

Assignment 2

Simona_Cernat

2023-01-15

INTRODUCTION

The data is part of a study which investigates the brain (cortex) protein expression pattern of several mice in various conditions. In this data set the conditions are represented by the variables: Genotype, Behaviour, Treatment and class (combination of all three). The first distinction between mice was the Genotype: “control” healthy mice were compared to the “trisomic mice” which constitute the animal models for the Down Syndrome (characterized by impaired learning abilities). They were further selected based on the Behavior feature: mice were stimulated to learn a task - CS (context shock) while others were not - SC. Additionally, about half of the mice were given memantine, a drug which promotes learning recovery, or a saline solution, which acts as control and has no effect. The mice were divided in 8 groups/ classes:

Classes:

- c-CS-s: control mice, stimulated to learn, injected with saline
- c-CS-m: control mice, stimulated to learn, injected with memantine
- c-SC-s: control mice, not stimulated to learn, injected with saline
- c-SC-m: control mice, not stimulated to learn, injected with memantine
- t-CS-s: trisomy mice, stimulated to learn, injected with saline
- t-CS-m: trisomy mice, stimulated to learn, injected with memantine
- t-SC-s: trisomy mice, not stimulated to learn, injected with saline
- t-SC-m: trisomy mice, not stimulated to learn, injected with memantine
- 38 control mice and 34 trisomic

The protein expression is measured for 77 proteins. There are 15 measurements for each protein/mouse combination. The aim of the study was to identify the proteins that are relevant and significantly different for the classes.

Data cleaning

```
# Load the packages needed
library(tidyverse)
library( readxl )
library(patchwork)
library(ggdendro)
```

```
library(fpc)
library(mclust)
library(dendextend)
library(fossil)
library(cluster)
```

```
# Load the dataset
set.seed(227)
ds1 <- read_excel("./raw_data/Data_Cortex_Nuclear.xls")

# Transform the dataset into a tibble and transform into factors where appropriate

ds1 <- as_tibble(ds1)
ds1 <- ds1 %>%
  mutate(Genotype = as.factor(Genotype)) %>%
  mutate(Treatment = as.factor(Treatment)) %>%
  mutate(Behavior = as.factor(Behavior)) %>%
  mutate(class = as.factor(class))
```

a) Missing values

First the dataset will be checked for missing values.

```
# Looking at the missing values
sapply(ds1, function(x) sum(is.na(x)))
```

##	MouseID	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N
##	0	3	3	3	3
##	NR2A_N	pAKT_N	pBRAF_N	pCAMKII_N	pCREB_N
##	3	3	3	3	3
##	pELK_N	pERK_N	pJNK_N	PKCA_N	pMEK_N
##	3	3	3	3	3
##	pNR1_N	pNR2A_N	pNR2B_N	pPKCAB_N	pRSK_N
##	3	3	3	3	3
##	AKT_N	BRAF_N	CAMKII_N	CREB_N	ELK_N
##	3	3	3	3	18
##	ERK_N	GSK3B_N	JNK_N	MEK_N	TRKA_N
##	3	3	3	7	3
##	RSK_N	APP_N	Bcatenin_N	SOD1_N	MTOR_N
##	3	3	18	3	3
##	P38_N	pMTOR_N	DSCR1_N	AMPKA_N	NR2B_N
##	3	3	3	3	3
##	pNUMB_N	RAPTOR_N	TIAM1_N	pP70S6_N	NUMB_N
##	3	3	3	3	0
##	P70S6_N	pGSK3B_N	pPKCG_N	CDK5_N	S6_N
##	0	0	0	0	0
##	ADARB1_N	AcetylH3K9_N	RRP1_N	BAX_N	ARC_N
##	0	0	0	0	0
##	ERBB4_N	nNOS_N	Tau_N	GFAP_N	GluR3_N
##	0	0	0	0	0
##	GluR4_N	IL1B_N	P3525_N	pCASP9_N	PSD95_N
##	0	0	0	0	0

```
##          SNCA_N      Ubiquitin_N pGSK3B_Tyr216_N          SHH_N          BAD_N
##          0          0          0          0          213
##          BCL2_N      pS6_N      pCFOS_N          SYP_N      H3AcK18_N
##          285          0          75          0          180
##          EGR1_N      H3MeK4_N      CaNA_N      Genotype      Treatment
##          210          270          0          0          0
##          Behavior      class
##          0          0
```

```
#Tidyverse version
```

```
#ds1 %>%
  #map(is.na) %>%
  #map(sum) %>%
  #bind_cols()
```

```
#All the rows that have missing values
```

```
ds1 %>% filter_all(any_vars(is.na(.))) %>% nrow()
```

```
## [1] 528
```

```
#approx 530 of them have a missing value
```

Aprox 530/1080 of mice/rows have at least 1 missing data. Given the high number of NAs, imputing the data seems like a better strategy than deleting the rows that have a missing value.

