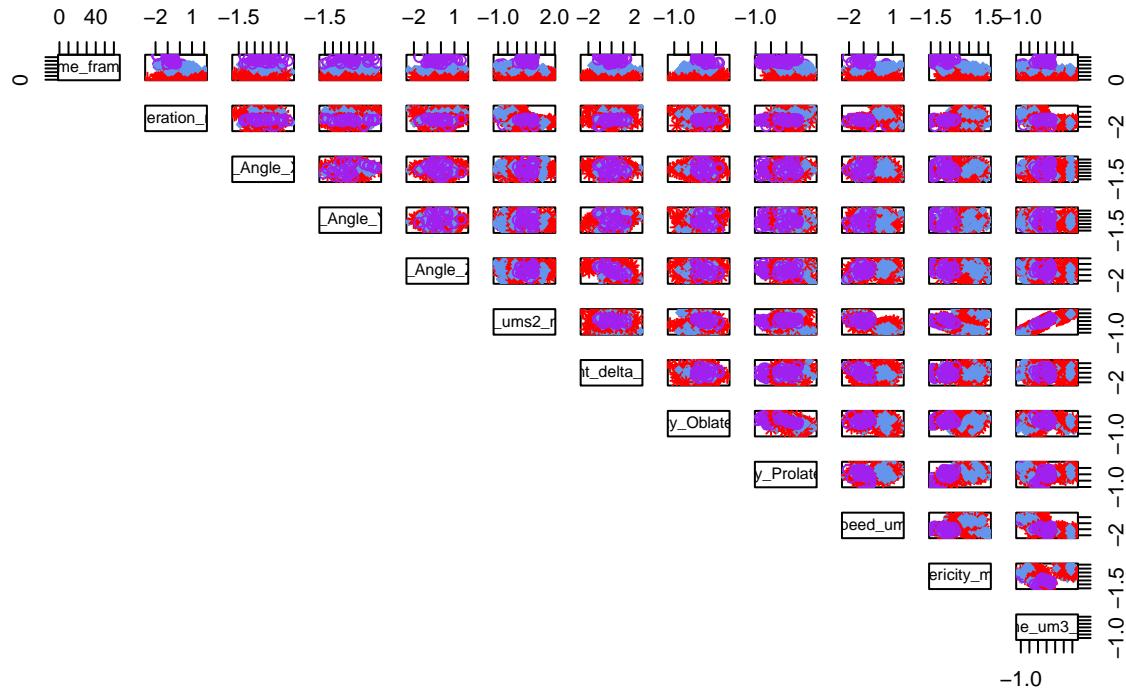
## We can also plot the correlation for a visual representation

# Adding a group indicator (three groups 1, 2 & 3) to our example data to simulate colours for different
# Time frame data ranges from 0 to 64 so we will split into 3 groups

group <- NA #here we start with the group as NA so we can begin to edit it
group[cell.measurements.time_frame$time_frame < 20] <- 1 #We assign the first group to all time frames less than 20
group[cell.measurements.time_frame$time_frame >= 20 & cell.measurements.time_frame$time_frame <= 40] <- 2 #We assign the second group to all time frames between 20 and 40
group[cell.measurements.time_frame$time_frame > 40] <- 3 #Finally, we assign anything greater than 30 as the third group

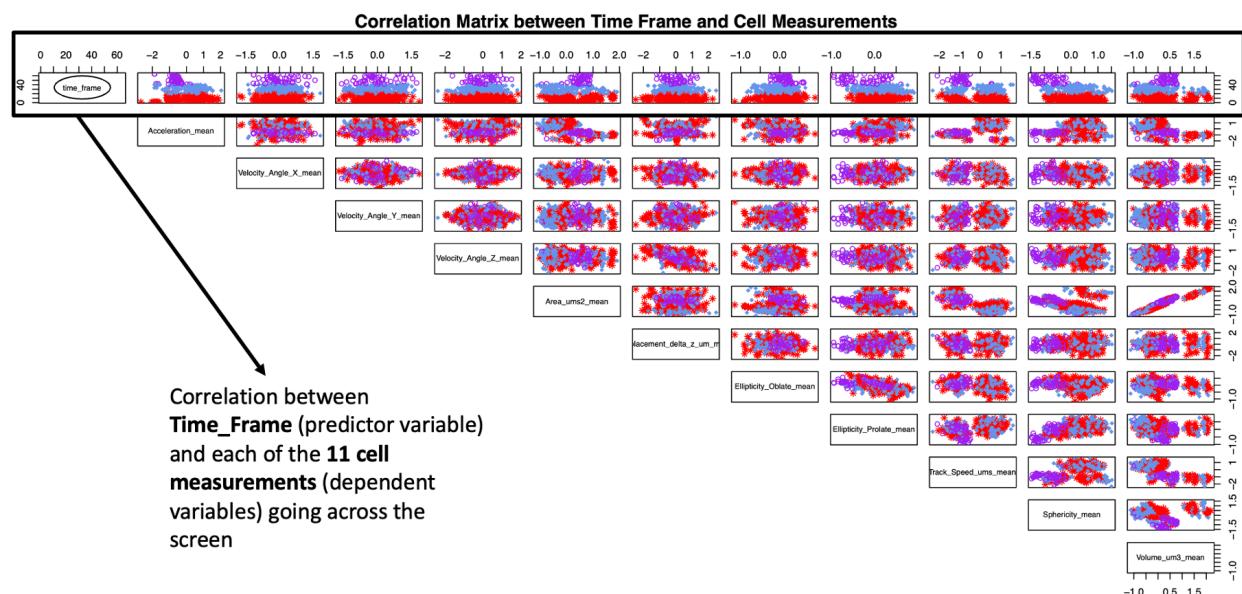
#Plotting a correlation matrix using the pairs function built into r
Cor_time_frame_cell <- pairs(cell.measurements.time_frame[, 1:12], #selecting columns to test as 1:12
  lower.panel = NULL, #dropping the lower panel as this is not necessary
  col = c("red", "cornflowerblue", "purple")[group], # Changing color by group. Concatenate lists
  pch = c(8, 18, 1)[group], # Changing points by group, once again using
  main = "Correlation Matrix between Time Frame and Cell Measurements") #adding a title to the graph
```

Correlation Matrix between Time Frame and Cell Measurements



```
# The top row is the column that displays the correlation between the time frame continuous predictor v
#dependent continuous variables of cell measurement
```

Graph:



The time frame Pearson statistical significance correlation and p values with the 11 cell mea-

surements:

```
# C) Time frame statistical test: correlation and p values

#Redefine the cell measurements time frame dataframe
cell.measurements.time_frame = subset(cell.measurements, select = -c(Animal,Treatment,Injury))

#Calculate the correlation and display p values
library(Hmisc) #loading the hmisc library

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following object is masked from 'package:lares':
## 
##     impute

## The following objects are masked from 'package:dplyr':
## 
##     src, summarize

## The following objects are masked from 'package:base':
## 
##     format.pval, units

# A simple function to define the format of the correlation matrix where:
# cormat : matrix of the correlation coefficients
# pmat : matrix of the correlation p-values

flattenCorrMatrix <- function(cormat, pmat) {
  ut <- upper.tri(cormat)
  data.frame(
    row = rownames(cormat)[row(cormat)[ut]],
    column = rownames(cormat)[col(cormat)[ut]],
    cor = (cormat)[ut],
    p = pmat[ut]
  )
}

# Creating the correlation matrix for my dataframe

res2<-rcorr(as.matrix(cell.measurements.time_frame[,1:12]))
time_frame_correlation <- flattenCorrMatrix(res2$r, res2$P)

# Removing all rows apart from time frame
time_frame_correlation <- time_frame_correlation[time_frame_correlation$row=="time_frame", ]
```

Results:

```
time_frame_correlation
```

```
##          row            column      cor       p
## 1 time_frame Acceleration_mean -0.11306647 1.030615e-02
## 2 time_frame Velocity_Angle_X_mean  0.01538210 7.279094e-01
## 4 time_frame Velocity_Angle_Y_mean  0.09866601 2.529107e-02
## 7 time_frame Velocity_Angle_Z_mean -0.02665629 5.465248e-01
## 11 time_frame Area_um2_mean    0.06358436 1.500076e-01
## 16 time_frame Displacement_delta_z_um_mean -0.01775224 6.880353e-01
## 22 time_frame Ellipticity_Oblate_mean -0.01248915 7.775829e-01
## 29 time_frame Ellipticity_Prolate_mean -0.20527235 2.697838e-06
## 37 time_frame Track_Speed_um2_mean -0.15011192 6.393382e-04
## 46 time_frame Sphericity_mean   -0.44869289 0.000000e+00
## 56 time_frame Volume_um3_mean   -0.03832002 3.859539e-01
```

Pearson correlation code and results between the categorical predictor variable (Injury) and continuous dependent variables (cell measurements)

Visualising the Correlation

Code and Graphs:

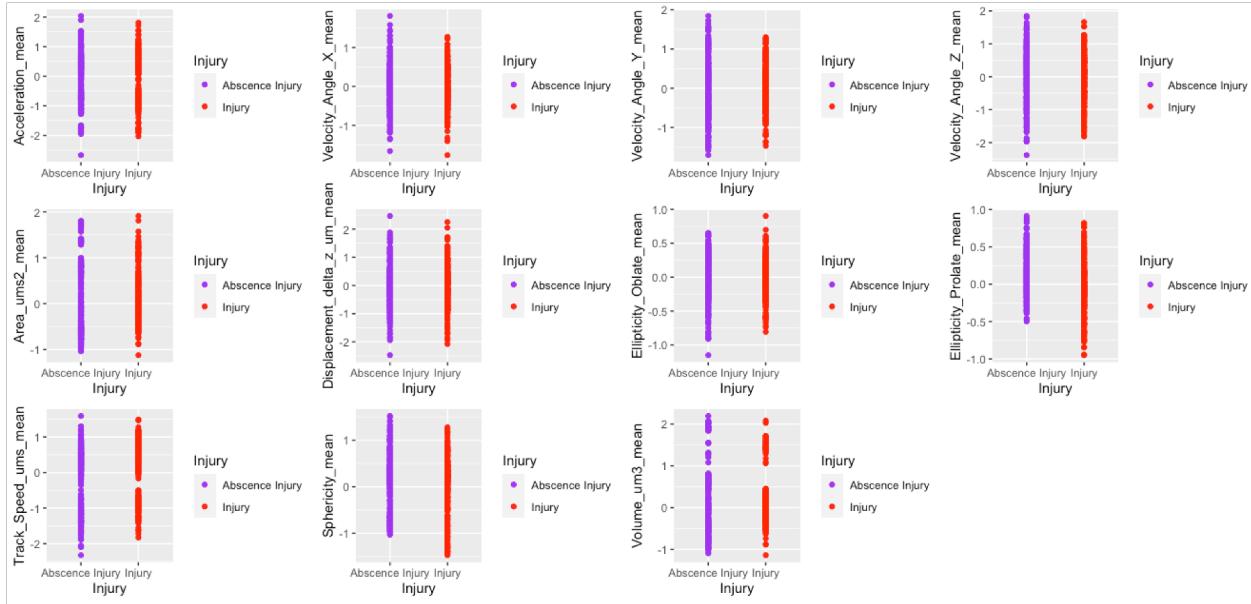
```
#Refreshing dataframe
cell.measurements.injury = subset(cell.measurements, select = -c(Animal,Treatment,time_frame))

#Libraries
library(ggplot2)
library(patchwork)

#Saving the column names as a variable to put into the lapply function
colNames <- names(cell.measurements.injury)[2:12]

# Creating 11 distinct box plots to visualise the categorical correlation between injury
#and the 11 cell measurements

Cor_tests_injury <- do.call(wrap_plots, lapply(colNames, function(x){ #opening the lapply function
  # plotting code
  cell.measurements.injury %>%
    mutate(Injury = ifelse(Injury == "1", "Injury", 'Abscence Injury')) %>% #Creating binary values for c
  #data
  ggplot(aes_string(x = 'Injury',y = x, color = 'Injury')) + geom_point() +
    scale_color_manual(values = c("Injury" = "red", "Abscence Injury" = "purple")) #adding unique colours
  #the previous plots
})) #closing the lapply function
```



The injury Pearson statistical significance correlation and p values with the 11 cell measurements:

```

# Injust statistical test: correlation and p values

#Redefine the cell measurements injury dataframe
cell.measurements.injury = subset(cell.measurements, select = -c(Animal,Treatment,time_frame))

#Calculate the correlation and display p values

#loading the hmisc library
library(Hmisc)

# A simple function to define the format of the correlation matrix where:
# cormat : matrix of the correlation coefficients
# pmat : matrix of the correlation p-values

flattenCorrMatrix <- function(cormat, pmat) {
  ut <- upper.tri(cormat)
  data.frame(
    row = rownames(cormat)[row(cormat)[ut]],
    column = rownames(cormat)[col(cormat)[ut]],
    cor = (cormat)[ut],
    p = pmat[ut]
  )
}

# Creating the correlation matrix for my dataframe

res2<-rcorr(as.matrix(cell.measurements.injury[,1:12]))
injury_correlation <- flattenCorrMatrix(res2$r, res2$P)

# Removing all rows apart from injury
injury_correlation <- injury_correlation[injury_correlation$row=="Injury", ]

```

Results:

```
injury_correlation
```

```
##      row          column      cor        p
## 1 Injury Acceleration_mean 0.037383116 3.976832e-01
## 2 Injury Velocity_Angle_X_mean -0.001690148 9.695083e-01
## 4 Injury Velocity_Angle_Y_mean  0.076648698 8.255145e-02
## 7 Injury Velocity_Angle_Z_mean -0.033021607 4.550438e-01
## 11 Injury Area_um2_mean    0.033296044 4.513030e-01
## 16 Injury Displacement_delta_z_um_mean 0.020076782 6.497499e-01
## 22 Injury Ellipticity_Oblate_mean -0.017480813 6.925604e-01
## 29 Injury Ellipticity_Prolate_mean -0.347197302 4.440892e-16
## 37 Injury Track_Speed_um3_mean  0.119912623 6.492436e-03
## 46 Injury Sphericity_mean     -0.110879585 1.188790e-02
## 56 Injury Volume_um3_mean    0.015273502 7.297551e-01
```

Pearson correlation code and results the categorical predictor variable (Treatment) and continuous dependent variables (cell measurements)

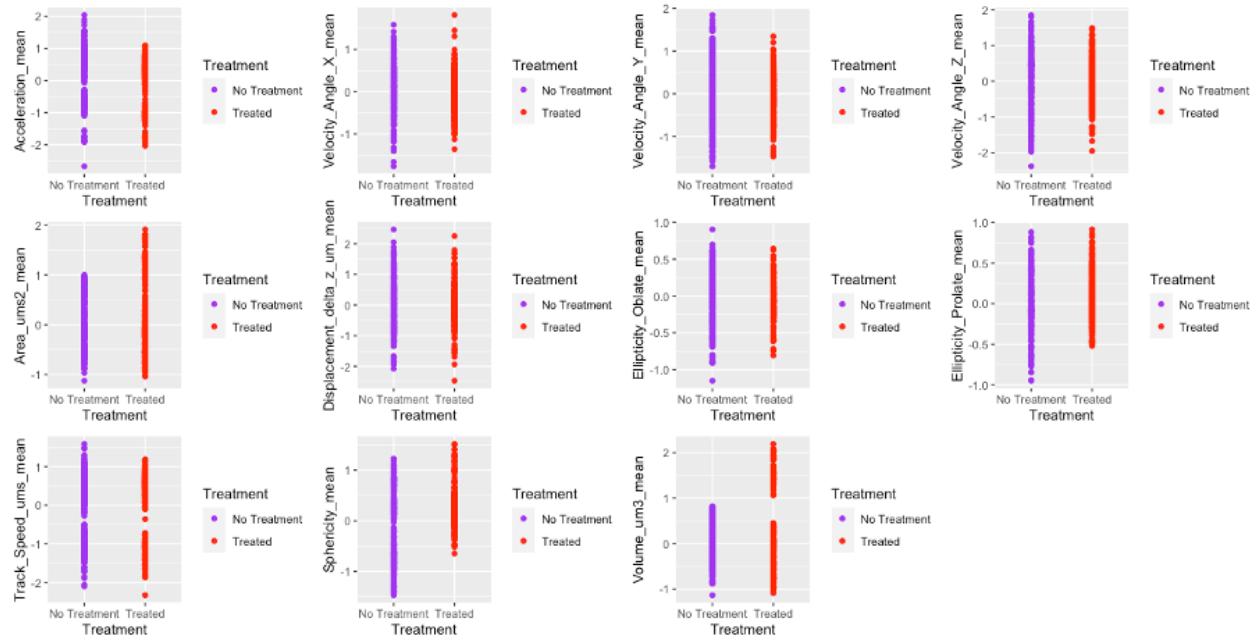
```
#Redefine the dataframe
cell.measurements.treatment = subset(cell.measurements, select = -c(Animal,Injury,time_frame))

