Mice Protein Expression

Dani Beltrán

11 de febrero de 2019

Introduction

The goal of this project is to analyze the data and predict mice genotype, behavior and treatment from the protein expression. It is also determine, in average, how many proteins are necessary to make an accurate prediction. Random forest and Monte Carlo simulations are performed in order to achieve these objectives.

Data can be loaded from the kaggle web page (https://www.kaggle.com/ruslankl/mice-protein-expression). The data set contains 1080 observations with 82 columns: A unique MouseID, the mice genotype (Ts65Dn or Control), behavior (C/S or S/C) and treatment (Memantine or Saline), a summarizing column of the last 3 parameters called "class" and 77 proteins expressions.

This project is contextualized in the Data Science edX course. Specifically, it is the submission for the Capstone Project: IDV Learners section.

Methods

Packages and data loading

Packages are loaded

```
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(caret)) install.packages("caret")
if(!require(gtools)) install.packages("gtools")
```

Data is loaded from the desktop

```
library(tidyverse)
ds <- read_csv("C:/Users/Desktop/Data_Cortex_Nuclear.csv")
```

Data exploration and visualization

First of all, the no protein columns are analyzed.

It is verified that there are no duplicated MouseIDs

```
mean(duplicated(ds$MouseID))
```

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

The unique values of the other columns and their quantities are also displayed. No NAs are found.

```
table(ds$Genotype)
```

```
##
## Control  Ts65Dn
##     570     510
```

```
table(ds$Behavior)
```

```
##
## C/S S/C
## 525 555
```

```
table(ds$Treatment)
```

```
##
## Memantine    Saline
##       570       510
```

```
table(ds$class)
```

```
##
## c-CS-m c-CS-s c-SC-m c-SC-s t-CS-m t-CS-s t-SC-m t-SC-s
##    150    135    150    135    135    105    135    135
```

As shown, there are 2 genotypes, 2 treatments, 2 behaviors and 8 classes. The class column is a summary of de 3 previous columns. Data is analyzed in order to check for contradictions between these columns.

The genotype values (Control and Ts65Dn) are found in the class as "c-x-x"" and "t-x-x". It is verified.

```
table(ds %>% filter(Genotype == "Control") %>% select(class))
```

```
##
## c-CS-m c-CS-s c-SC-m c-SC-s
##    150    135    150    135
```

```
table(ds %>% filter(Genotype == "Ts65Dn") %>% select(class))
```

```
##
## t-CS-m t-CS-s t-SC-m t-SC-s
##    135    105    135    135
```

The behavior values (C/S and S/C) are found in the class as "x-CS-x"" and "x-SC-x". It is verified.

```
table(ds %>% filter(Behavior == "C/S") %>% select(class))
```

```
##
## c-CS-m c-CS-s t-CS-m t-CS-s
##    150    135    135    105
```

```
table(ds %>% filter(Behavior == "S/C") %>% select(class))
```

```
##
## c-SC-m c-SC-s t-SC-m t-SC-s
##    150    135    135    135
```

The treatment values (Memantine and Saline) are found in the class as "x-x-m" and "x-x-s". It is verified.

```
table(ds %>% filter(Treatment == "Memantine") %>% select(class))
```

```
##
## c-CS-m c-SC-m t-CS-m t-SC-m
##    150    150    135    135
```

```
table(ds %>% filter(Treatment == "Saline") %>% select(class))
```

```
##
## c-CS-s c-SC-s t-CS-s t-SC-s
##    135    135    105    135
```

No contradictions are found.

Next, protein columns are analyzed. NAs are quantified in order to know how much data is missing.

```
table(round(unname((unlist(ds %>%
  select(-MouseID,-Genotype,-Behavior,-Treatment,-class) %>%
  transmute_all(is.na) %>%
  summarise_all(mean) ))),4))
```

```
##
##      0 0.0028 0.0065 0.0167 0.0694 0.1667 0.1944 0.1972   0.25 0.2639
##     28     40      1      2      1      1      1      1      1      1
```

