# Fast and furious boosting: a simulation study in proteomic variable selection and prediction

Quinton Neville, qn2119

4/26/2020

## 1. Simulation

### 1.1 Proteomic Expression Level Data Generation

N observations = 50, 100, 500, 1000 N vars = 100, 200, 500? Clusters = 1, 5, 10 ? Plan is to generate "groups" of correlated gamma rv's (1, 5, 10, 20), then generate a binary outcome based on 1, 2, 5, 10 signals (1 signal per cluster max) with a linear, quadratic, exponential, and "fill in the blank"

```
dt <- genData(1000, def)

  dt %>%
  as_tibble() %>%
  gather(var, value, -dfSim) %>%
  group_by(var) %>%
  summarize(mean = mean(value))

dt %>%
  as_tibble() %>%
  gather(var, value, -dfSim) %>%
  ggplot(aes(x = value, fill = var, colour = var)) +
  geom_density(alpha = 0.6) +
  scale_colour_viridis_d() +
  scale_fill_viridis_d()


gamma_generator <- function(mean = 1, precision = 10) {

  pmap_df(list(x = mean,
               y = precision,
               id = str_c("gamma_", str_c(1:length(mean)))
               ),
          function(x, y, id) {
               defData(varname = id,
               dist = "gamma",
               formula = x,
               variance = y,
               id = "idnum")
            }
          )
}

mean <- 1:99/100
precision <- rep(10, length = length(mean))

beta_generator(mean, precision)

addCorFlex(genData(10, beta_generator(0.5, 10)), beta_generator(0.5, 10), tau = 0.9, corstr = "cs")
```

```
def <- defData(varname="xUni", dist="uniform", formula="10;20", id = "myID")
def <- defData(def, varname="xNorm", formula="xUni * 2", dist="normal", variance=8)
dt <- genData(250, def)
mu <- c(3, 8, 15)
sigma <- c(1, 2, 3)
dtAdd <- addCorData(dt, "myID", mu = mu, sigma = sigma, rho = .7, corstr = "cs")
dtAdd
```

## 2. Application - D.S. in Mice

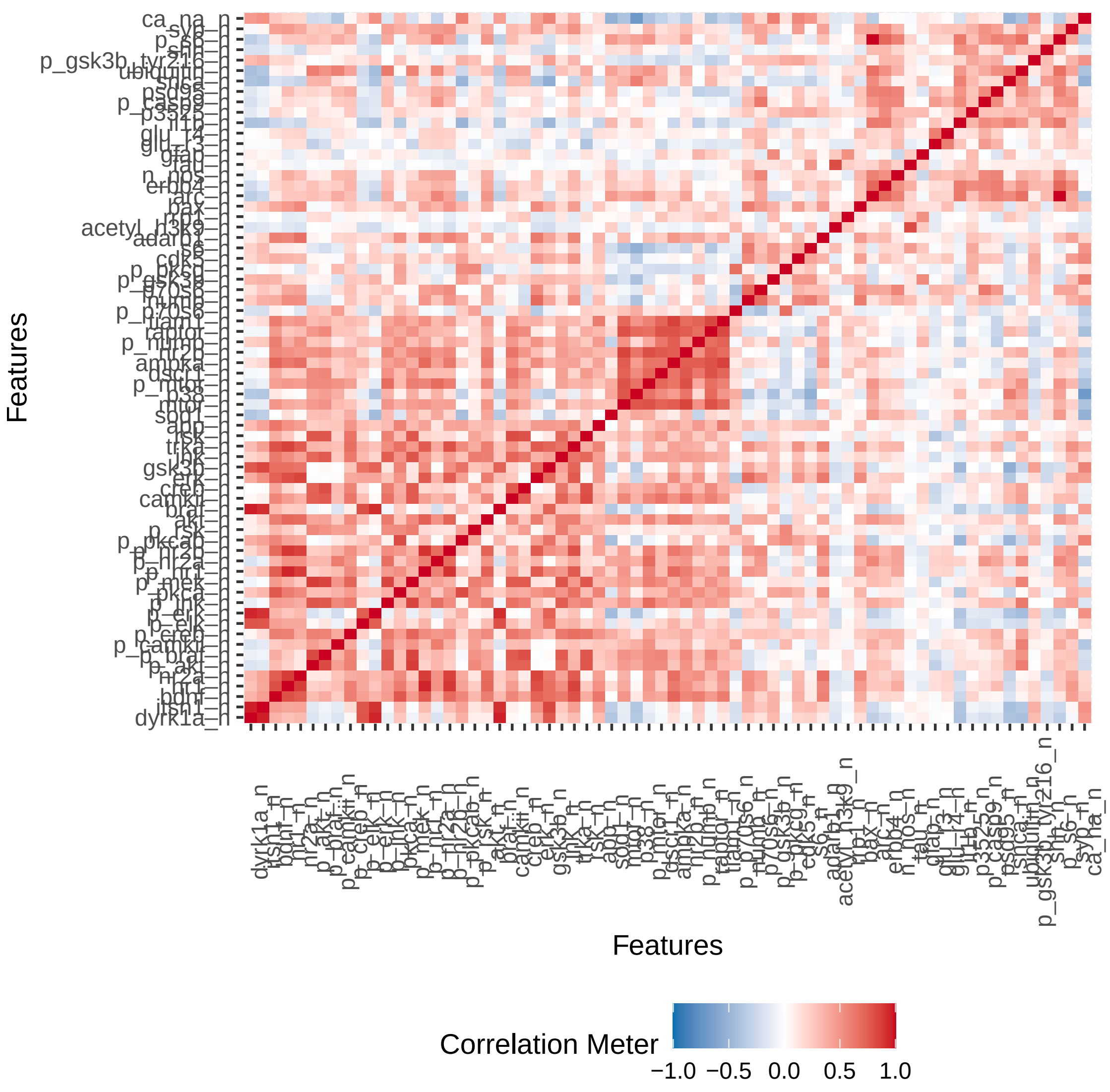
```
down.df <- read_csv("./data/mice_down_syndrome.csv") %>%
  filter(!(MouseID %in% c("3426_13", "3426_14", "3426_15"))) %>%
  dplyr::select( -c("BCL2_N","H3MeK4_N","BAD_N","EGR1_N","H3AcK18_N","pCFOS_N","Bcatenin_N","MEK_N","EL
  janitor::clean_names() %>%
  rename(
    id = mouse_id,
    down_syndrome = class
  ) %>%
  mutate(
    down_syndrome = ifelse(down_syndrome == "Control", FALSE, TRUE)
  ) %>%
  dplyr::select(id, down_syndrome, everything())
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   MouseID = col_character(),
##   Class = col_character()
## )

## See spec(...) for full column specifications.
```