#Libraries
library(ggplot2)
library(patchwork)

#Saving the column names as a variable to put into the lapply function
colNames <- names(cell.measurements.treatment)[2:12]

# Creating 11 distinct box plots to visualise the categorical correlation between treatment
#and the 11 cell measurements

Cor_tests_treatent <- do.call(wrap_plots, lapply(colNames, function(x){ #opening the lapply function
  # plotting code
  cell.measurements.treatment %>%
    mutate(Treatment = ifelse(Treatment == "1", "Treated", 'No Treatment')) %>% #Creating binary values for the treatment
    #data
    ggplot(aes_string(x = 'Treatment',y = x, color = 'Treatment')) + geom_point() +
    scale_color_manual(values = c("Treated" = "red", "No Treatment" = "purple")) #adding unique colours to the previous plots
})) #closing the lapply function
```



The treatment Pearson statistical significance correlation and p values with the 11 cell measurements:

```
# Treatment statistical test: correlation and p values

#Redefine the cell measurements injury dataframe
cell.measurements.treatment = subset(cell.measurements, select = -c(Animal, Injury, time_frame))

#Calculate the correlation and display p values

#loading the hmisc library
library(Hmisc)

# A simple function to define the format of the correlation matrix where:
# cormat : matrix of the correlation coefficients
# pmat : matrix of the correlation p-values

flattenCorrMatrix <- function(cormat, pmat) {
  ut <- upper.tri(cormat)
  data.frame(
    row = rownames(cormat)[row(cormat)[ut]],
    column = rownames(cormat)[col(cormat)[ut]],
    cor = (cormat)[ut],
    p = pmat[ut]
  )
}

# Creating the correlation matrix for my dataframe

res2<-rcorr(as.matrix(cell.measurements.treatment[,1:12]))
treatment_correlation <- flattenCorrMatrix(res2$r, res2$P)

# Removing all rows apart from treatment
```

```
treatment_correlation <- treatment_correlation[treatment_correlation$row=="Treatment", ]
```

Results:

```
treatment_correlation
```

```
##          row      column        cor         p
## 1 Treatment Acceleration_mean -0.084281358 5.619284e-02
## 2 Treatment Velocity_Angle_X_mean -0.040492264 3.595810e-01
## 4 Treatment Velocity_Angle_Y_mean -0.091411463 3.829018e-02
## 7 Treatment Velocity_Angle_Z_mean  0.001721543 9.689421e-01
## 11 Treatment Area_um2_mean    0.126081673 4.197264e-03
## 16 Treatment Displacement_delta_z_um_mean 0.030344106 4.924410e-01
## 22 Treatment Ellipticity_Oblate_mean -0.112533232 1.067357e-02
## 29 Treatment Ellipticity_Prolate_mean  0.142906135 1.159307e-03
## 37 Treatment Track_Speed_um3_mean  0.129716526 3.217315e-03
## 46 Treatment Sphericity_mean    0.313516031 3.468337e-13
## 56 Treatment Volume_um3_mean   0.209536692 1.649130e-06
```

Discussion

Explain which test you used and why its the appropriate one. What do results tell us?

Why I used Pearson test The default method of the core/core.test function in R is Pearson I used a Pearson Correlation test. This is because pearson tests are suitable for correlation tests between two continuous variables and also include a special ‘point-biserial correlation’ case for categorical (dichotomous variables) with continuous variables. This means I can use this test for injury, treatment and time_frame which are categorical, categorical and continuous predictor variables, respectively. The reason that Pearson can do this is bdue to the ‘point-biserial correlation’. This kind of statistical test can be used when you want to measure the relationship between a continuous variable and a dichotomous variable, or one that has two values.

Data visualisation

Before calculating p values and correlation, I visualised the data. The continous data can be visualised quite simply with a scatter plot. This can be done in a matrix using a funciton built into R: pairs. However, for the categorical predictor variables (injury and treatment), data has to be visualised differently as there are only two discrete factors. This can be done using a boxplot or point graph. I achieved this for all the 11 measurements for each predictor using ggplot within an lapply for loop.

Results

Exercise SR3

Question:

What's the difference between parametric and non-parametric data? How would you choose between performing a parametric & nonparametric tests? [max 500 words]

Answer:

The key things to consider when choosing a statistical test is ‘what is the main study hypothesis’. Some hypotheses can be confirmatory where they test a pre-supposed relationship and some can be exploratory where they are suggested by the data. Before you calculate a hypothesis test, you must check the assumptions. The fundamental difference between parametric tests and nonparametric is that parametric tests make 4 key assumptions about the parameters of the population distribution of the sample. Parametric tests include a t-test, ANOVA or Pearson correlation. Non-parametric tests usually include Mann-Whitney U tests or Spearman correlations.

The first, most important assumption, is that data must fit a certain distribution: parametric tests state that data must be normally distributed. On the other hand, non-parametric tests are ‘distribution free’: this means they can be used for non-normal variables. It could be argued that non-parametric tests can be used for all cases as a result, as they do not have any requirements; however, parametric are preferred should the data be normal for 3 key reasons. First of all, the estimates of parameters and confidence intervals in a parametric test allow us to say something about the population from which the samples came as well as obtaining a significant test. Secondly, it is hard to do flexible modelling (like allowing for confounding factors using multiple regression) with non-parametric tests. Thirdly, parametric tests have more statistical power: it is more likely to detect significant differences when they truly exist. This means smaller samples can be used to reject a null hypothesis.

A second assumption of parametric tests is homogeneity in variance. This means the variance should not change systematically throughout the data: the variance within each population is equal for all populations. If the variances are not homogenous between populations, they are heteroscedastic. This assumption is key to use the two-sample t-test and ANOVA. A non-parametric equivalent of ANOVA is the Krustal-Wallis test which compares three or more independent groups on a continuous outcome; this violates the assumption of homogeneity of variance between groups in the parametric ANOVA analysis.

A third assumption of parametric data is interval data which means that the points of the scale should be equal at all parts along the scale in parametric tests.

The fourth assumption of parametric tests is independence of data which means that the values corresponding to one subject do not influence the values corresponding to another subject: there is one measure per subject.

In conclusion, parametric tests should always be used when the above assumptions are met due to their greater statistical power. Nonparametric tests are only used when the scale level is not metric, the true distribution of the random variables isn’t known or the sample is too small to assume a normal distribution. Non parametric tests are thus more robust and can be calculated in many more situations.

Table:

Most common parametric and non-parametric tests for different sample sizes: for a parametric test there is always a non-parametric counterpart.

Sample Size	Parametric Tests	Nonparametric tests
One sample	Simple t-Test	Wilcoxon test for one sample
Two dependent samples	Paired Sample t-Test	Wilcoxon-Test
Two independent samples	Unpaired Sample t-Test	Mann-Whitney U Test
More than two independent samples	One factorial ANOVA	Kruskal-Wallis-Test
More than two dependent samples	Repeated Measures ANOVA	Friedman-Test
Correlation between two variables	Pearson Correlation	Spearman-Korrelation

Section 3: RNaseq Exercises

Exercise RSQ1

Question:

Write a short assay on RNA-seq analysis, based on the following questions: How does RNA-seq works? What are the key steps of a RNA-seq pipeline starting from the raw data? What are the challenges of RNA-seq?

Answer:

The RNA-seq pipeline begins with sequencing, followed by data analysis and finally functional and differential expression analysis.

The TruSeq Stranded mRNA RNA Kit (figure 1) is the chosen RNaseq workflow to analyse the gene expression profiling of raw biological samples through high throughput sequencing technology. This is because relative abundance of mRNA is enriched in the sample to increase the accuracy of gene expression measurements. This is done by polyA enrichment methods to isolate mRNA molecules. After enrichment, adjusted RNA is cleaned, fragmented and primed in a single step. This is followed by the synthesis of the first cDNA strand and the addition of an actinomycin compound which prevents simultaneous second strand synthesis. The RNA template is degraded, ensuring only the second cDNA strand will be produced in the next synthesis step. The second cDNA strand synthesis uses dUTP nucleotides to differentiate between the two strands of DNA after synthesis. The resulting double stranded DNA is prepared for adapter ligation via adenylation of the 3' end (this allows hybridisation of cDNA with thymine on the 3' end of the adaptors). After adenylation, adaptors are ligated onto 3' ends of cDNA and they enzymatically remove dUTPs. Lacking dUTPs, the fragments are enriched via PCR amplification and the resulting library is validated prior to sequencing: this is done quantitatively with qPCR and qualitatively with a bioanalyser to verify if there is enough good quality DNA.

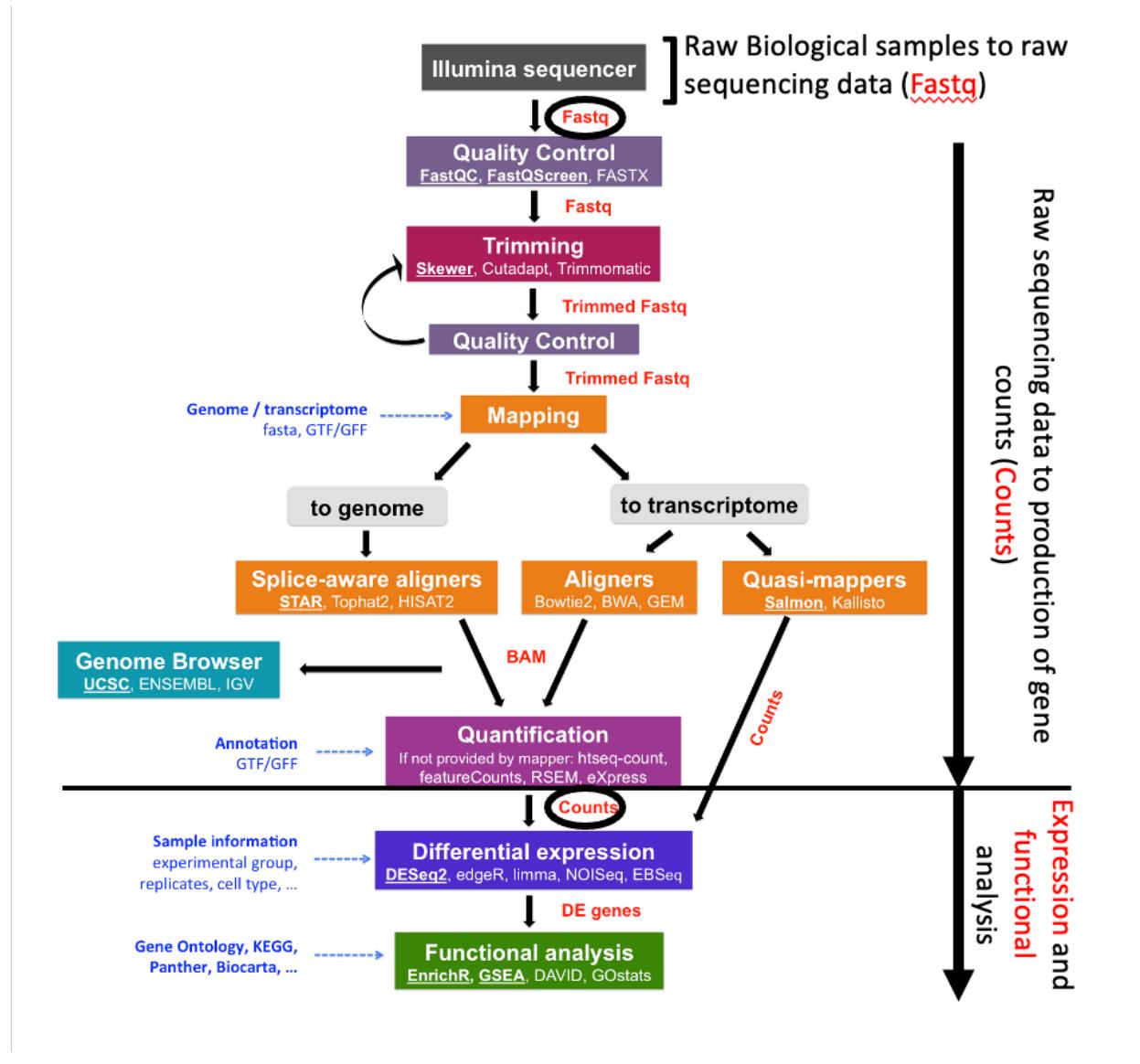
The raw sequencing data output can be in a .bcl format which is sorted to separate reads and converted to fastq files via demultiplexing. It can be paired-end data (PET) or single end data (SET). The RNA-seq analysis pipeline (figure 1) is applied to these raw data fastq files: read alignment, transcript assembly and gene quantification. Quality assessment is performed pre-alignment with FastQC with Cutadapt and post-alignment with Picard Tools. RNaseq reads only map to areas of the genome that code for RNA transcripts (not whole genome sequencing) so reads alignment is performed by special splice-aware mapping aligners like STAR. The two-pass method of STAR aligns each read group separately against a reference genome (created in advance) and merges resulting alignments into one and, including a splice junction detection step to generate the final alignment. Khallisto can also be used for a faster k-mer pseudo-alignment process to a transcriptome reference that may cope better with sequencing errors. STAR can generate an Aligned Genomic BAM; for aliquots with at least one set of paired-end reads an Aligned Transcription BAM and Aligned Chimeric BAM can also be produced. Transcription Fusion data of gene names, junction read count and breakpoint information can be generated using the STAR-fusion pipeline. STAR generates raw count files following alignment excluding reads that are mapped to more than one different gene. These RNA-Seq expression level read counts need to be normalised using count transformations and basic annotations which can be done with FPKM, FPKM-UQ or TPM. It is easier to interpret TPM values than FPKM values as the sum of normalized values will always equal 10^6 for each library. Differential expression analysis can be done on this data in R or functional gene ontology analysis using online tools such as GOrilla.

Overall RNA-seq limitations boil down to its high cost, time intensity and onerous processes mainly due to the large volumes of data generated and input. If a sequencer's accuracy is 99% this means 1 base pair is read incorrectly out of 100 which can add up quickly considering the high amount of output. This can be circumvented with higher coverage per genome which can be longer and more costly. If there is more

or less DNA than library protocol this can also cause less efficient sequencing reaction runs which develops low quality runs due to read problems (flow cell saturation or reduced coverage). If duplicate fragments are produced, the sequencing reaction will be bias towards them due to overrepresentation. It is difficult to accurately quantify expression levels of rare transcripts due to limited sequencing depth/coverage. There is also a risk of bias in the results depending on the alignment process and normalisation method used. Additionally, there is often a lack of reference genomes or transcriptomes for certain organisms, making it difficult to accurately map the sequence reads. Finally, there can be a lack of knowledge about the biology of the organism being studied, making it difficult to interpret the results.

Image

RNAseq Workflow ('Read Mapping'. Biocorecrg.Github.io, 2022):



Exercise RSQ2

Question:

Use the fastq files and perform a quality control of the reads. Discuss and visualise the results.

Code:

Code to download homebrew, wget and then FastQC from babraham bioinformatics.