There are many columns with exactly the same number of NAs (a very small proportion), many columns with no NAs and just a few with other proportions. Some of them are missing a quarter of the observations. The proportion of NAs in now displayed next to the name of each protein. Proteins with the biggest lack of information are pCFOS_N (6.94%), H3AcK18_N (16.7%), EGR1_N(19.4%), BAD_N (19.7%), H3MeK4_N (25%) and BCL2_N (26.4%). For this reason, these proteins may be problematic predictors.

```
options(dplyr.width = Inf)
ds %>%
  select(-MouseID,-Genotype,-Behavior,-Treatment,-class) %>%
  transmute_all(is.na) %>%
  summarise_all(mean)
```

```
## # A tibble: 1 x 77
##   DYRK1A_N ITSN1_N  BDNF_N   NR1_N  NR2A_N  pAKT_N pBRAF_N pCAMKII_N
##      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>     <dbl>
## 1  0.00278 0.00278 0.00278 0.00278 0.00278 0.00278 0.00278   0.00278
##   pCREB_N  pELK_N  pERK_N  pJNK_N  PKCA_N  pMEK_N  pNR1_N pNR2A_N pNR2B_N
##     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 0.00278 0.00278 0.00278 0.00278 0.00278 0.00278 0.00278 0.00278 0.00278
##   pPKCAB_N  pRSK_N   AKT_N  BRAF_N CAMKII_N  CREB_N  ELK_N   ERK_N GSK3B_N
##      <dbl>   <dbl>   <dbl>   <dbl>    <dbl>   <dbl>  <dbl>   <dbl>   <dbl>
## 1  0.00278 0.00278 0.00278 0.00278  0.00278 0.00278 0.0167 0.00278 0.00278
##     JNK_N   MEK_N  TRKA_N   RSK_N   APP_N Bcatenin_N  SOD1_N  MTOR_N
##     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>      <dbl>   <dbl>   <dbl>
## 1 0.00278 0.00648 0.00278 0.00278 0.00278     0.0167 0.00278 0.00278
##     P38_N pMTOR_N DSCR1_N AMPKA_N  NR2B_N pNUMB_N RAPTOR_N TIAM1_N pP70S6_N
##     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>    <dbl>   <dbl>    <dbl>
## 1 0.00278 0.00278 0.00278 0.00278 0.00278 0.00278  0.00278 0.00278  0.00278
##   NUMB_N P70S6_N pGSK3B_N pPKCG_N CDK5_N  S6_N ADARB1_N AcetylH3K9_N RRP1_N
##    <dbl>   <dbl>    <dbl>   <dbl>  <dbl> <dbl>    <dbl>        <dbl>  <dbl>
## 1      0       0        0       0      0     0        0            0      0
##   BAX_N ARC_N ERBB4_N nNOS_N Tau_N GFAP_N GluR3_N GluR4_N IL1B_N P3525_N
##   <dbl> <dbl>   <dbl>  <dbl> <dbl>  <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
## 1     0     0       0      0     0      0       0       0      0       0
##   pCASP9_N PSD95_N SNCA_N Ubiquitin_N pGSK3B_Tyr216_N SHH_N BAD_N BCL2_N
##      <dbl>   <dbl>  <dbl>       <dbl>           <dbl> <dbl> <dbl>  <dbl>
## 1        0       0      0           0               0     0 0.197  0.264
##   pS6_N pCFOS_N SYP_N H3AcK18_N EGR1_N H3MeK4_N CaNA_N
##   <dbl>   <dbl> <dbl>     <dbl>  <dbl>    <dbl>  <dbl>
## 1     0  0.0694     0     0.167  0.194     0.25      0
```

When we analyze NAs in one of the proteins from the group which is missing a small proportion, we realize that all NAs in this group are from the same observations. They belong to the MouseID 3426_13, 3426_14 and 3426_15.

```
ds %>% select(-Genotype,-Behavior,-Treatment,-class) %>%
  filter(is.na(BDNF_N))
```

```
## # A tibble: 3 x 78
##   MouseID DYRK1A_N ITSN1_N BDNF_N NR1_N NR2A_N pAKT_N pBRAF_N pCAMKII_N
##   <chr>      <dbl>   <dbl>  <dbl> <dbl>  <dbl>  <dbl>   <dbl>     <dbl>
## 1 3426_13       NA      NA     NA    NA     NA     NA      NA        NA
## 2 3426_14       NA      NA     NA    NA     NA     NA      NA        NA
## 3 3426_15       NA      NA     NA    NA     NA     NA      NA        NA
##   pCREB_N pELK_N pERK_N pJNK_N PKCA_N pMEK_N pNR1_N pNR2A_N pNR2B_N
##     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>   <dbl>   <dbl>
## 1      NA     NA     NA     NA     NA     NA     NA      NA      NA
## 2      NA     NA     NA     NA     NA     NA     NA      NA      NA
## 3      NA     NA     NA     NA     NA     NA     NA      NA      NA
##   pPKCAB_N pRSK_N AKT_N BRAF_N CAMKII_N CREB_N ELK_N ERK_N GSK3B_N JNK_N
##      <dbl>  <dbl> <dbl>  <dbl>    <dbl>  <dbl> <dbl> <dbl>   <dbl> <dbl>
## 1       NA     NA    NA     NA       NA     NA    NA    NA      NA    NA
## 2       NA     NA    NA     NA       NA     NA    NA    NA      NA    NA
## 3       NA     NA    NA     NA       NA     NA    NA    NA      NA    NA
##   MEK_N TRKA_N RSK_N APP_N Bcatenin_N SOD1_N MTOR_N P38_N pMTOR_N DSCR1_N
##   <dbl>  <dbl> <dbl> <dbl>      <dbl>  <dbl>  <dbl> <dbl>   <dbl>   <dbl>
## 1    NA     NA    NA    NA         NA     NA     NA    NA      NA      NA
## 2    NA     NA    NA    NA         NA     NA     NA    NA      NA      NA
## 3    NA     NA    NA    NA         NA     NA     NA    NA      NA      NA
##   AMPKA_N NR2B_N pNUMB_N RAPTOR_N TIAM1_N pP70S6_N NUMB_N P70S6_N pGSK3B_N
##     <dbl>  <dbl>   <dbl>    <dbl>   <dbl>    <dbl>  <dbl>   <dbl>    <dbl>
## 1      NA     NA      NA       NA      NA       NA  0.180   0.976    0.193
## 2      NA     NA      NA       NA      NA       NA  0.189   1.01     0.192
## 3      NA     NA      NA       NA      NA       NA  0.184   1.04     0.185
##   pPKCG_N CDK5_N  S6_N ADARB1_N AcetylH3K9_N RRP1_N BAX_N ARC_N ERBB4_N
##     <dbl>  <dbl> <dbl>    <dbl>        <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1    2.66  0.304 0.524    1.05         0.237  0.177 0.171 0.116   0.160
## 2    2.64  0.335 0.525    0.999        0.237  0.183 0.185 0.119   0.165
## 3    2.64  0.329 0.526    1.00         0.242  0.190 0.167 0.123   0.155
##   nNOS_N Tau_N GFAP_N GluR3_N GluR4_N IL1B_N P3525_N pCASP9_N PSD95_N
##    <dbl> <dbl>  <dbl>   <dbl>   <dbl>  <dbl>   <dbl>    <dbl>   <dbl>
## 1  0.175 0.288  0.120   0.189   0.114  0.505   0.317     1.47    2.15
## 2  0.191 0.290  0.128   0.192   0.115  0.537   0.322     1.50    2.15
## 3  0.173 0.300  0.128   0.184   0.122  0.542   0.320     1.60    2.22
##   SNCA_N Ubiquitin_N pGSK3B_Tyr216_N SHH_N BAD_N BCL2_N pS6_N pCFOS_N SYP_N
##    <dbl>       <dbl>           <dbl> <dbl> <dbl>  <dbl> <dbl>   <dbl> <dbl>
## 1  0.162        1.21           0.998 0.195 0.182  0.134 0.116  0.113  0.406
## 2  0.169        1.22           0.998 0.194 0.180  0.134 0.119  0.121  0.423
## 3  0.162        1.28           1.06  0.207 0.181  0.140 0.123  0.0983 0.427
##   H3AcK18_N EGR1_N H3MeK4_N CaNA_N
##       <dbl>  <dbl>    <dbl>  <dbl>
## 1     0.152  0.163    0.209   1.44
## 2     0.175  0.185    0.195   1.44
## 3     0.172  0.201    0.233   1.50
```