Next, data imputation will be performed by converting the NAs to the mean of the selected columns with respect to the individual condition/class.

```
#Table with the means per protein per individual class
```

```
Means <- ds1 %>%
  filter(complete.cases(.)) %>%
  group_by(class) %>%
  summarise_all(mean) %>%
  select(-MouseID, -Treatment, -Behavior)
```

```
#Little bit of code for data imputation
```

```
imputed_ds = ds1
```

```
#iterates through row and column, checks for missing values and if the missing value is found is replaced
```

```
for (i in 1:nrow(imputed_ds)) {
  for (col in names(imputed_ds)) {
    if (!(col %in% c("MouseID", "Treatment", "Behavior", "class"))) && (is.na(imputed_ds[i, col])) {
      imputed_ds[i, col] = (Means %>% filter(class == imputed_ds[i, "class"]$class))[[col]]
    }
  }
}
```

```
imputed_ds %>% filter_all(any_vars(is.na(.))) %>% nrow()
```

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

```
ds1 %>% filter_all(any_vars(is.na(.))) %>% nrow()
```

```
## [1] 528
```

b) Normalization

As a last step data cleaning, the data set will be normalized. While the data is on the same scale of measurement, for some clustering methods, such as k-means clustering, the results might be heavily influenced by greater values. To perform well, k-means clustering assumes equal variances because the algorithm tends to create “round” clusters. If the variance is left unequal then clusters will separate more around the variables with greater variance.

For example in the summary below it can be seen that some protein values vary more.

```
imputed_ds %>% summarise_if(is.numeric, var)
```

```
## # A tibble: 1 x 77
##   DYRK1A_N ITSN1_N BDNF_N NR1_N NR2A_N pAKT_N pBRAF_N pCAMKI-1 pCREB_N pELK_N
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  0.0620 0.0631 0.00243 0.120 0.869 0.00173 0.000729 1.67 0.00106 0.218
## # ... with 67 more variables: pERK_N <dbl>, pJNK_N <dbl>, PKCA_N <dbl>,
## #   pMEK_N <dbl>, pNR1_N <dbl>, pNR2A_N <dbl>, pNR2B_N <dbl>, pPKCAB_N <dbl>,
## #   pRSK_N <dbl>, AKT_N <dbl>, BRAF_N <dbl>, CAMKII_N <dbl>, CREB_N <dbl>,
## #   ELK_N <dbl>, ERK_N <dbl>, GSK3B_N <dbl>, JNK_N <dbl>, MEK_N <dbl>,
## #   TRKA_N <dbl>, RSK_N <dbl>, APP_N <dbl>, Bcatenin_N <dbl>, SOD1_N <dbl>,
## #   MTOR_N <dbl>, P38_N <dbl>, pMTOR_N <dbl>, DSCR1_N <dbl>, AMPKA_N <dbl>,
## #   NR2B_N <dbl>, pNUMB_N <dbl>, RAPTOR_N <dbl>, TIAM1_N <dbl>, ...
```

Below I will normalize the data using z-score normalization which will bring all variances to 1.

```
# z-score normalization
imputed_nr <- select_if(imputed_ds, is.numeric)
imputed_chr <- select_if(imputed_ds, is.factor)
s_imputed_ds <- scale(imputed_nr)

ds_normalized <- cbind(imputed_chr, s_imputed_ds)

rm(imputed_nr, imputed_chr)
```

```
#There are two data sets that I will use further. The normalized and the un-normalized one
#saveRDS(ds_normalized, file = "ds_normalized.rds")
#saveRDS(imputed_ds, file = "imputed_ds.rds")
```

```
#Here the two data sets can be re-read
```

```
ds_normalized <- readRDS("./processed_data/ds_normalized.rds")
imputed_ds <- readRDS("./processed_data/imputed_ds.rds")
```

