Capstone Project - Predicting Depression from plasma measurements

Zsombor Szoke-Kovacs

2022-04-22

#**Introduction**

#**Project Background**

#*Methods and Data Analysis Workflow*

#**DATA PREPARATION** First we load the data sheet downloaded from the Metabolomics website: https://www.metabolomicsworkbench.org/data/DRCCMetadata.php?Mode=Study&StudyID=ST000062&StudyType=MS&ResultType=1 I have edited the data set; some of the measured molecules have been removed from the list due to simplicity and also due to these did not have names, only ID numbers. A simplified data sheet is used to do the analysis.

```
library("readxl")
library("dplyr")
library("ggplot2")
library('corrplot')
library(purrr)
# Here, I set the file to a path on my computer, but once this is saved to a
# different computer, the path will need to be updated. The simplified data sheet
# can also be downloaded from my git repository.
file_original <- "~/Desktop/HarvardX, EdX, Data Science/Capstone Project/Capstone Project - Chosen Proje
temp_file <- read_excel(file_original)
# Converting the temp file into a transposed data frame.
file_df <- as.data.frame(t(temp_file))
# removing unnecessary rows/columns
data <- file_df[c(-2,-3,-4),c(-1,-3,-4,-5,-7)]
```

To be able to work with the data sheet, new column names are introduced:

```
# Adding new column names for the molecules and the arm:
data[1,1] <- 'Samples'
data[1,2] <- 'Arm'
colnames(data) <-data[1,]
# Remove first row from the data frame:
data <- data[-1,]
# By investigating the data, we can see that measurements come from two groups:
# Group 1 (control) and Group 2 (patients diagnosed with depression):
data %>% group_by(Arm) %>% summarise(n = n())
```

```
## # A tibble: 2 x 2
##   Arm                    n
##   <chr>              <int>
## 1 Group 1 - Score 0     48
## 2 Group 2 - Score 50    49
```

To analyse the relationship between the different measurements between the two groups, I first separate the two arms and remove unnecessary columns:

```
# Separate the two arms from the data set:
group_1 <- data %>% group_by(Arm) %>% filter(Arm == "Group 1 - Score 0")
group_1_truncated <- group_1[, c(-1, -2)]
group_2 <- data %>% group_by(Arm) %>% filter(Arm == "Group 2 - Score 50")
group_2_truncated <- group_2[, c(-1, -2)]
```

#**DATA ANALYSIS** ##*Data distribution* To be able to compare the differences between the two groups, I next look at the distribution of the data in each measurement and in each arm. I generated and investigated the distribution of each measured parameter within the two groups using the codes below. However, due to these produce many plots (144 plots per arm), I commented these out in the .rmd file.

```
# Group 1:

#for (i in group_1_truncated){
# plot <- group_1_truncated %>% ggplot(aes(x = as.numeric(i))) +
#  geom_density()
# print(plot)
#}

# Group 2

#for (i in group_2_truncated){
#  plot <- group_2_truncated %>% ggplot(aes(x = as.numeric(i))) +
#    geom_density()
#  print(plot)
#}
```