Also related proteins are searched by correlations.

First of all, proteins are listed and all the possible combinations of pairs of proteins are calculated. There are 2926 possible combinations.

```
proteins <- ds %>% select(-MouseID,-Genotype,-Behavior,-Treatment,-class) %>% names(.)
combs <- combinations(length(proteins),2,proteins)
nrow(combs)
```

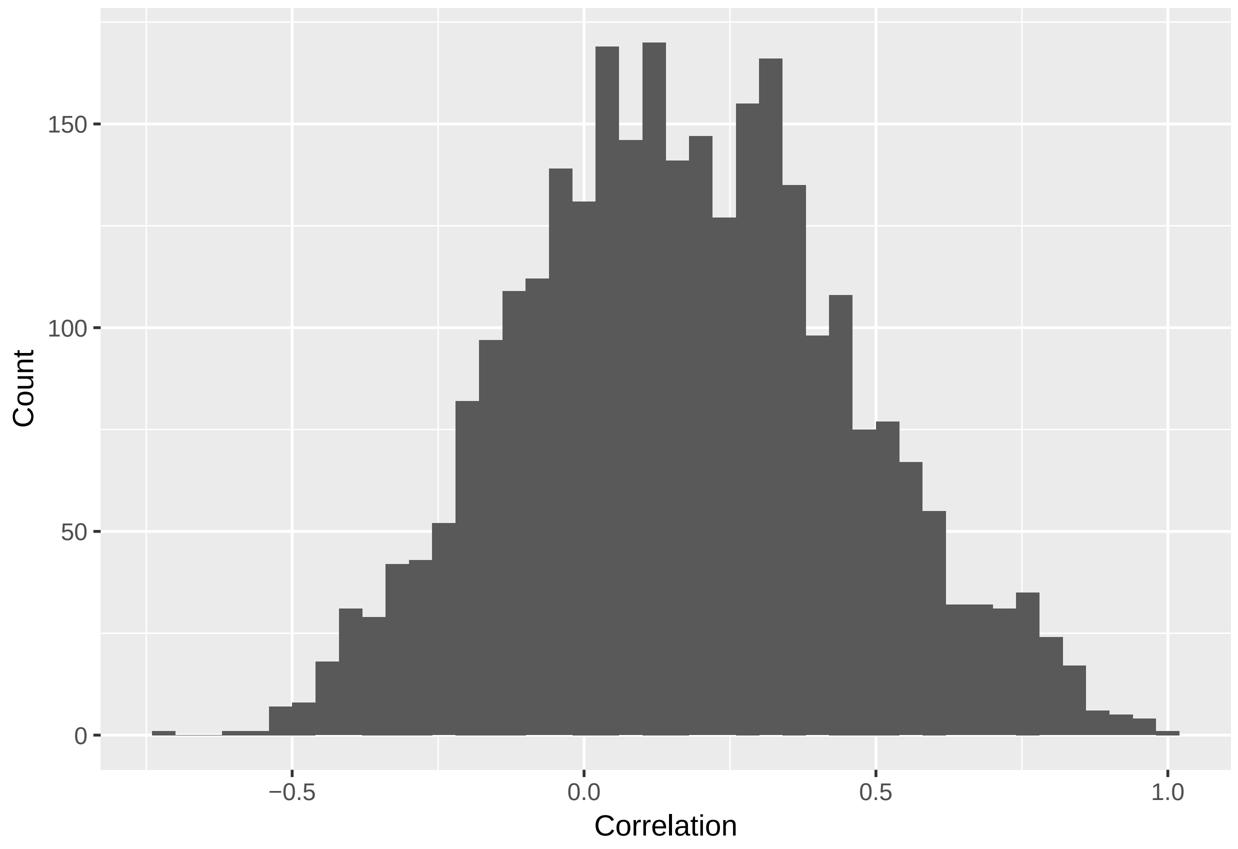
```
## [1] 2926
```

The correlation of each pair of proteins is calculated. NAs are not taken in count.

```
cors <- vector("numeric")
for (i in 1:nrow(combs)){
  cors[i] <- cor(ds[[combs[i,1]]],ds[[combs[i,2]]], use="complete.obs")
}
```