Clustering

```
norm_pca <- prcomp(ds_normalized %>% select(-Treatment, -class, -Behavior, -Genotype), center = TRUE, s
summary(norm_pca)
```

```
## Importance of components:
##
##          PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation  4.4177 3.6522 2.8158 2.40167 1.94997 1.88035 1.65825
## Proportion of Variance 0.2535 0.1732 0.1030 0.07491 0.04938 0.04592 0.03571
## Cumulative Proportion 0.2535 0.4267 0.5296 0.60456 0.65394 0.69986 0.73557
##
##          PC8      PC9      PC10      PC11      PC12      PC13      PC14
## Standard deviation  1.53575 1.35104 1.14327 1.08301 0.97777 0.93189 0.9035
## Proportion of Variance 0.03063 0.02371 0.01697 0.01523 0.01242 0.01128 0.0106
## Cumulative Proportion 0.76620 0.78991 0.80688 0.82211 0.83453 0.84581 0.8564
##
##          PC15      PC16      PC17      PC18      PC19      PC20      PC21
## Standard deviation  0.87345 0.82978 0.77262 0.75928 0.70128 0.67523 0.65193
## Proportion of Variance 0.00991 0.00894 0.00775 0.00749 0.00639 0.00592 0.00552
## Cumulative Proportion 0.86632 0.87526 0.88301 0.89050 0.89689 0.90281 0.90833
##
##          PC22      PC23      PC24      PC25      PC26      PC27      PC28
## Standard deviation  0.6505 0.63821 0.6082 0.60437 0.58754 0.56405 0.55806
## Proportion of Variance 0.0055 0.00529 0.0048 0.00474 0.00448 0.00413 0.00404
## Cumulative Proportion 0.9138 0.91911 0.9239 0.92866 0.93314 0.93727 0.94132
##
##          PC29      PC30      PC31      PC32      PC33      PC34      PC35
## Standard deviation  0.51740 0.50030 0.48150 0.47679 0.45840 0.44889 0.41790
## Proportion of Variance 0.00348 0.00325 0.00301 0.00295 0.00273 0.00262 0.00227
## Cumulative Proportion 0.94480 0.94805 0.95106 0.95401 0.95674 0.95936 0.96162
##
##          PC36      PC37      PC38      PC39      PC40      PC41      PC42
## Standard deviation  0.40535 0.40136 0.3920 0.38903 0.37706 0.35846 0.35726
## Proportion of Variance 0.00213 0.00209 0.0020 0.00197 0.00185 0.00167 0.00166
## Cumulative Proportion 0.96376 0.96585 0.9678 0.96981 0.97166 0.97333 0.97498
##
##          PC43      PC44      PC45      PC46      PC47      