```
#Downloading homebrew using the curl function in terminal
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

#Configuring homebrew
echo '# Set PATH, MANPATH, etc., for Homebrew.' >> /Users/judepops/.zprofile
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> /Users/judepops/.zprofile
eval "$(/opt/homebrew/bin/brew shellenv)"

#Downloading wget
brew install wget

#Downloading fastqc using the link from babraham bioinformatics
wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.9.zip

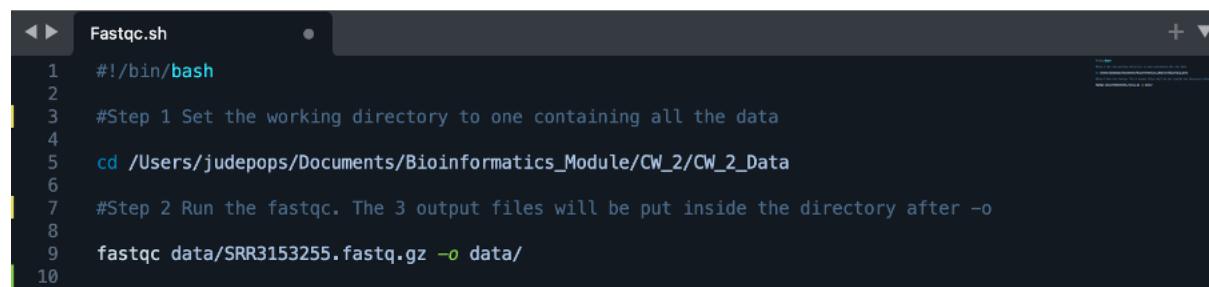
#Creating a fastqc shortcut to call (without specifying entire path)

## Navigating to the expanded folder containing the fastqc
cd Downloads/FastQC/
# giving permissions
chmod +x fastqc
## Navigating to fastqc via absolute path and putting it into the bin
sudo ln -s /Users/judepops/Downloads/FastQC/fastqc /usr/local/bin/fastqc
## Allowing fastqc to run on the mac in terminal without shell denying access. This is done via the sudo
sudo su
chmod 777 /Users/judepops/Downloads/FastQC/fastqc
```

Creating a script to run the fastqc

```
#Moving to the script location
#Makign a script and setting its output to a new data subdirectory made inside the CW_2_Data directory
cd Documents/Bioinformatics_Module/CW_2/CW_2_Data/script/

#Give permissions to the script
chmod 755 Fastqc.sh
```



```
Fastqc.sh
1 #!/bin/bash
2
3 #Step 1 Set the working directory to one containing all the data
4
5 cd /Users/judepops/Documents/Bioinformatics_Module/CW_2/CW_2_Data
6
7 #Step 2 Run the fastqc. The 3 output files will be put inside the directory after -o
8
9 fastqc data/SRR3153255.fastq.gz -o data/
10
```

Running the fastqc script in terminal

```
#Running the script via bash  
./Fastqc.sh
```

```
(base) judepops@Judes-MacBook-Pro script % ./Fastqc.sh  
Started analysis of SRR3153255.fastq.gz  
Approx 5% complete for SRR3153255.fastq.gz  
Approx 10% complete for SRR3153255.fastq.gz  
Approx 15% complete for SRR3153255.fastq.gz  
Approx 20% complete for SRR3153255.fastq.gz  
Approx 25% complete for SRR3153255.fastq.gz  
Approx 30% complete for SRR3153255.fastq.gz  
Approx 35% complete for SRR3153255.fastq.gz  
Approx 40% complete for SRR3153255.fastq.gz  
Approx 45% complete for SRR3153255.fastq.gz  
Approx 50% complete for SRR3153255.fastq.gz  
Approx 55% complete for SRR3153255.fastq.gz  
Approx 60% complete for SRR3153255.fastq.gz  
Approx 65% complete for SRR3153255.fastq.gz  
Approx 70% complete for SRR3153255.fastq.gz  
Approx 75% complete for SRR3153255.fastq.gz  
Approx 80% complete for SRR3153255.fastq.gz  
Approx 85% complete for SRR3153255.fastq.gz  
Approx 90% complete for SRR3153255.fastq.gz  
Approx 95% complete for SRR3153255.fastq.gz  
Analysis complete for SRR3153255.fastq.gz  
(base) judepops@Judes-MacBook-Pro script %
```

###

Code

Code to save fastq files into a variable and create a quality score boxplot

```
#Setting working directory  
#setwd('/Users/judepops/Documents/Bioinformatics_Module/CW_2/CW_2_Data/SRR3153255.fastq.gz')  
  
#Loading fastq files into a dataframe  
fastq.files <- list.files(path = '/Users/judepops/Documents/Bioinformatics_Module/CW_2/CW_2_Data/data/SRR3153255',  
pattern = ".fastq.gz$", full.names = TRUE)  
fastq.files  
  
## character(0)  
  
#Loading Rsubread for quality control  
library(Rsubread)  
  
#Extracting quality scores  
qs <- qualityScores(filename='/Users/judepops/Documents/Bioinformatics_Module/CW_2/CW_2_Data/data/SRR3153255')
```