Distribution of correlations is displayed. Best correlations (>90%) are searched. There are 10 correlations greater than 0.9, one of them is equal to 1. There are no correlations lower than -0.9. The lowest correlations is around -0.7.

```
ggplot(data=as.data.frame(cors),aes(cors)) + geom_histogram(binwidth = 0.04) +
xlab("Correlation") + ylab("Count")
```



```
cors[cors > 0.9]
```

```
##  [1] 1.0000000 0.9146104 0.9595779 0.9176078 0.9269842 0.9595118 0.9457194
##  [8] 0.9062887 0.9478717 0.9065036
```

```
cors[cors < -0.9]
```

```
## numeric(0)
```

```
min(cors)
```

```
## [1] -0.700474
```

The proteins with correlation 1 are found: ARC_N and pS6_N.

```
combs[which(cors>0.99),]
```

```
## [1] "ARC_N" "pS6_N"
```

This correlation is visually checked. These 2 proteins have exactly the same values.

```
ds[sample(nrow(ds), 10),] %>% select(ARC_N, pS6_N)
```

```
## # A tibble: 10 x 2
##     ARC_N  pS6_N
##     <dbl>  <dbl>
##  1 0.103  0.103
##  2 0.135  0.135
##  3 0.123  0.123
##  4 0.125  0.125
##  5 0.130  0.130
##  6 0.129  0.129
##  7 0.0983 0.0983
##  8 0.122  0.122
##  9 0.138  0.138
## 10 0.139  0.139
```

```
sum(ds$ARC_N != ds$pS6_N)
```

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

Now, proteins influenced by the mouse parameters (genotype, behavior and treatment) are searched.

The first parameter focused is genotype. Data is grouped by genotype and mean of each group is calculated. Then the total mean is calculated. Finally, standard deviations of grouped data are also calculated.

```
dst1 <- ds %>%
  select(-MouseID,-Behavior,-Treatment,-class) %>%
  group_by(Genotype) %>%
  summarise_all(funs(mean(., na.rm = TRUE))) %>%
  select(-Genotype)
dst2 <- ds %>%
  select(-MouseID,-Genotype,-Behavior,-Treatment,-class) %>%
  summarise_all(funs(mean(., na.rm = TRUE)))
dst3 <- ds %>%
  select(-MouseID,-Behavior,-Treatment,-class) %>%
  group_by(Genotype) %>%
  summarise_all(funs(sd(., na.rm = TRUE))) %>%
  select(-Genotype)
```

A new data frame with genotype related data is set. It includes the means and standard deviations of each group: Control and Ts65Dn. In addition, the difference between the group means divided by the total mean is saved as "meanDiff". It is an indicator of how much different are the means of each group in proportion. Also the mean of the both standard deviations is saved as "meanSD". Finally a new indicator is calculated: "opt". Opt is calculated as the meanDiff divided by the meanSD and it is and indicator of how optimal is this protein to help predict the genotype. High opt values relate to proteins whose groups means are more different and their standard deviations are low.

```r
geno <- data.frame( name = proteins,
  cMean = as.numeric(as.vector(dst1[1,])),
  tsMean = as.numeric(as.vector(dst1[2,])),
  meanDiff = abs(as.numeric(as.vector(dst1[1,])) -
                 as.numeric(as.vector(dst1[2,])))/as.numeric(as.vector(dst2[1,])),
  cSD = as.numeric(as.vector(dst3[1,])),
  tsSD = as.numeric(as.vector(dst3[2,])),
  meanSD = (as.numeric(as.vector(dst3[1,])) + as.numeric(as.vector(dst3[2,])))/2 ) %>%
  mutate(opt = meanDiff/meanSD)
```