Instead, I used Shapiro-Wilk's method (http://www.sthda.com/english/wiki/normality-test-in-r) to get a value of the normality for each measured parameter. The null hypothesis of this tests is that "the sample distribution is normal". So, if the p-value is >0.05, that implies that the distribution of the data is not significantly different from the normal distribution. In other words, if the p-value is >0.05 we can assume normality. First, I loop through the truncated and transposed list and generate Shapiro-Wilk's test for each column in the data set. I use the magicfor library to record p-values in a vector:

```
library(magicfor)
# Group 1:
magic_for(print)
for (c in group_1_truncated){
  # shap test for each col
  shap_test <- shapiro.test(as.numeric(c))
  output <- shap_test$p.value
  print(output)
}
```

```
## [1] 0.03075705
## [1] 0.08522262
## [1] 2.431518e-08
## [1] 0.4683182
## [1] 0.003183624
## [1] 0.4751875
## [1] 0.237512
## [1] 0.00431784
## [1] 0.1507297
## [1] 0.03231325
## [1] 0.2276085
## [1] 0.02835112
## [1] 0.9503814
## [1] 0.5639228
## [1] 0.0001596303
## [1] 0.0001859755
## [1] 3.838608e-10
## [1] 0.5760497
## [1] 0.5129488
## [1] 0.001635735
## [1] 2.859378e-14
## [1] 5.950413e-06
## [1] 0.7951635
## [1] 0.7440938
## [1] 0.0001307812
## [1] 1.769107e-10
## [1] 5.592586e-07
## [1] 2.080603e-14
## [1] 7.373533e-08
## [1] 0.2458592
## [1] 0.1436593
## [1] 4.844873e-13
## [1] 2.119992e-07
## [1] 0.0008157663
## [1] 0.02371112
## [1] 0.006136389
## [1] 0.8390248
## [1] 6.586207e-12
## [1] 0.2145127
## [1] 0.001970347
## [1] 0.002844868
## [1] 0.5173994
## [1] 0.04001211
## [1] 0.0001117784
## [1] 0.1276793
## [1] 0.02324038
## [1] 0.005191888
## [1] 0.008093098
## [1] 6.88849e-05
## [1] 7.739744e-07
## [1] 0.0008360514
## [1] 0.8068052
## [1] 0.1150337
## [1] 0.01927027
## [1] 0.5908586
## [1] 0.005363307
## [1] 0.1046918
## [1] 0.08097101
## [1] 1.582163e-13
## [1] 3.709977e-07
## [1] 2.203858e-09
## [1] 0.04381657
## [1] 0.7613584
## [1] 0.8516382
## [1] 2.83857e-05
## [1] 1.156385e-08
## [1] 0.1914479
## [1] 0.0895071
## [1] 3.574687e-06
## [1] 4.420659e-07
## [1] 5.170312e-10
## [1] 7.584198e-06
## [1] 0.05041785
## [1] 2.896216e-07
## [1] 0.03230108
## [1] 0.0001333469
## [1] 7.273573e-10
## [1] 3.547011e-13
## [1] 0.008958414
## [1] 0.351065
## [1] 0.0004660933
## [1] 0.000318582
## [1] 4.474704e-08
## [1] 6.085384e-09
## [1] 0.3867491
## [1] 0.06902782
## [1] 0.5782418
## [1] 8.471097e-07
## [1] 0.007310657
## [1] 1.555231e-05
## [1] 3.457164e-08
## [1] 0.183637
## [1] 1.163639e-05
## [1] 3.332581e-07
## [1] 0.0002195566
## [1] 0.01156348
## [1] 0.08063735
## [1] 0.02113055
## [1] 1.603749e-08
## [1] 2.500726e-14
## [1] 6.602879e-05
## [1] 0.02840995
## [1] 0.5292255
## [1] 0.5413767
## [1] 0.008912426
## [1] 2.88464e-11
## [1] 0.02485436
## [1] 1.950487e-11
## [1] 0.07610434
## [1] 0.5720348
## [1] 2.000887e-06
## [1] 9.319506e-05
## [1] 0.6336717
## [1] 1.371973e-13
## [1] 0.001600674
## [1] 0.2434581
## [1] 0.007982874
## [1] 1.671043e-08
## [1] 0.008609122
## [1] 2.115987e-12
## [1] 0.0834015
## [1] 1.176368e-05
## [1] 0.004877992
## [1] 6.80456e-06
## [1] 0.003929345
## [1] 0.2785123
## [1] 2.755291e-07
## [1] 0.006870093
## [1] 6.047253e-12
## [1] 2.886682e-10
## [1] 3.685127e-08
## [1] 0.0002518801
## [1] 0.03174344
## [1] 0.1348054
## [1] 9.868894e-09
## [1] 3.383021e-08
## [1] 0.00286715
## [1] 0.01370772
## [1] 0.467026
## [1] 0.00181973
## [1] 2.310402e-07
## [1] 8.595669e-11
## [1] 0.2939691
```

```r
# Saving printed p-values as a vector:
p_values_vector_group_1 <- magic_result_as_vector()
# Binding vector to the original data, so the last row is the p-value from the
# Shapiro-Wilk's test:
group_1_truncated_with_pvalues <- rbind(group_1_truncated,p_values_vector_group_1)

# Group 2:
magic_for(print)
# For loop for collecting all p-values and printing them to the console:
for (c in group_2_truncated){
  # shap test for each col
  shap_test <- shapiro.test(as.numeric(c))
  output <- shap_test$p.value
  print(output)
}
```

```
## [1] 0.01268071
## [1] 0.5320017
## [1] 1.046158e-14
## [1] 0.9271982
## [1] 0.3338198
## [1] 0.002995611
## [1] 0.6509885
## [1] 0.3337033
## [1] 0.2176103
## [1] 0.2517892
## [1] 5.691081e-05
## [1] 0.008071441
## [1] 0.9496703
## [1] 0.003835506
## [1] 1.276141e-06
## [1] 4.014598e-06
## [1] 7.396027e-15
## [1] 0.003291311
## [1] 2.583564e-05
## [1] 1.078828e-09
## [1] 6.213368e-12
## [1] 9.487599e-10
## [1] 0.4116826
## [1] 5.813022e-08
## [1] 0.004757193
## [1] 1.024567e-07
## [1] 5.331477e-08
## [1] 2.607211e-08
## [1] 2.121411e-14
## [1] 0.6218422
## [1] 0.3725114
## [1] 1.326105e-12
## [1] 3.895945e-09
## [1] 1.232802e-11
## [1] 0.09996526
## [1] 0.4281124
## [1] 0.214296
## [1] 2.509511e-10
## [1] 2.26814e-08
## [1] 8.840691e-12
## [1] 3.020783e-07
## [1] 0.0003244317
## [1] 0.3325172
## [1] 2.297722e-06
## [1] 0.1971648
## [1] 7.522532e-08
## [1] 0.1380847
## [1] 0.002045618
## [1] 2.969678e-10
## [1] 2.122113e-09
## [1] 0.0001097306
## [1] 0.2254222
## [1] 1.308554e-05
## [1] 0.00055218
## [1] 0.1044934
## [1] 0.2794276
## [1] 0.03188302
## [1] 0.1232555
## [1] 1.348212e-11
## [1] 6.331095e-14
## [1] 4.309545e-12
## [1] 0.2554861
## [1] 5.045201e-08
## [1] 0.01483991
## [1] 0.03036964
## [1] 4.80947e-12
## [1] 0.2054814
## [1] 0.0001040936
## [1] 3.371332e-09
## [1] 0.006638151
## [1] 1.565909e-08
## [1] 0.002225898
## [1] 0.2305581
## [1] 2.791335e-11
## [1] 0.04784196
## [1] 1.563023e-09
## [1] 8.730435e-06
## [1] 8.412952e-11
## [1] 0.6350672
## [1] 0.2121618
## [1] 1.692299e-05
## [1] 1.901465e-05
## [1] 2.076581e-09
## [1] 1.23874e-07
## [1] 2.796222e-07
## [1] 0.9728612
## [1] 0.6713909
## [1] 2.197602e-08
## [1] 7.510541e-06
## [1] 8.175529e-06
## [1] 0.000740605
## [1] 0.3345837
## [1] 0.0009565749
## [1] 6.451756e-12
## [1] 3.312299e-05
## [1] 0.7167231
## [1] 4.774386e-12
## [1] 0.003379862
## [1] 6.348476e-09
## [1] 5.033355e-14
## [1] 0.0001545562
## [1] 0.0001110415
## [1] 0.02478078
## [1] 0.02641833
## [1] 0.1739291
## [1] 2.784913e-11
## [1] 1.167685e-05
## [1] 3.381197e-10
## [1] 0.01268761
## [1] 0.06897864
## [1] 4.75765e-07
## [1] 4.527211e-05
## [1] 5.633739e-11
## [1] 4.312549e-15
## [1] 1.72198e-06
## [1] 0.1555038
## [1] 0.003044546
## [1] 0.4012447
## [1] 0.0002663313
## [1] 3.665618e-14
## [1] 6.023239e-05
## [1] 1.736724e-06
## [1] 0.05283555
## [1] 8.825776e-07
## [1] 6.194916e-13
## [1] 0.1922048
## [1] 6.154861e-15
## [1] 0.08027211
## [1] 9.050325e-13
## [1] 2.338873e-06
## [1] 2.232031e-08
## [1] 0.05178879
## [1] 1.921007e-05
## [1] 0.0005122628
## [1] 0.0004112289
## [1] 0.0005876738
## [1] 1.251389e-06
## [1] 2.130454e-07
## [1] 0.0724654
## [1] 0.002222042
## [1] 1.046887e-09
## [1] 1.936056e-10
## [1] 6.314063e-05
```

```r
# Saving printed p-values as a vector:
p_values_vector_group_2 <- magic_result_as_vector()
# Binding vector to the original data, so the last row is the p-value from the
# Shapiro-Wilk's test:
group_2_truncated_with_pvalues <- rbind(group_2_truncated,p_values_vector_group_2)
# Remove magicalization:
magic_free()
# The last row in these two data frames are the Shapiro-Wilk's p-values:
group_1_truncated_with_pvalues %>%
  summarise(Arm = 'Group 2',
            nrow = dim(group_1_truncated_with_pvalues)[1],
            ncol = dim(group_1_truncated_with_pvalues)[2])
```

```
## # A tibble: 1 x 3
##   Arm      nrow  ncol
##   <chr>   <int> <int>
## 1 Group 2    49   143
```

```r
group_2_truncated_with_pvalues %>%
  summarise(Arm = 'Group 2',
            nrow = dim(group_2_truncated_with_pvalues)[1],
            ncol = dim(group_2_truncated_with_pvalues)[2])
```

```
## # A tibble: 1 x 3
##   Arm      nrow  ncol
##   <chr>   <int> <int>
## 1 Group 2    50   143
```