```

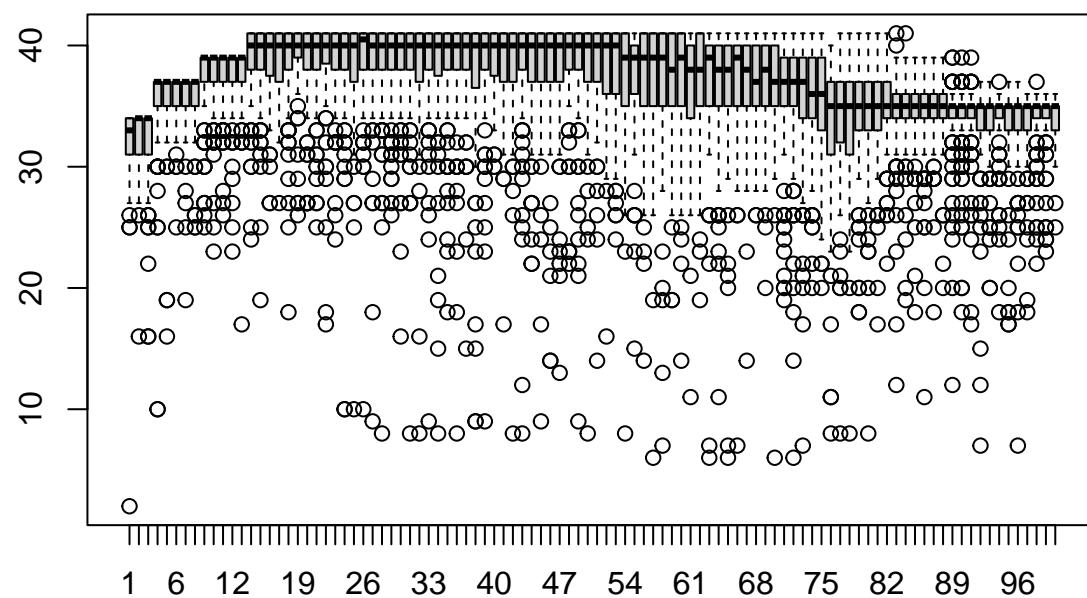
## qualityScores Rsubread 2.12.2
##
## Scan the input file...
##   1000000 reads have been scanned in 0.8 seconds.
##   2000000 reads have been scanned in 1.7 seconds.
##   3000000 reads have been scanned in 2.5 seconds.
##   4000000 reads have been scanned in 3.4 seconds.
##   5000000 reads have been scanned in 4.3 seconds.
##   6000000 reads have been scanned in 5.1 seconds.
##   7000000 reads have been scanned in 6.0 seconds.
##   8000000 reads have been scanned in 6.9 seconds.
##   9000000 reads have been scanned in 7.8 seconds.
##  10000000 reads have been scanned in 8.6 seconds.
## Totally 10873108 reads were scanned; the sampling interval is 108731.
## Now extract read quality information...
##   1000000 reads have been scanned in 10.2 seconds.
##   2000000 reads have been scanned in 11.0 seconds.
##   3000000 reads have been scanned in 11.9 seconds.
##   4000000 reads have been scanned in 12.7 seconds.
##   5000000 reads have been scanned in 13.5 seconds.
##   6000000 reads have been scanned in 14.4 seconds.
##   7000000 reads have been scanned in 15.2 seconds.
##   8000000 reads have been scanned in 16.0 seconds.
##   9000000 reads have been scanned in 16.8 seconds.
##  10000000 reads have been scanned in 17.7 seconds.
##
## Completed successfully. Quality scores for 100 reads (equally spaced in the file) are returned.

```

Plot (RESULTS):

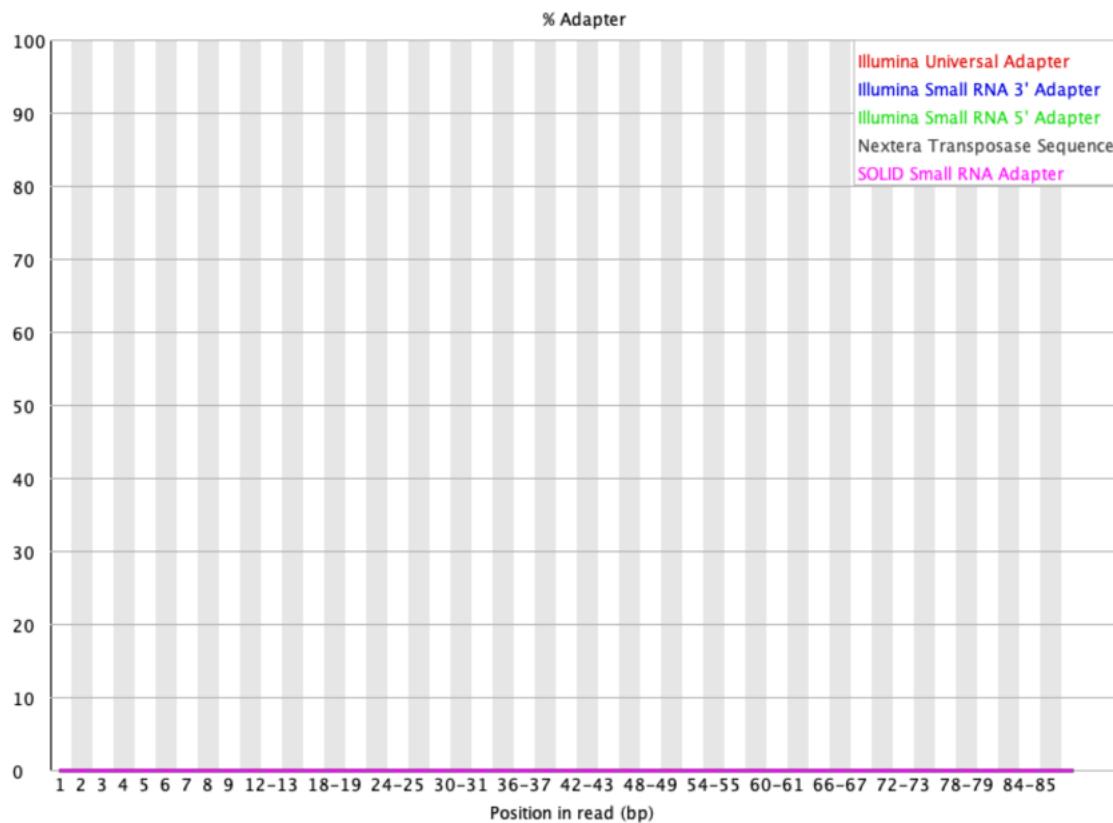
A boxplot to show the distribution of quality scores for 100 reads using the boxplot function

```
#Visualising the reads
boxplot(qs)
```



RNAseq QC results

Adapter Content

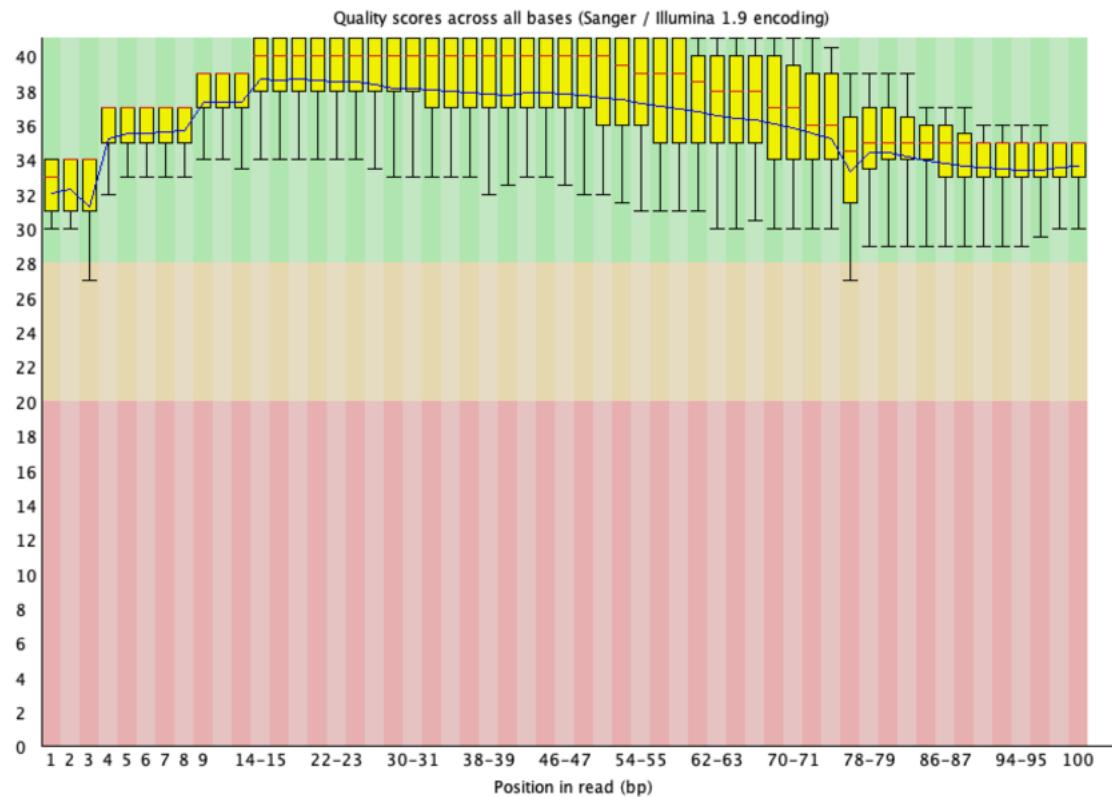


Basic Statistics

Measure	Value
Filename	SRR3153255.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	10873108
Sequences flagged as poor quality	0
Sequence length	21-100
%GC	44



Per base sequence quality



Discussion:

What the visualised results tell us?

The basic statistics tell us some key things. The per base sequence quality tells us if any of the reads have poor quality (FRET scores less than 20) they will need to be trimmed but that is not the case.

The reads for most platforms drop towards the ends.

Overrepresented sequences can be viewed in adaptor content.

There are no poor quality bases and no adapter sequences so trimming is not required for our data.

There is also debate over whether trimming improves the mapping of RNAseq reads since a lot of aligners also soft-clip sequencing reads which mask portions of reads that do not align to the reference. Thus trimming may not make much of a difference.

Trimming the data anyway with Rshortread

Exercise RSQ3

Question:

Use the count matrix and the related annotation present in the downloaded folder. Perform a differential expression analysis and discuss each step of the analysis along with the results. What are the top 10

differential expressed genes?

Code:

Preparing the provided raw counts data and column data for a DEseq analysis