The proteins with higher opt values are displayed

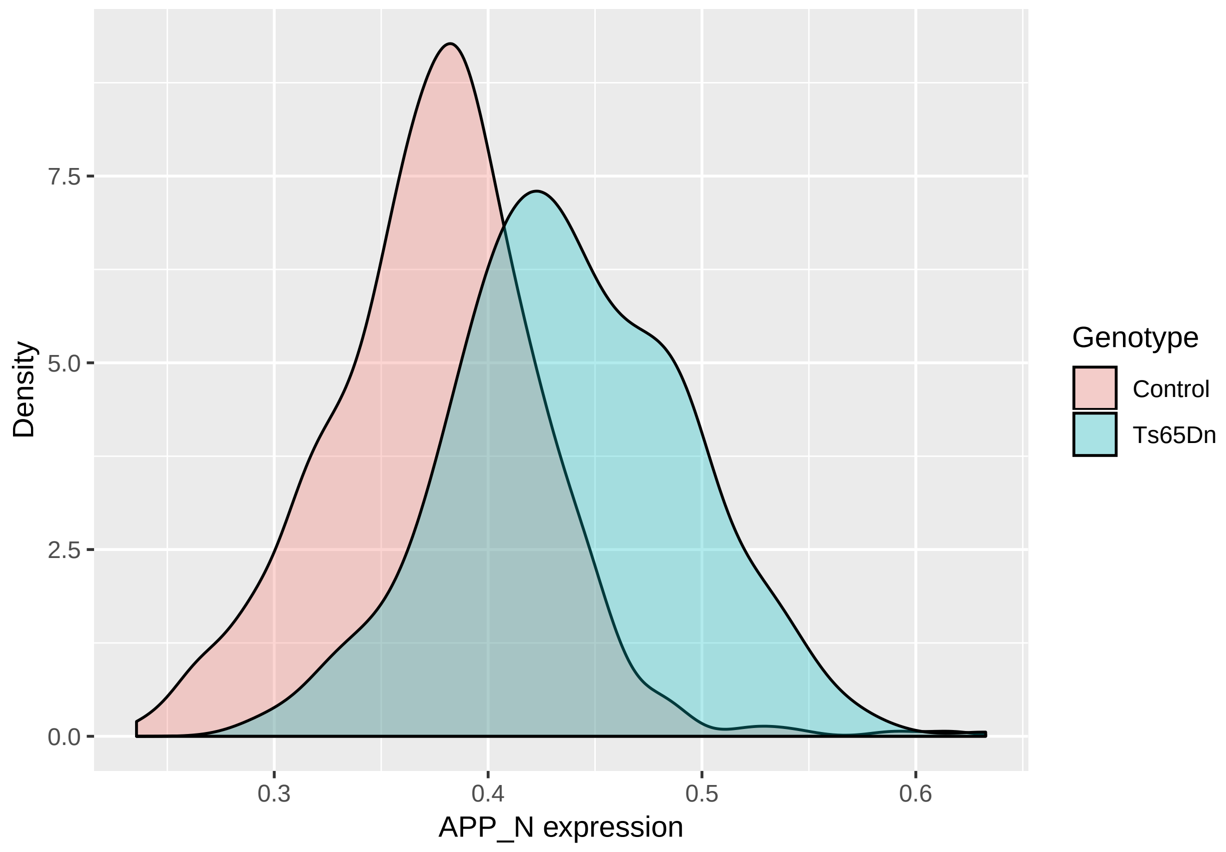
```r
head(geno %>% select(name,meanDiff,meanSD,opt) %>% arrange(desc(opt)))
```

```
##            name   meanDiff     meanSD      opt
## 1         APP_N 0.15847222 0.05222679 3.034309
## 2     H3AcK18_N 0.15650336 0.05659538 2.765303
## 3         Tau_N 0.17849161 0.06653252 2.682773
## 4       GluR3_N 0.08157970 0.03345076 2.438800
## 5 AcetylH3K9_N 0.41785930 0.18006363 2.320620
## 6         ARC_N 0.03213484 0.01417105 2.267640
```

The highest opt belongs to the protein APP_N. The distribution of both groups from this protein is displayed.

```r
ds %>% ggplot(aes(x=APP_N,fill=Genotype)) + geom_density(alpha=.3) +
xlab("APP_N expression") + ylab("Density")
```

```
## Warning: Removed 3 rows containing non-finite values (stat_density).
```



Next, the same genotype analysis is performed with the behavior parameter.

```
dst1 <- ds %>%
  select(-MouseID,-Genotype,-Treatment,-class) %>%
  group_by(Behavior) %>%
  summarise_all(funs(mean(., na.rm = TRUE))) %>%
  select(-Behavior)
dst2 <- ds %>%
  select(-MouseID,-Genotype,-Behavior,-Treatment,-class) %>%
  summarise_all(funs(mean(., na.rm = TRUE)))
dst3 <- ds %>%
  select(-MouseID,-Genotype,-Treatment,-class) %>%
  group_by(Behavior) %>%
  summarise_all(funs(sd(., na.rm = TRUE))) %>%
  select(-Behavior)

beha <- data.frame( name = proteins,
  csMean = as.numeric(as.vector(dst1[1,])),
  scMean = as.numeric(as.vector(dst1[2,])),
  meanDiff = abs(as.numeric(as.vector(dst1[1,])) -
                 as.numeric(as.vector(dst1[2,])))/as.numeric(as.vector(dst2[1,])),
  csSD = as.numeric(as.vector(dst3[1,])),
  scSD = as.numeric(as.vector(dst3[2,])),
  meanSD = (as.numeric(as.vector(dst3[1,])) + as.numeric(as.vector(dst3[2,])))/2 ) %>%
  mutate(opt = meanDiff/meanSD)
```