Now that I have the p-values for the Shapiro-Wilk's test for the measured parameters from each arm, I transposevthe data frames, so the p-values are in a separate column and the data frame in tidy format:

```
# Group 1 - transpose
group_1_tidy <- as.data.frame(t(group_1_truncated_with_pvalues))
# adding on extra row that will be used for the column names
group_1_tidy<- add_row(group_1_tidy, .before = 1)
# using sample names for column names
group_1_tidy[1,1:48] <- group_1$Samples
# adding name for extra column
group_1_tidy[1,49] <- "Shapiro-Wilk's p-values"
# use first row as column names
colnames(group_1_tidy) <- group_1_tidy[1,]
group_1_tidy <- group_1_tidy[-1,]
# The last column is the Shapiro-Wilk's p-values
group_1_tidy %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy)[1],
                           ncol = dim(group_1_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 1  143   49
```

```
# Group 2 -transpose
group_2_tidy <- as.data.frame(t(group_2_truncated_with_pvalues))
# adding on extra row that will be used for the column names
group_2_tidy<- add_row(group_2_tidy, .before = 1)
# using sample names for column names
group_2_tidy[1,1:49] <- group_2$Samples
# adding name for extra column
group_2_tidy[1,50] <- "Shapiro-Wilk's p-values"
# use first row as column names
colnames(group_2_tidy) <- group_2_tidy[1,]
group_2_tidy <- group_2_tidy[-1,]
# The last column is the Shapiro-Wilk's p-values
group_2_tidy %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy)[1],
                           ncol = dim(group_2_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 2  143   50
```