PC48      PC49
## Standard deviation  0.34286 0.33883 0.32924 0.32138 0.31419 0.30541 0.30235
## Proportion of Variance 0.00153 0.00149 0.00141 0.00134 0.00128 0.00121 0.00119
## Cumulative Proportion 0.97651 0.97800 0.97941 0.98075 0.98203 0.98324 0.98443
##
##          PC50      PC51      PC52      PC53      PC54      PC55      PC56
## Standard deviation  0.29577 0.28260 0.27962 0.27404 0.26707 0.25650 0.24470
## Proportion of Variance 0.00114 0.00104 0.00102 0.00098 0.00093 0.00085 0.00078
## Cumulative Proportion 0.98557 0.98660 0.98762 0.98859 0.98952 0.99038 0.99115
##
##          PC57      PC58      PC59      PC60      PC61      PC62      PC63
## Standard deviation  0.23616 0.23133 0.22832 0.22460 0.22025 0.21719 0.20570
## Proportion of Variance 0.00072 0.00069 0.00068 0.00066 0.00063 0.00061 0.00055
## Cumulative Proportion 0.99188 0.99257 0.99325 0.99390 0.99453 0.99515 0.99570
##
##          PC64      PC65      PC66      PC67      PC68      PC69      PC70
## Standard deviation  0.1963 0.19227 0.19074 0.18564 0.1759 0.16648 0.15767
## Proportion of Variance 0.0005 0.00048 0.00047 0.00045 0.0004 0.00036 0.00032
## Cumulative Proportion 0.9962 0.99668 0.99715 0.99760 0.9980 0.99836 0.99868
##
##          PC71      PC72      PC73      PC74      PC75      PC76      PC77
## Standard deviation  0.15415 0.14106 0.12838 0.1240 0.12198 0.10534 1.962e-15
## Proportion of Variance 0.00031 0.00026 0.00021 0.0002 0.00019 0.00014 0.000e+00
## Cumulative Proportion 0.99899 0.99925 0.99946 0.9997 0.99986 1.00000 1.000e+00
```

```

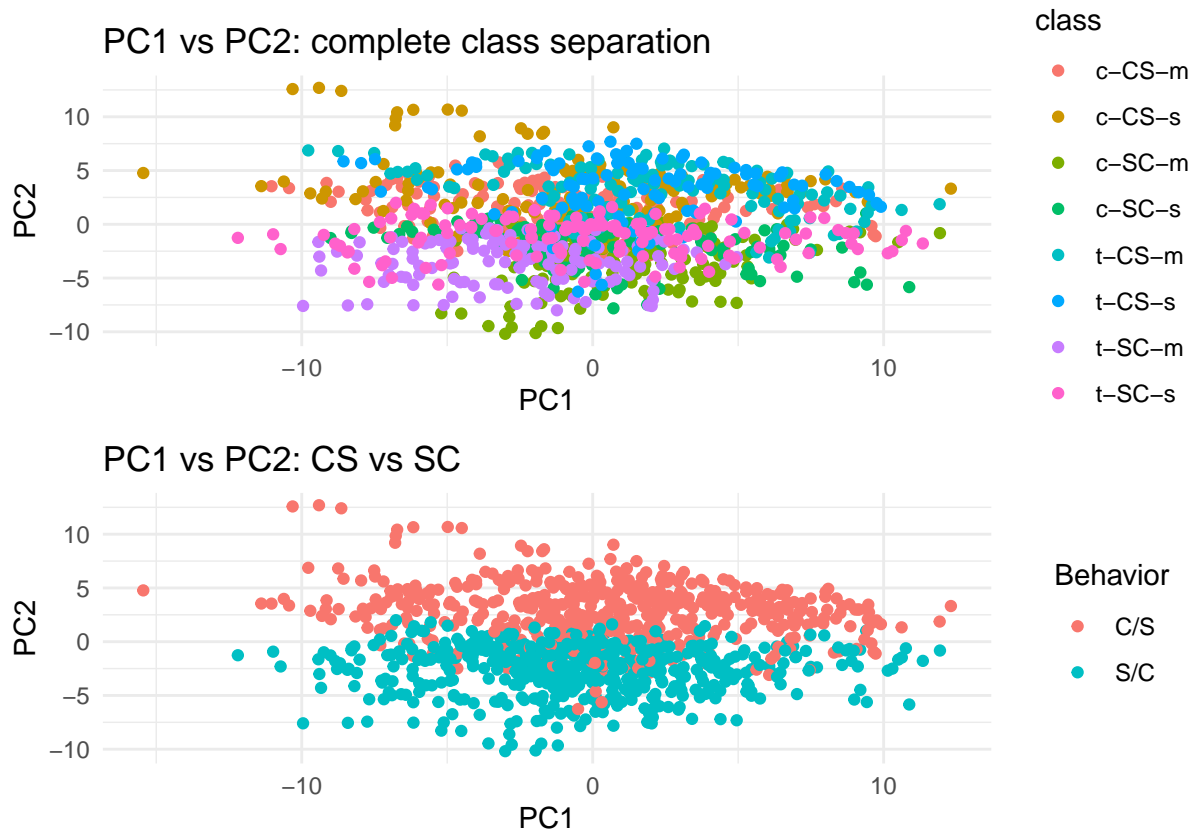
# Add the class labels to the PCA results
ds_normalized$PC1 <- norm_pca$x[,1]
ds_normalized$PC2 <- norm_pca$x[,2]

# Create a scatter plot of PC1 vs PC2 colored by class
pcaplot1 <- ggplot(ds_normalized, aes(x = PC1, y = PC2, color = class)) +
  theme_minimal() +
  ggtitle("PC1 vs PC2: complete class separation") +
  geom_point()

pcaplot2 <- ggplot(ds_normalized, aes(x = PC1, y = PC2, color = Behavior)) +
  theme_minimal() +
  geom_point() +
  ggtitle("PC1 vs PC2: CS vs SC")

pcaplot1 + pcaplot2 +
  plot_layout(ncol = 1, nrow = 2)