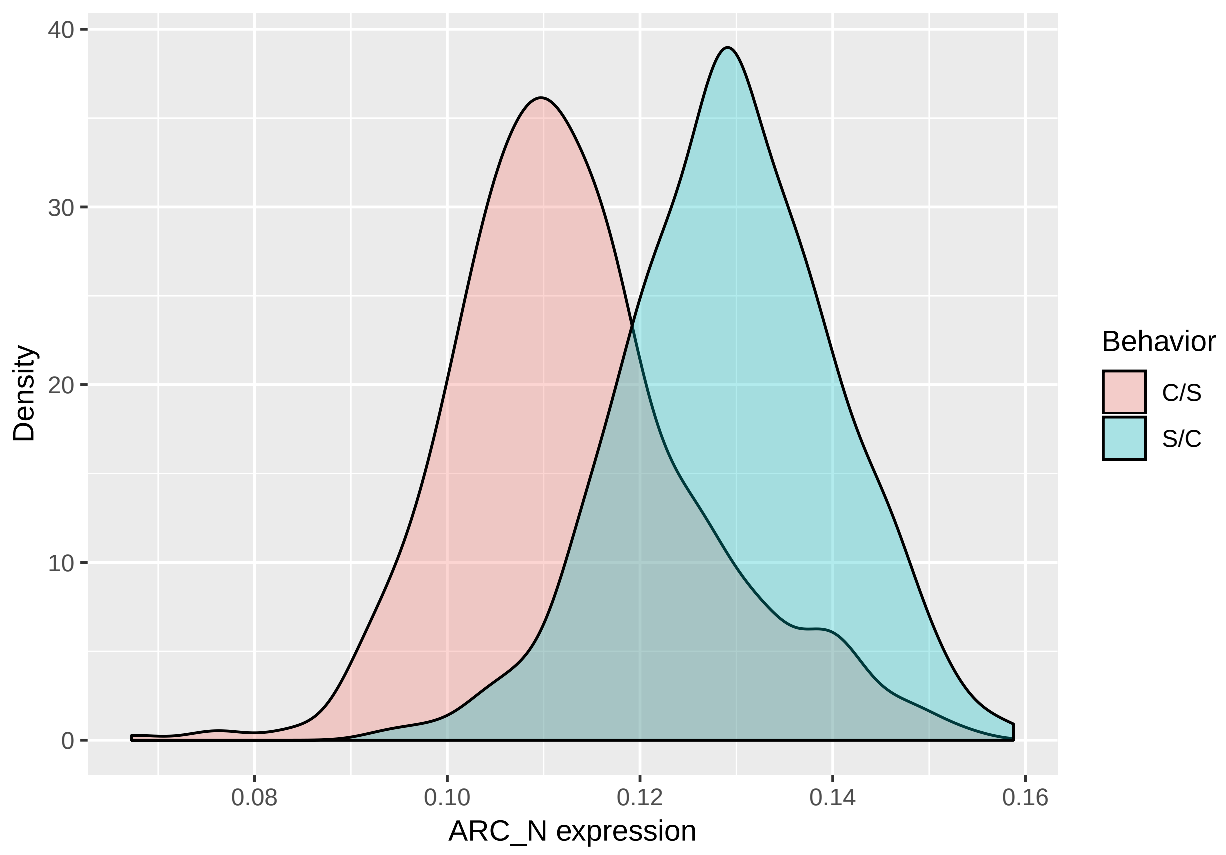
In this case, the "opt" values are quite higher than with the genotype. ARC_N is heading the list.

```
head(beha %>% select(name,meanDiff,meanSD,opt) %>% arrange(desc(opt)))
```

```
##       name  meanDiff     meanSD       opt
## 1    ARC_N 0.1324815 0.01178050 11.245824
## 2    pS6_N 0.1324815 0.01178050 11.245824
## 3   SNCA_N 0.1502541 0.02094363  7.174211
## 4 pGSK3B_N 0.1134443 0.01695464  6.691051
## 5   SOD1_N 0.7993907 0.14247384  5.610789
## 6   EGR1_N 0.1808527 0.03629662  4.982633
```

The distribution of both groups from ARC_N is displayed.

```
ds %>% ggplot(aes(x=ARC_N,fill=Behavior)) + geom_density(alpha=.3) +
xlab("ARC_N expression") + ylab("Density")
```



Finally, the same analysis is repeated with the las parameter: Treatment.

```
dst1 <- ds %>%
  select(-MouseID,-Genotype,-Behavior,-class) %>%
  group_by(Treatment) %>%
  summarise_all(funs(mean(., na.rm = TRUE))) %>%
  select(-Treatment)
dst2 <- ds %>%
  select(-MouseID,-Genotype,-Behavior,-Treatment,-class) %>%
  summarise_all(funs(mean(., na.rm = TRUE)))
dst3 <- ds %>%
  select(-MouseID,-Genotype,-Behavior,-class) %>%
  group_by(Treatment) %>%
  summarise_all(funs(sd(., na.rm = TRUE))) %>%
  select(-Treatment)

treat <- data.frame( name = proteins,
  mMean = as.numeric(as.vector(dst1[1,])),
  sMean = as.numeric(as.vector(dst1[2,])),
  meanDiff = abs(as.numeric(as.vector(dst1[1,])) -
               as.numeric(as.vector(dst1[2,])))/as.numeric(as.vector(dst2[1,])),
  mSD = as.numeric(as.vector(dst3[1,])),
  sSD = as.numeric(as.vector(dst3[2,])),
  meanSD = (as.numeric(as.vector(dst3[1,])) + as.numeric(as.vector(dst3[2,])))/2 ) %>%
  mutate(opt = meanDiff/meanSD)
```