Now that I have the data for the two arms, together with the p-values for normal distribution, I filter the data to keep the measured parameters, where the distribution was approximately normal. In other words, I keep all measured parameters, where the p-value was >0,05:

```
group_1_tidy <- group_1_tidy %>% filter(`Shapiro-Wilk's p-values`>0.05)
group_2_tidy <- group_2_tidy %>% filter(`Shapiro-Wilk's p-values`>0.05)
# There are 97 and 112 measured parameters where the p-value is >0.05 in Group 1
# and Group 2, respectively. Group 1 has 48 patients, whereas Group 2 has 49. The extra
# column in each data frame is the Shapiro-Wilk's p-value.
group_1_tidy %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy)[1],
                           ncol = dim(group_1_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   97   49
```

```
group_2_tidy %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy)[1],
                           ncol = dim(group_2_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 2  112   50
```

Due to the number of the normally distributed measured parameters are different in the two groups, I will work with the list from the control group (Group 1 - baseline), where the normally distributed parameters were 97 (as opposed to Group 2 where it was 112). I use semi_join to keep only the records from Group 2, that have a match in Group 1.

```
# Adding row names as an extra column, so I can use semi_join:
group_1_tidy <- cbind(group_1_tidy, rownames = rownames(group_1_tidy))
group_2_tidy <- cbind(group_2_tidy, rownames = rownames(group_2_tidy))
# we should have one extra column in each data frame:
group_1_tidy %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy)[1],
                           ncol = dim(group_1_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   97   50
```

```
group_2_tidy %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy)[1],
                           ncol = dim(group_2_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 2  112   51
```

```
# Keep everything from Group 1 with a match in Group 2:
group_1_tidy <- semi_join(group_1_tidy, group_2_tidy, by = "rownames")
# Keep everything from Group 2 with a match in Group 1:
group_2_tidy <- semi_join(group_2_tidy, group_1_tidy, by = "rownames")
# Investigating the dimensions of the two newly generated data frames, we can see, that
# both arms have 78 measured parameters, as well as 48 and 49 sample count (plus the two columns
# with p-values and row names), respectively.
group_1_tidy %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy)[1],
                           ncol = dim(group_1_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   78   50
```

```
group_2_tidy %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy)[1],
                           ncol = dim(group_2_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 2   78   51
```