```



There is not a clear separation between classes, on the contrary, they quite overlap in the middle. It is noticeable that there is a division between the CS and SC groups, meaning the state of learning vs not-learning produced the most differences in gene expression. Of course, this is just plotting on two dimensions. Additionally, these are the results of PCA but other dimensionality reduction methods such as t-SNE might give better results.

Usually a clear separation between clusters is desired but here from a biological perspective we expect some

overlap. We expect mice that behave similarly to have similar protein expression patterns. The study presents the trisomic (naturally learning disability groups) and the control mice. The trisomic CS mice injected with saline did not learn the task but the group injected with the memantine did. The control CS mice learn equally well regardless of the injection. Therefore we expect t-CS-s vs t-CS-m to be different but the c-CS-s vs c-CS-m to be similar. The former cannot be noticed here. The differences between CS vs SC produce too much noise to be able to distinguish other dissimilarities.

A. k-means clustering

I will use the PCs that explain at least 90% of the variation.

```
#K means clustering

#The first 21 PCs add up to aprox 90% of the variation
norm_pca21 <- norm_pca$x[, 1:21]
norm_pca21 <- as_tibble(norm_pca21)

model_k8 <- kmeans(norm_pca21, center = 8)
model_k2 <- kmeans(norm_pca21, center = 2)

plot_k8 <- norm_pca21 %>% mutate(clu = as_factor(model_k8$cluster)) %>%
  ggplot(mapping = aes(x = PC1, y = PC2)) +
  geom_point(aes(color = clu)) +
  theme_minimal() +
  scale_fill_viridis_d(guide = "none")+
  ggtitle("k-means clustering - 8 classes")

plot_k2 <- norm_pca21 %>% mutate(clu = as_factor(model_k2$cluster)) %>%
  ggplot(mapping = aes(x = PC1, y = PC2)) +
  geom_point(aes(color = clu)) +
  theme_minimal() +
  scale_fill_viridis_d(guide = "none") +
  ggtitle("k-means clustering - 2 classes")

plot_k8 + plot_k2 +
  plot_layout(ncol = 1, nrow = 2)
```



```
#K-means - Stability test
boot_test <- clusterboot(data = norm_pca21, B = 1000, clustermethod = kmeansCBI, k = 8,
  count = FALSE)

boot_test2 <- clusterboot(data = norm_pca21, B = 1000, clustermethod = kmeansCBI, k = 2,
  count = FALSE)

boot_test$bootmean
```

```
## [1] 0.6817269 0.4275436 0.6467505 0.5054740 0.6387589 0.4598820 0.7667898
## [8] 0.5614647
```

```
boot_test2$bootmean
```

```
## [1] 0.9361517 0.9493407
```

8 clusters: It can be noticed that the clusters look a little bit more well-distinguishable but different from the PCA plot above. By consulting the stability measurement, the stability of the clusters seems low. In other words, the clusters are to a certain degree randomly assigned. However, this is to be expected as there does not seem to be a clear separation between the 8 classes.

However it is noticeable that the data is more readily identified by the CS/SC groups. When testing whether the 2 clusters can be captured by the k-means algorithm, we notice that this does not happen and the wrong clusters are assigned respective to the actual groups

From now on, I will drop trying to recreate the 8 classes, instead I will only use the two CS/SC separation.

I will next calculate some metrics that assess the quality of the clusters. **In an unsupervised project usually the true labels are not known. However, given that the true labels are given I will employ them to test the external validity of the clusters. This would be my primary goal from now on.** Therefore I will use the Rand Index (ARI) to compare the similarity between the true labels and the predicted classes/ clusters. It ranges from 1 to -1, with 1 being a perfect match.

```
#K-means - Rand Index
```

```
adjustedRandIndex(as.vector(imputed_ds %>% mutate(Behavior = ifelse(Behavior == "C/S", 1, 2)) %>% select(
```

```
## [1] 0.03005498
```

Confirm what had been observed above. The two clusters do not reproduce the cs/sc classes.

Part B - Hierarchical correlation-based clustering

Next, I will focus my attention on hierarchical clustering. My opinion is that this type of clustering will render better results compared to k-means. In k-means, the Euclidean distance is used. I believe a better metric for expression data is correlation between the samples/mice.