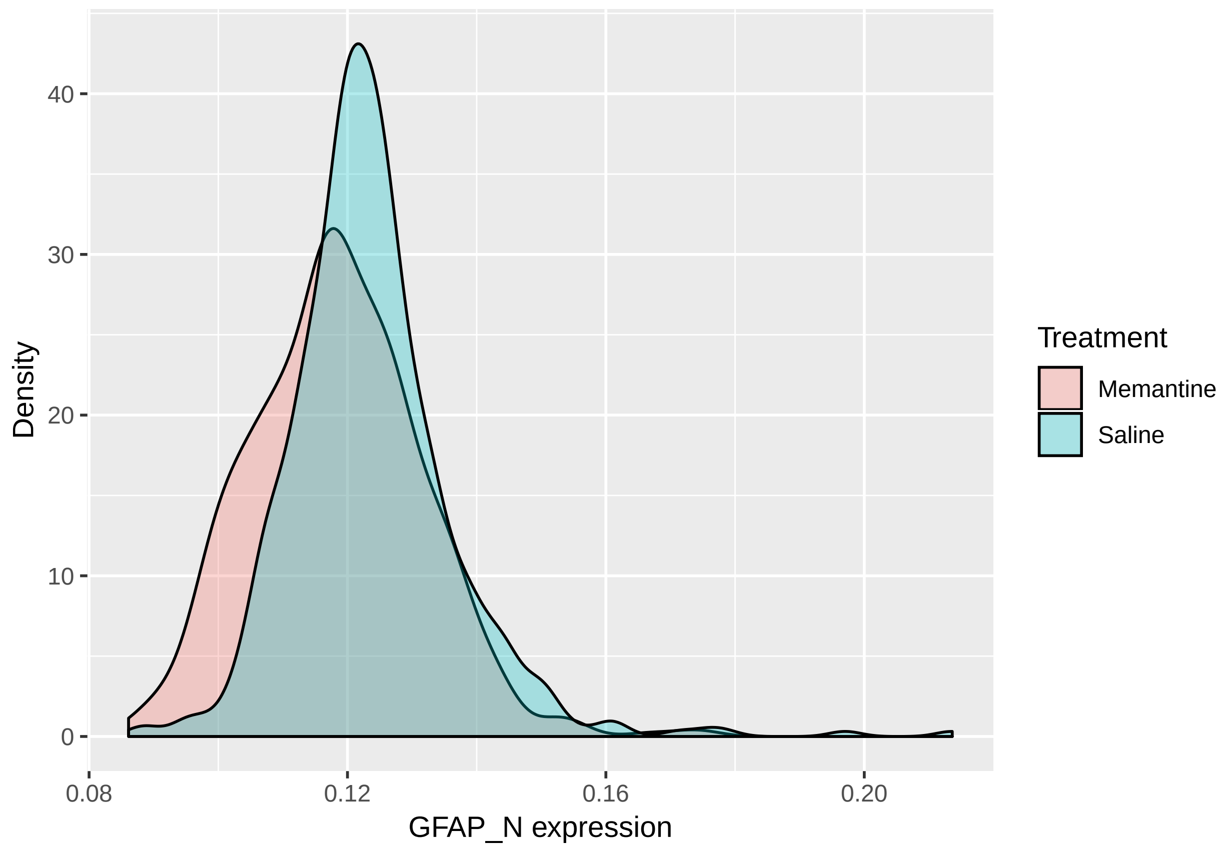
Opt values are similar to the genotype. GFAP_N has achieved the highest value.

```r
head(treat %>% select(name,meanDiff,meanSD,opt) %>% arrange(desc(opt)))
```

```
##       name   meanDiff     meanSD      opt
## 1   GFAP_N 0.04430634 0.01295433 3.420196
## 2 pGSK3B_N 0.05385350 0.01884599 2.857557
## 3   NUMB_N 0.08020524 0.02839151 2.824973
## 4    ARC_N 0.03230695 0.01400961 2.306057
## 5    pS6_N 0.03230695 0.01400961 2.306057
## 6  pBRAF_N 0.05597307 0.02659705 2.104484
```

The distribution of both groups from GFAP_N is displayed.

```r
ds %>% ggplot(aes(x= GFAP_N,fill=Treatment)) + geom_density(alpha=.3) +
xlab("GFAP_N expression") + ylab("Density")
```



As shown in the lasts figures, there is a big overlap region between the groups. Trying to predict the Genotype with the best predictors would be better than guessing but with an important error. However, there are many proteins and all together may be enough to reach an accurate prediction. Therefore, the machine learning strategy for this data will be random forest. It is a suitable method for many predictors.

Data partition and modeling

Data is split in a train set and a test set, with a 90% and a 10% of the whole data set respectively. The seed is set to 1.

```r
set.seed(1)
partition <- createDataPartition(y = ds$Genotype, times = 1, p = 0.1, list = FALSE)
train <- ds[-partition,]
test <- ds[partition,]
```

The length of each data set is verified.

```r
nrow(train)
```

```
## [1] 972
```

```r
nrow(test)
```

```
## [1] 108
```

The random forest package is loaded.

```r
if(!require(randomForest)) install.packages("randomForest")
```

Fit models are developed for the three mice parameters. First, a subset of the train set is made. All not necessary columns are removed and the "y" column is transformed in factors. The model is fit with all the proteins as predictors. NAs are imputed with the median/mode by using the "na.roughfix"" function.

This process is repeated with the genotype,

```r
t <- train %>% select(-MouseID,-Behavior,-Treatment,-class) %>%
               mutate(Genotype=as.factor(Genotype))
genoFit <- randomForest(Genotype ~ ., data = t, na.action=na.roughfix)
```

behavior,

```r
t <- train %>% select(-MouseID,-Genotype,-Treatment,-class) %>%
               mutate(Behavior=as.factor(Behavior))
behaFit <- randomForest(Behavior ~ ., data = t, na.action=na.roughfix)
```

and treatment.

```r
t <- train %>% select(-MouseID,-Genotype,-Behavior,-class) %>%
           mutate(Treatment=as.factor(Treatment))
treatFit <- randomForest(Treatment ~ ., data = t, na.action=na.roughfix)
```

Predictions from these 3 models are performed in the results section.

The accuracy of the random forest method is also evaluated with a smaller number of predictors. Monte Carlo simulations are run estimate this accuracy. This is a heavy simulation, so it will be run only with the first parameter: genotype. This parameter has low "opt" values, very similar to the treatment parameter. Genotype and treatment are probably the limiting parameters when looking for the minimum necessary number of proteins to achieve a reasonable accuracy.

Different numbers of random proteins are used as predictors. The mean accuracy of predictions with the same number of predictors are saved. This number of predictors goes from 1 to 77.

```r
results <- vector("numeric")
  pnumber <- 1:length(proteins)
  for (ps in pnumber){

    r <- replicate(100,{
      t <- train %>% select(-MouseID,-Behavior,-Treatment,-class) %>%
                     mutate(Genotype=as.factor(Genotype)) %>%
                     select(sample(1:77, ps),78)
      fit <- randomForest(Genotype ~ ., data = t, na.action=na.roughfix)
      pred <- predict(fit,  na.roughfix(test[,2:78]))
      mean(pred==test$Genotype)
    })

    # This line is for me to know how is it going
    message(" || MC Simulation ", ps,"/", length(proteins))

    results[ps] <- mean(r)
  }
```

The results of this simulation are displayed in the results section.