```
# DIFFERENTIAL GENE EXPRESSION ANALYSIS

#Load libraries
library(DESeq2)
library(ggplot2)

#Load in the counts table
counts_data <- read.delim('/Users/judepops/Documents/Bioinformatics_Module/CW_2/CW_2_Data/RNAseq/SRP049')

#Preparing the counts dataframe but moving gene names in counts_data to row names
counts_data=counts_data[counts_data$width>0,] #extracting all rows where width is greater than 0.
#This remove duplicate genes as some have no width but still appear in the data frame
counts <- as.matrix(counts_data[-1], header = T) #Creating a new counts matrix which is
#the counts dataframe without the first column
rownames(counts) <- counts_data$X #moving the gene names (column X) in counts_data
#to the rownames of counts

#Removing the width column from counts
counts <- as.matrix(subset(counts, select = c(-width)))

#Reading in the column data file
coldata_file <- read_delim('/Users/judepops/Documents/Bioinformatics_Module/CW_2/CW_2_Data/RNAseq/SRP049')

## Warning: One or more parsing issues, call 'problems()' on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

coldata <- as.data.frame(coldata_file$group) #creating a dataframe out of the coldata_file's groups
rownames(coldata)<-coldata_file$source_name #moving the source name (aka the column names in the counts
#the rownames of the coldata dataframe that was just created
colnames(coldata) <- "Condition"
coldata <- coldata %>%
  mutate(Condition = ifelse(Condition == "lung carcinoma cells_overexpression_EHF CASE", "CASE", 'CTRL'))
#merging the columns for CTRL as there is an odd column present which will cause problems later. This is
#done using the elseif function that replaces the condition with the first string after the comma and
#everything else with the second string after the comma
```

Beginning the DESeq2 analysis

```
# Starting DESeq functions

#Creating dds
dds <- DESeqDataSetFromMatrix(countData = counts, colData = coldata, design = ~Condition)

## converting counts to integer mode
```

```

## Warning in DESeqDataSet(se, design = design, ignoreRank): 2 duplicate rownames
## were renamed by adding numbers

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors

#For each gene, we count the total number of reads for that gene in all samples
#and remove those that don't have at least 1 read.
dds <- dds[ rowSums(DESeq2::counts(dds)) > 1, ]

#Running deseq on dds
dds <- DESeq(dds)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

#print(dds)

#Getting the results
DEresults = results(dds, contrast = c("Condition", 'CASE', 'CTRL'))
#contrast tells the function what samples to compare.
#this is especially useful if there are more than two samples

# RESULTS

#The log2foldchange is important: if it is negative that means the CASE samples were
#downregulated for that gene and if it is positive that means it was upregulated
#in the control samples.

#The basemean is the average of the normalised counts taken over all the samples
#the lfcSE are the standard error estimates for the log2foldchange
#the stat

#The padv value is more important than the p value because it has to be corrected for
#multiple testing. There were alot of genes tested but some of those might be
#significant just by chance. It is the corrected p value. This is because
#when a statistical test is performed there is a p value of 0.05 which means
#5% of the diff expressed genes are due to random chance and the drug has
#no real effect on them. A lot of false positive accounted for by padv.

```

Results

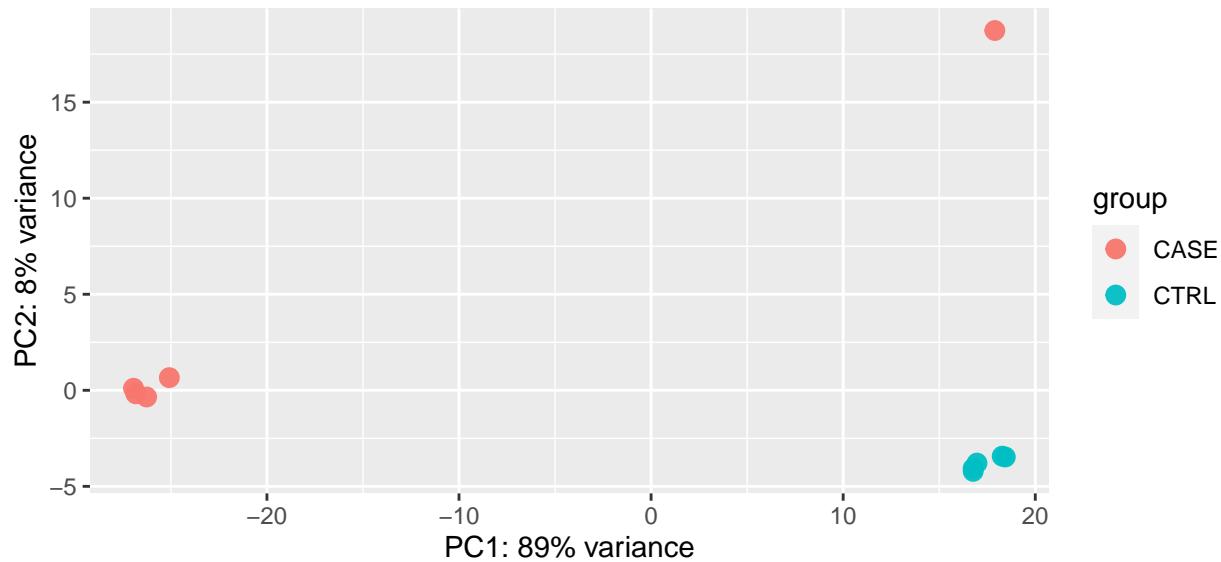
Visualising the expression

```
# Looking at a PCA plot of the data
```

```
#This plot shows the relationship between the expression levels of the genes being compared. The PCA pl
```

```
vsdata <- vst(dds, blind = FALSE) #running vst and saving it as a new variable
```