Note: for this I will use the unnormalized data set. That is because this type of clustering is free from the assumptions of the k-means clustering. Additionally, I will not use the principal components to compute the distances as this would probably be wrong. We are interested in how samples' protein expression correlate with each other.

```
#Hierarchical clustering -Correlation
```

```
#calculate the correlation matrix between each sample/mice (per row)
cor <- imputed_ds %>%
  select(-class, - Genotype, -Behavior, -Treatment, -MouseID) %>%
  #use the transpose to calculate the correlation per row and not column
  t() %>%
  cor(method = "spearman" , use="pairwise.complete.obs")

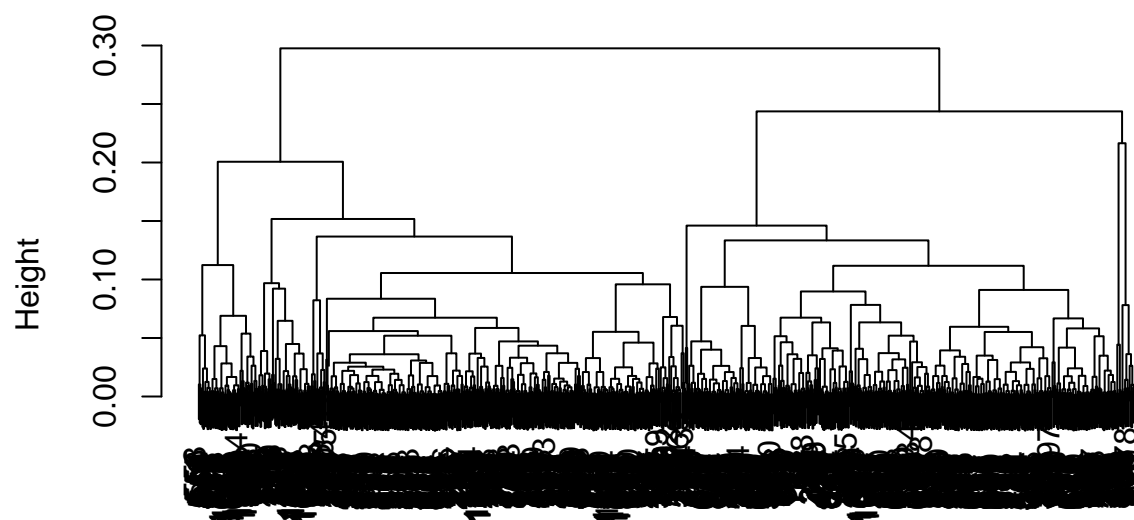
#tried a measure of dissimilarity instead of similarity
dist_sperman <- as.dist(1 - cor)

sperman_tree <- hclust(dist_sperman, method="complete")

clus_2 <- as_factor(cutree(sperman_tree, 2))

plot(sperman_tree)
```