Results

Fit models are used to predict genotype, behavior and treatment in the validation set. NAs in the validation set are also imputed with the na.roughfix function. Finally, the accuracy with the test set is measured.

Genotype results in a 100% accuracy.

```
genoPred <- predict(genoFit,  na.roughfix(test[,2:78]))
mean(genoPred==test$Genotype)
```

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

Behavior results in a 100% accuracy.

```
behaPred <- predict(behaFit,  na.roughfix(test[,2:78]))
mean(behaPred==test$Behavior)
```

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

Treatment results in 100% accuracy.

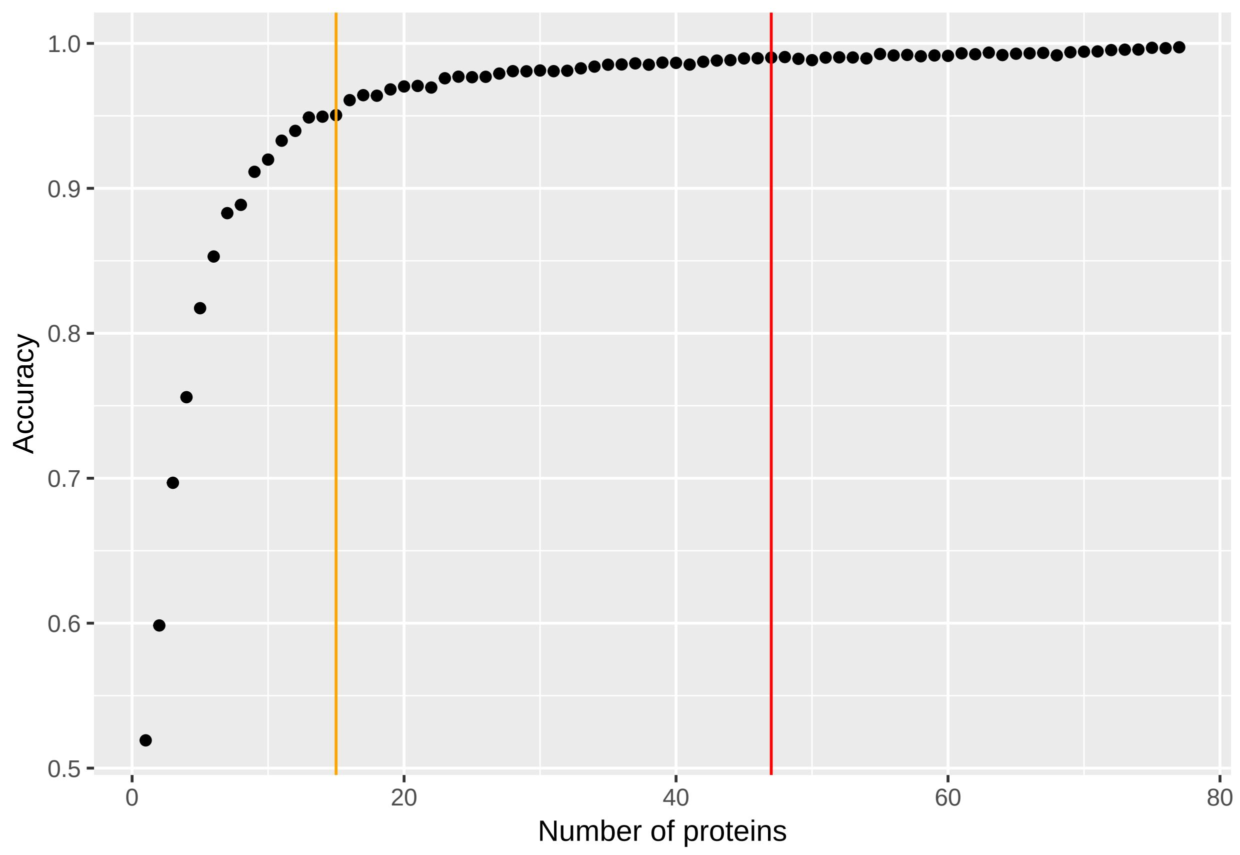
```
treatPred <- predict(treatFit,  na.roughfix(test[,2:78]))
mean(treatPred==test$Treatment)
```

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

All parameters are perfectly predicted.

With a lower number of proteins to make the prediction it may result in loss of accuracy. Next figure illustrates how the number of proteins and the accuracy are related.

```
data.frame(proteins=pnumber, accuracy=results) %>% ggplot(aes(proteins,accuracy)) +
   geom_point() + geom_vline(xintercept=min(pnumber[results > 0.99]), colour="red") +
   geom_vline(xintercept=min(pnumber[results > 0.95]), colour="orange") +
   xlab("Number of proteins") + ylab("Accuracy")
```



The number of proteins needed to reach a 95% and a 99% of accuracy are displayed.

```
min(pnumber[results > 0.95])
```

```
## [1] 15
```

```
min(pnumber[results > 0.99])
```

```
## [1] 47
```

Conclusions

All 3 parameters: genotype, behavior and treatment, can be perfectly predicted from the concentration of proteins found in the data set.

Accuracy increases with the number of proteins used to fit the model in a logarithmic scale. Accuracy higher than 95% can be achieved with around 20 proteins and higher than 99% with around 50 proteins.