```
PCA_PLOT <- plotPCA(vsdata, intgroup = 'Condition') #saving PCA plot made using vsdata produced by vst .  
PCA_PLOT
```

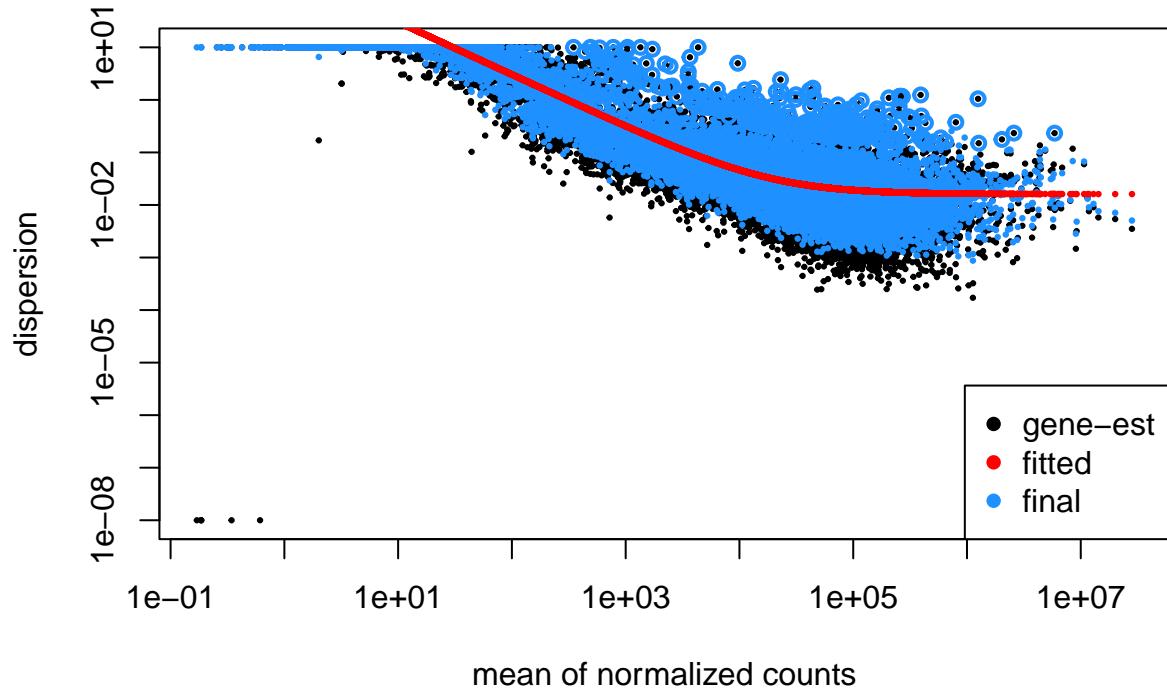


```
#PCA plot uses VST normalisation techniques to
```

```
# Looking at the dispersion of the data
```

```
#This plot shows the dispersion of the data. It is the variability between replicates as a  
#function of normalised read counts. As the read counts get higher, there is less variations  
#so there is a decrease in the slope of the fitted slope.
```

```
plotDispEsts(dds)
```



```
#The aim is for the red line to trend below one: the lower it is, the better
```

Computing and visualised the top differentially expressed genes in a heatmap

```
# Re-ordering results by adjusted p value
DEresults <- DEresults[order(DEresults$padj),]

#Top 10 differentially expressed genes
TOP_10_DGE <- head(DEresults[order(DEresults$padj),], 10)
TOP_10_DGE <- rownames(TOP_10_DGE)

# We are using vst to compute the top 10 differentially expressed genes. This
# is because counts are not proportional to expression across genes. The count
# numbers for a gene can be related to the length of the transcript,
# so it is best to correct for that using a normalisation technique
# such as VST or TPM.

library(pheatmap)
#Creating a file for count visualization by vst normalization
vsd <- vst(dds)
# We use the vst function which stands for variance stabilizing transformation. This transformation adj-
# The vst values are stored in vsd which as a package from DESeq2

#
mat <- assay(vsd)[head(match(row.names(DEresults), row.names(vsd)), 10), ] #We compare the normalised
```

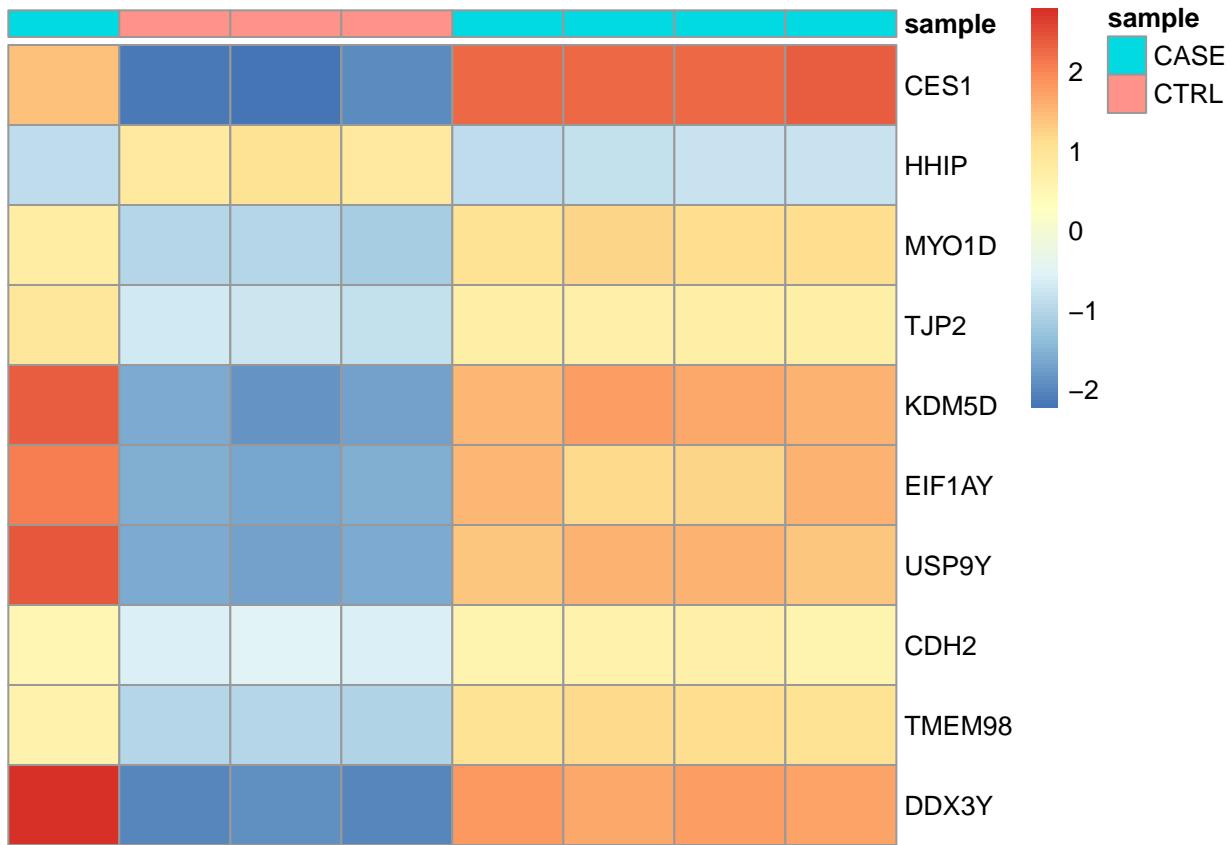
```

#just transformed package (vsd) with the DEresults, match row.names and extract the 10 top
mat <- mat - rowMeans(mat)
df <- as.data.frame(colData(vsd)[, c("Condition")])
df <- data.frame(sample=df$`colData(vsd)[, c("Condition")]`)
rownames(df) <- colnames(mat)

#Reorder sample column for the heatmap
#           CTRL          CASE
mat <- mat[,c( 5, 6, 7, 8,      1, 2, 3, 4)]

#Produce heatmap
pheatmap(mat, annotation_col = df, cluster_rows = F , cluster_cols = F ,show_colnames = F)

```



Exercise RSQ4

Question:

With BioMart, find the function of the top 10 differential expressed genes, using the Gene Ontology Biological Processes.

Dataset: Human genes (GRCh38.p13)

Answer:

GJB6

GJG6 is a human gene known as 'gap junction protein beta 6'.

MMRN1

FCN3

FNDC7

CXCL14

BATF

SERPINB12

SPINK6

AGTR2

C12ORF74