##*Two-sample t-test* In this next section, I will calculate two-sample t-tests for the selected parameters, so I can see if there is a significant difference in any parameters between the two groups. First, I transpose the data frame generated above and remove unnecessary rows.

```
# Transpose tidy data, so I can loop through the columns: Group 1
group_1_tidy_t <- as.data.frame(t(group_1_tidy))
# Removing last two rows with p-values and row names:
group_1_tidy_t <- group_1_tidy_t[c(-49,-50),]
# Transpose tidy data, so I can loop through the columns: Group 2
group_2_tidy_t <- as.data.frame(t(group_2_tidy))
# Removing last two rows with p-values and row names:
group_2_tidy_t <- group_2_tidy_t[c(-50,-51),]
# The two data set has 78 measured parameters and 48 and 49 samples, respectively:
group_1_tidy_t %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy_t)[1],
                             ncol = dim(group_1_tidy_t)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   48   78
```

```
group_2_tidy_t %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy_t)[1],
                             ncol = dim(group_2_tidy_t)[2])
```

```
##       Arm nrow ncol
## 1 Group 2   49   78
```

Now, that I have the two data frames with the same measured parameters in both, and all of the measurements show approximately normal distribution, I can test the vectors for significant differences:

```
# Two-sample t-test by looping through the columns:
magic_for(print)
for (j in seq(ncol(group_1_tidy_t))){
  testresults <- t.test(as.numeric(group_1_tidy_t[,j]), as.numeric(group_2_tidy_t[,j]))
  print(testresults$p.value)
}
```

```
## [1] 0.901493
## [1] 0.1944979
## [1] 0.5386687
## [1] 0.3329934
## [1] 0.9450576
## [1] 0.9846481
## [1] 0.5639318
## [1] 0.1921548
## [1] 4.942357e-05
## [1] 0.04571605
## [1] 0.01102079
## [1] 0.781581
## [1] 0.1682779
## [1] 0.0419918
## [1] 0.3691553
## [1] 0.3899988
## [1] 0.270009
## [1] 0.001947676
## [1] 0.09498654
## [1] 0.1854972
## [1] 0.3880945
## [1] 0.4855856
## [1] 0.4947039
## [1] 0.1916466
## [1] 0.1236281
## [1] 0.0005894853
## [1] 0.08959645
## [1] 0.2790141
## [1] 0.3967777
## [1] 0.00106033
## [1] 0.02861927
## [1] 0.1148812
## [1] 0.0752034
## [1] 0.275934
## [1] 0.002000258
## [1] 0.5798523
## [1] 0.8924322
## [1] 0.1573088
## [1] 0.05893773
## [1] 0.3321228
## [1] 0.1155188
## [1] 0.6678818
## [1] 0.1238112
## [1] 0.38568
## [1] 0.5241215
## [1] 0.06437688
## [1] 0.0001549329
## [1] 0.2873975
## [1] 0.6738781
## [1] 0.4753776
## [1] 0.09426011
## [1] 0.04067713
## [1] 0.01122825
## [1] 0.1402982
## [1] 0.5078799
## [1] 0.535678
## [1] 0.5192594
## [1] 0.3900095
## [1] 0.02823496
## [1] 0.5166175
## [1] 0.5375167
## [1] 0.08410382
## [1] 0.3515712
## [1] 0.9093331
## [1] 0.000952908
## [1] 0.1326068
## [1] 0.1060869
## [1] 0.5360112
## [1] 0.001629588
## [1] 0.9311662
## [1] 0.2831985
## [1] 0.086737
## [1] 0.1851199
## [1] 0.9373875
## [1] 0.2306857
## [1] 0.3310121
## [1] 0.9455772
## [1] 0.2934011
```

```
# Saving p-values from the two-sample t-test into a dataframe, and adding the
twosample_ttest <- magic_result_as_dataframe()
magic_free()
# Adding the names of the measured parameters to the p-values:
colnames(twosample_ttest)[1] <- 'rownames'
twosample_ttest$`rownames` <- colnames(group_1_tidy_t)
# Filtering out measured parameters that showed significant differences between
# the two groups:
twosample_ttest_significant <- twosample_ttest %>% filter(`testresults$p.value` <= 0.05)
# There are 15 measured parameters that show normal distribution, and there is a significant difference
# between the two groups:
twosample_ttest_significant
```

```
##                rownames testresults$p.value
## 1           stearic acid        4.942357e-05
## 2               sorbitol        4.571605e-02
## 3          shikimic acid        1.102079e-02
## 4                ribitol        4.199180e-02
## 5         pseudo uridine        1.947676e-03
## 6           nicotinamide        5.894853e-04
## 7           myo-inositol        1.060330e-03
## 8                mannose        2.861927e-02
## 9                lyxitol        2.000258e-03
## 10  heptadecanoic acid        1.549329e-04
## 11        glutaric acid        4.067713e-02
## 12        glutamic acid        1.122825e-02
## 13          citric acid        2.823496e-02
## 14         behenic acid        9.529080e-04
## 15 alpha-ketoglutarate        1.629588e-03
```