Cluster Dendrogram

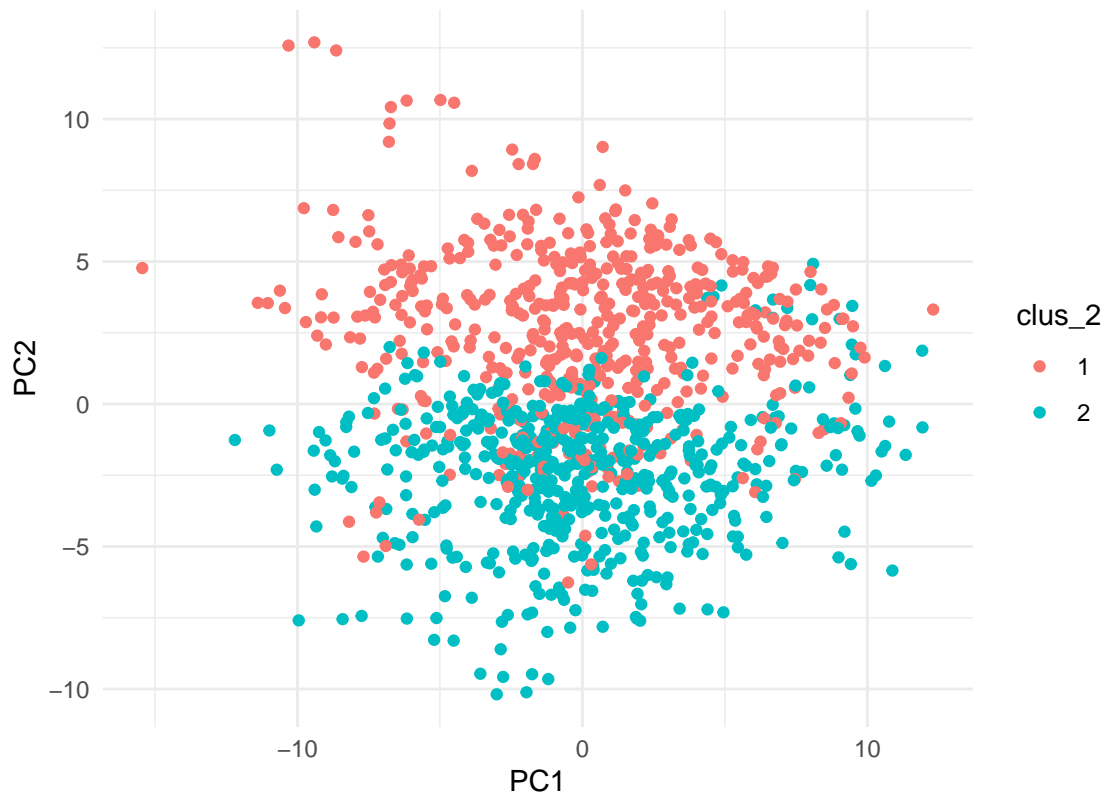


```
dist_sperman
hclust (*, "complete")
```

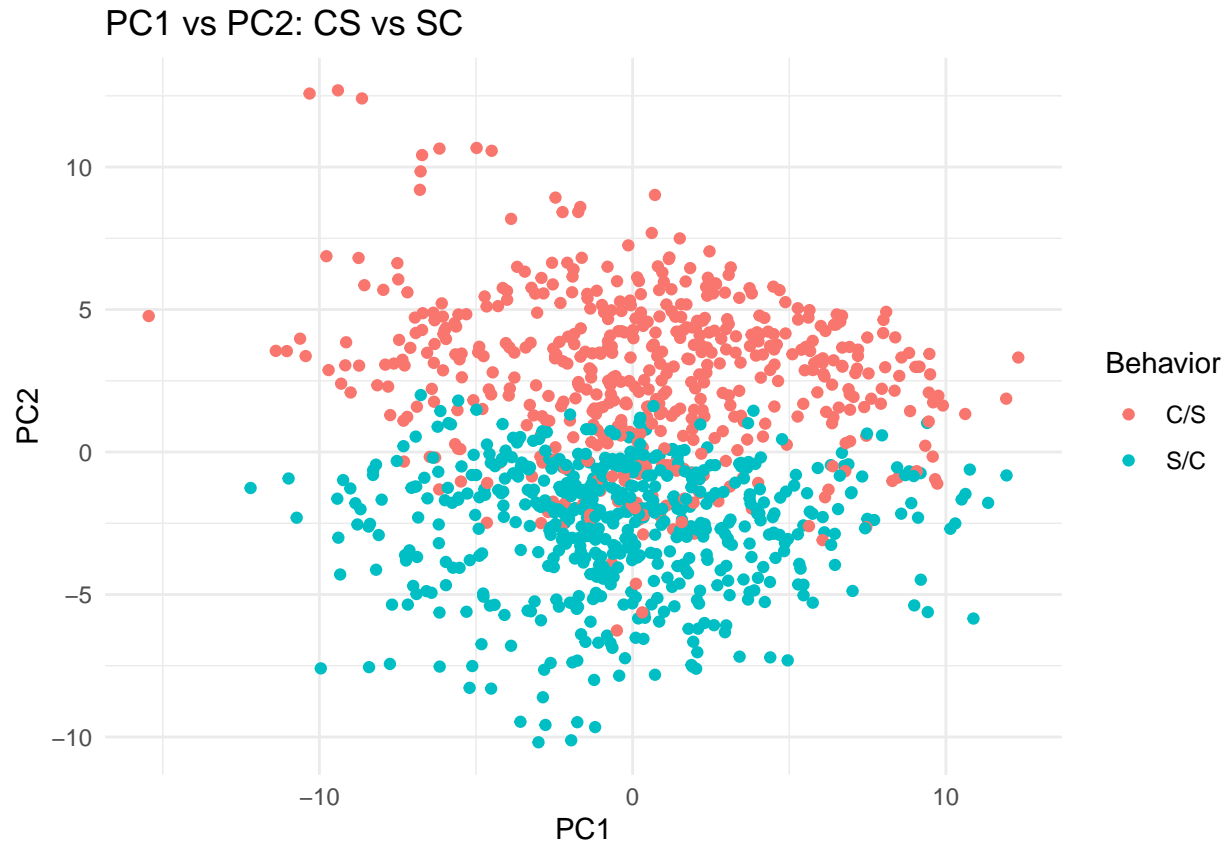
```
#Visualising the clusters on the PC plot
p2_spe1 <- ds_normalized %>%
  ggplot(aes(x = PC1, y = PC2, colour = clus_2)) +
  scale_fill_brewer(palette = 1) +
  geom_point() +
  ggtitle("Correlation-based hierachical clustering - 2 clusters") +
  theme(legend.position = "n") +
  theme_minimal() +
  coord_fixed()

p2_spe1
```