##*Correlation analysis* From the previous section, I have a set of measured parameters that show significant difference of the mean between the two arms. To see the actual relationship between the two groups, I will use correlation analysis for the 15 parameters. Initially, I will subset the two dataframes group_1_tidy and group_2_tidy, to only consist of the 15 parameters of interest.

```
group_1_final <- semi_join(group_1_tidy, twosample_ttest_significant, by = 'rownames')
group_1_final <- group_1_final[,c(-49,-50)]
```

```
group_2_final <- semi_join(group_2_tidy, twosample_ttest_significant, by = 'rownames')
group_2_final <- group_2_final[,c(-50,-51)]

# Here I have two dataframes from the two arms, one control and one diagnosed with depression,
# where the parameters of interest are included only. The dataframes consist of 48 and 49
# patients, respectively:
group_1_final %>% summarise(Arm = 'Group 1', nrow = dim(group_1_final)[1],
                            ncol = dim(group_1_final)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   15   48
```

```
group_2_final %>% summarise(Arm = 'Group 2', nrow = dim(group_2_final)[1],
                            ncol = dim(group_2_final)[2])
```

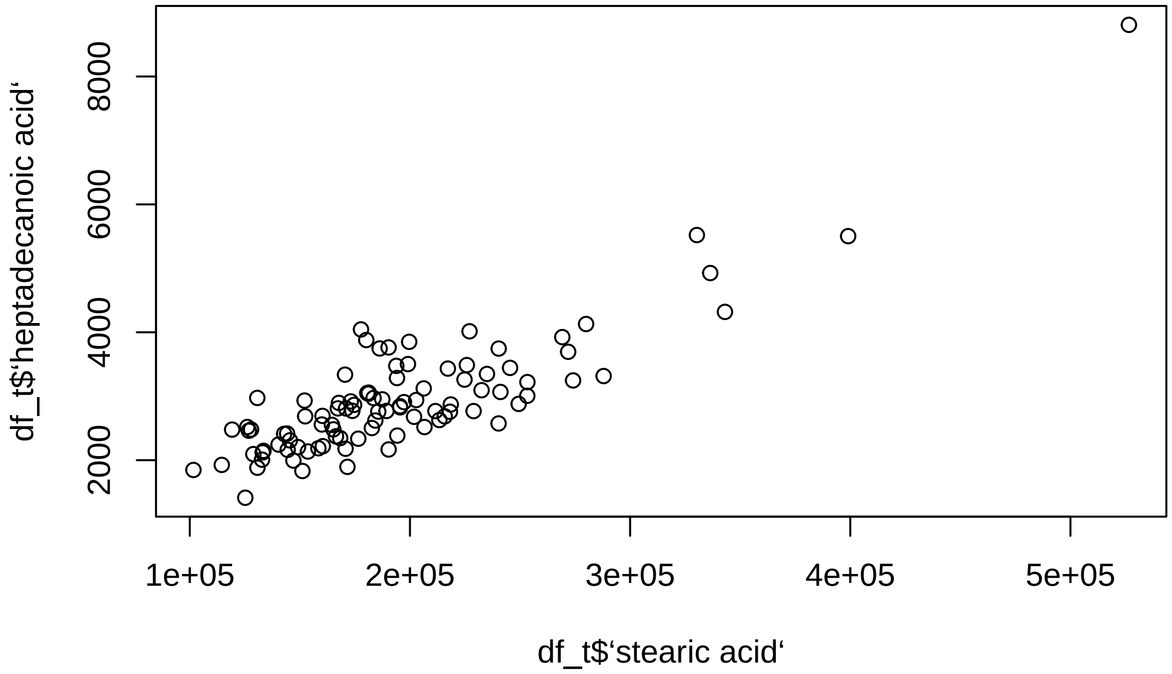
```
##       Arm nrow ncol
## 1 Group 2   15   49
```

Now that I have the two dataframes with the 15 measured parameters that showed approximately normal distribution and significant differences between the two groups, I will merge the two arms, and will generate a new dataframe with all of the subjects and the 15 measured parameters. I will use this dataframe for my further work:

```
# first, I create a new column in both dataframes, so I can merge these with
# left_join()
group_1_final <- cbind(group_1_final, rownames = rownames(group_1_final))
group_2_final <- cbind(group_2_final, rownames = rownames(group_2_final))
# Merging the two dataframes by rownames:
df <- left_join(group_1_final, group_2_final, by = "rownames")
# Adding rownames based on the rownames column
rownames(df) <- df$rownames
# Removing rownames column:
df <- subset(df, select = -rownames)
```

Testing for correlation:

```
# This is my dataset with all 15 parameters and the entire cohort.
# I now transpose it and will do a correlation analysis to see if any of these
# parameters are correlated:
df_t <- as.data.frame(t(df))
# A quick plotting of the data shows that there is a potential correlation between
# stearic acid and heptadecanoic acid: commented out so, otherwise many lots will be printed.
# plot(df_t)
plot(df_t$`stearic acid`, df_t$`heptadecanoic acid`)
```



```
# A correlation analysis between the two parameters shows a strong positive correlation
# with a value of 0.862:
cor.test(as.numeric(df_t$`stearic acid`), as.numeric(df_t$`heptadecanoic acid`))
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.numeric(df_t$‘stearic acid‘) and as.numeric(df_t$‘heptadecanoic acid‘)
## t = 16.583, df = 95, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8002618 0.9058056
## sample estimates:
##       cor
## 0.8621075
```

```
# Here, I convert all df_t to numeric, so I can do a correlation analysis:
df_num <-as.data.frame(sapply(df_t, as.numeric))
# This also shows, that the only correlation is between stearic acid and heptadecanoic acid:
cor_15_param <- as.data.frame(cor(df_num))
cor_15_param %>% filter(cor_15_param >= 0.7)
```

```
##                      stearic acid  sorbitol shikimic acid    ribitol
## stearic acid            1.0000000 0.2083188     0.2492042 0.15060595
## heptadecanoic acid      0.8621075 0.1501909     0.3128487 0.09976116
##                      pseudo uridine nicotinamide myo-inositol   mannose
## stearic acid            -0.19706050    0.3878188   0.04917488 0.1807085
## heptadecanoic acid      -0.05231672    0.2501691   0.13140503 0.2167668
##                         lyxitol heptadecanoic acid glutaric acid glutamic acid
## stearic acid       0.02011295          0.8621075  -0.034593242    0.05591120
## heptadecanoic acid 0.01543889          1.0000000  -0.005699254   -0.04747537
##                      citric acid behenic acid alpha-ketoglutarate
## stearic acid          0.08475043    0.4184708          0.02048958
## heptadecanoic acid  0.17991391    0.3845567         -0.01077332
```

##*Linearity between the two arms* The below code looks at whether the data is linearly separable between the two arms. Values from each measured parameters are plotted on y and the arm is plotted on x.

```
# for (v in data[,c(-1,-2)]){
#  plot(as.numeric(v), col = as.factor(data$Arm))
# }
```

Based on the plots generated by the above code, the values do not show a linear association between the two arms , therefore, when applying SVM, although the model performs well on the training data, on the test data, there is a significant drop in the model performance. See this in the subsequent sections.