Correlation-based hierachical clustering – 2 clusters



```
#For comparison with the true CS/SC classes  
pcaplot2
```



The clustering looks very similar to the true classes on the PCA plot.

```
#Assessment of the clusters
#Rand index
```

```
adjustedRandIndex(as.vector(imputed_ds %>% mutate(Behavior = ifelse(Behavior == "C/S", 1, 2)) %>% select(class, Behavior))
```

```
## [1] 0.8883775
```

Given that 1 is the maximum obtainable score, a almost 0.9 is pretty good, meaning that the clusters are representative of the CS/SC classes.

Another metric I would like to use is the silhouette score, which measures the similarity of an observation to its own cluster compared to other clusters. It takes values between 0 and 1, with a higher value indicating the better score.

```
sil_samples <- silhouette(as.integer(clus_2), dist(imputed_ds %>% select(-class, - Genotype, -Behavior, -Behavior_2)))
mean(sil_samples[, 3])
```

```
## [1] 0.1414123
```

This score is quite low but overall not too bad (they are mostly dissimilar to each other) and could be due to the fact that the clusters overlap to a certain degree in the middle, despite the clear separation at the periphery.

Discussion

In this assignment I used clustering methods in order to find patterns in the data. However, I did not focus on finding the relevant proteins that discriminate between classes, as I believed this is beyond the scope of this assignment. I could have chosen just the proteins/ features which showed the most variance and discard the 'stable' ones for this analysis, as this strategy would probably minimize the noise. I chose not to do this and just try the best I can to discover the patterns in the data. This data set had a high number of features (~80). To reduce the number of features, I used PCA. It became intuitive that a clear distinction between the 8 classes/ conditions is almost impossible. However, the CS vs SC groups can be well separated to a certain degree.

I chose to perform k-means clustering using the first 21 principal components, which explained 90% of the variation. As expected, the 8 classes cannot be well separated. Then I looked more into the CS vs SC separation, which in best case scenario would render the same outputs as the PCA plots. While in most unsupervised learning projects the true labels are not known, in this assignment I focused on choosing a clustering method that achieves the most accurate results in the 2 cluster scenario as opposed to the best separability. For the two clusters scenario, it is noticeable by looking at the plot and by consulting the Rand Index that the two clusters selected by k-means are distinct to the actual CS/SC classes. Additionally, the bootstrapped stability score is pretty low, meaning that the clusters are not even stable across multiple iterations. k-means clustering assumes spherical clusters but this might not be the case with these data.

Then correlation-based hierarchical clustering was performed. Intuitively, this would suit the data best. From a biological point of view, proteins in a certain state/tissue tend to co-express together. Indeed this seemed to be the case. 2 clusters that could correspond to CS/SC can be seen. The Rand Index shows a good similarity score compared to the actual classes. The Silhouette score was further used to assess how dissimilar the two clusters are. The score tells that there is some overlap, which can be also confirmed visually.

In this assignment, the focus was on clustering methods that rendered output close to the true labels of the data (external validity), in a semi-supervised fashion. To test the similarity I have used the rand index and visualization by plotting. Each clustering method was further assessed for different metrics and qualities. For this type of data and question, correlation-based hierarchical clustering rendered the most similar results to the true classes while the k-means clustering strategy failed to do so.