##*Creating training and testing datasets*

```
# Creating a training and test set from the above dataframe df:
# install.packages("caTools")
# install.packages("e1071")
library(caTools)
library(e1071)
index <- sample.split(data$Arm, SplitRatio = .7)
tr_set <- subset(data, index == TRUE)
final_val_set <- subset(data, index == FALSE)
# Splitting the training set into further training and test sets:
index_train <- sample.split(tr_set$Arm, SplitRatio = .5)
tr_set_train <- subset(tr_set, index_train == TRUE)
tr_set_test <- subset(tr_set, index_train == FALSE)
```

#**MODEL FITTING** ##*Support Vector Machine* Now, that I have the train_set, I will train the SVM on the training set. Here I train the SVM to predict the Arm, based on the most significant variables from the df dataset; these are stearic acid, nicotinamide, heptadecanoic acid and behenic acid.

```
# SVM model with linear kernel
svm_model_linear <- svm(as.factor(Arm)~
                          as.numeric(tr_set_train$`stearic acid`) +
                          as.numeric(tr_set_train$nicotinamide) +
                          as.numeric(tr_set_train$`heptadecanoic acid`) +
                          as.numeric(tr_set_train$`behenic acid`),
                        data = tr_set_train, method = "C", kernel = "linear", gamma = 1,
                        cost = 1)
# Getting the mean of the correctly predicted arm on the train data set:
predict_tr_linear <- predict(svm_model_linear, tr_set_train)
mean_linear_train <- mean(predict_tr_linear == as.factor(tr_set_train$Arm))
mean_linear_train
```

```
## [1] 0.7647059
```

```
# Getting the mean of the correctly predicted arm on the test data set:
predict_test_linear <- predict(svm_model_linear, tr_set_test)
mean_linear_test <- mean(predict_test_linear == tr_set_test$Arm)
mean_linear_test
```

```
## [1] 0.7647059
```

```
# SVM model with radial kernel:
svm_model_radial <- svm(as.factor(Arm)~
                          as.numeric(tr_set_train$`stearic acid`) +
                          as.numeric(tr_set_train$nicotinamide) +
                          as.numeric(tr_set_train$`heptadecanoic acid`) +
                          as.numeric(tr_set_train$`behenic acid`),
                 data = tr_set_train, method = "C-classification", kernel = "radial", gamma = 1,
                 cost = 1)
# Getting the mean of the correctly predicted arm on the train data set:
predict_tr_radial <- predict(svm_model_radial, tr_set_train)
mean_radial_train <- mean(predict_tr_radial == tr_set_train$Arm)
mean_radial_train
```

```
## [1] 0.8823529
```

```
# Getting the mean of the correctly predicted arm on the test data set:
predict_test_radial <- predict(svm_model_radial, tr_set_test)
mean_radial_test <- mean(predict_test_radial == tr_set_test$Arm)
mean_radial_test
```

```
## [1] 0.8823529
```

#**VALIDATION** ##*SVM* Validating the SVM model (linear and radial)

```
# Linear
predict_valid_linear <- predict(svm_model_linear, final_val_set)
mean_linear_valid <- mean(predict_valid_linear == final_val_set$Arm)
```

```
## Warning in `==.default`(predict_valid_linear, final_val_set$Arm): longer object
## length is not a multiple of shorter object length
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
mean_linear_valid
```

```
## [1] 0.7647059
```

```
# Radial
predict_valid_radial <- predict(svm_model_radial, final_val_set)
mean_radial_valid <- mean(predict_valid_radial == final_val_set$Arm)
```

```
## Warning in `==.default`(predict_valid_radial, final_val_set$Arm): longer object
## length is not a multiple of shorter object length

## Warning in `==.default`(predict_valid_radial, final_val_set$Arm): longer object
## length is not a multiple of shorter object length
```

```
mean_radial_valid
```

```
## [1] 0.7647059
```

#RESULTS AND CONCLUSION

```
model_results_tibble <- tibble(Models = c("SVM Linear","SVM Radial"),
                               TrainFit = c(mean_linear_train, mean_radial_train),
                               TestFit = c(mean_linear_test, mean_radial_test),
                               Validation = c(mean_linear_valid, mean_radial_valid)) %>%
  mutate(TestFit = sprintf("%0.4f", TestFit))

model_results_tibble
```

```
## # A tibble: 2 x 4
##   Models     TrainFit TestFit Validation
##   <chr>         <dbl> <chr>        <dbl>
## 1 SVM Linear    0.765 0.7647       0.765
## 2 SVM Radial    0.882 0.8824       0.765
```

#FUTURE PERSPECTIVES