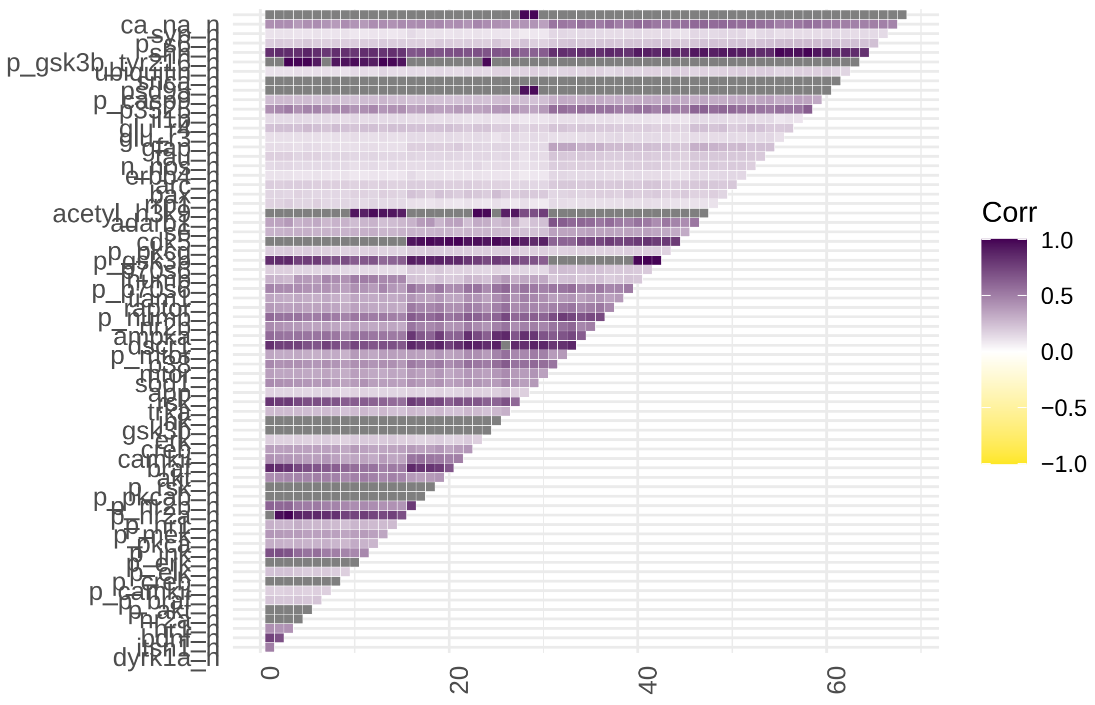
```
#Plot Correlation structure, try to mimic with simulation

introduce(down.df)
```

```
## # A tibble: 1 x 9
##    rows columns discrete_columns continuous_colu~ all_missing_col~
##   <int>   <int>            <int>            <int>            <int>
## 1  1077      70                2               68                0
## # ... with 4 more variables: total_missing_values <int>, complete_rows <int>,
## #   total_observations <int>, memory_usage <dbl>
```

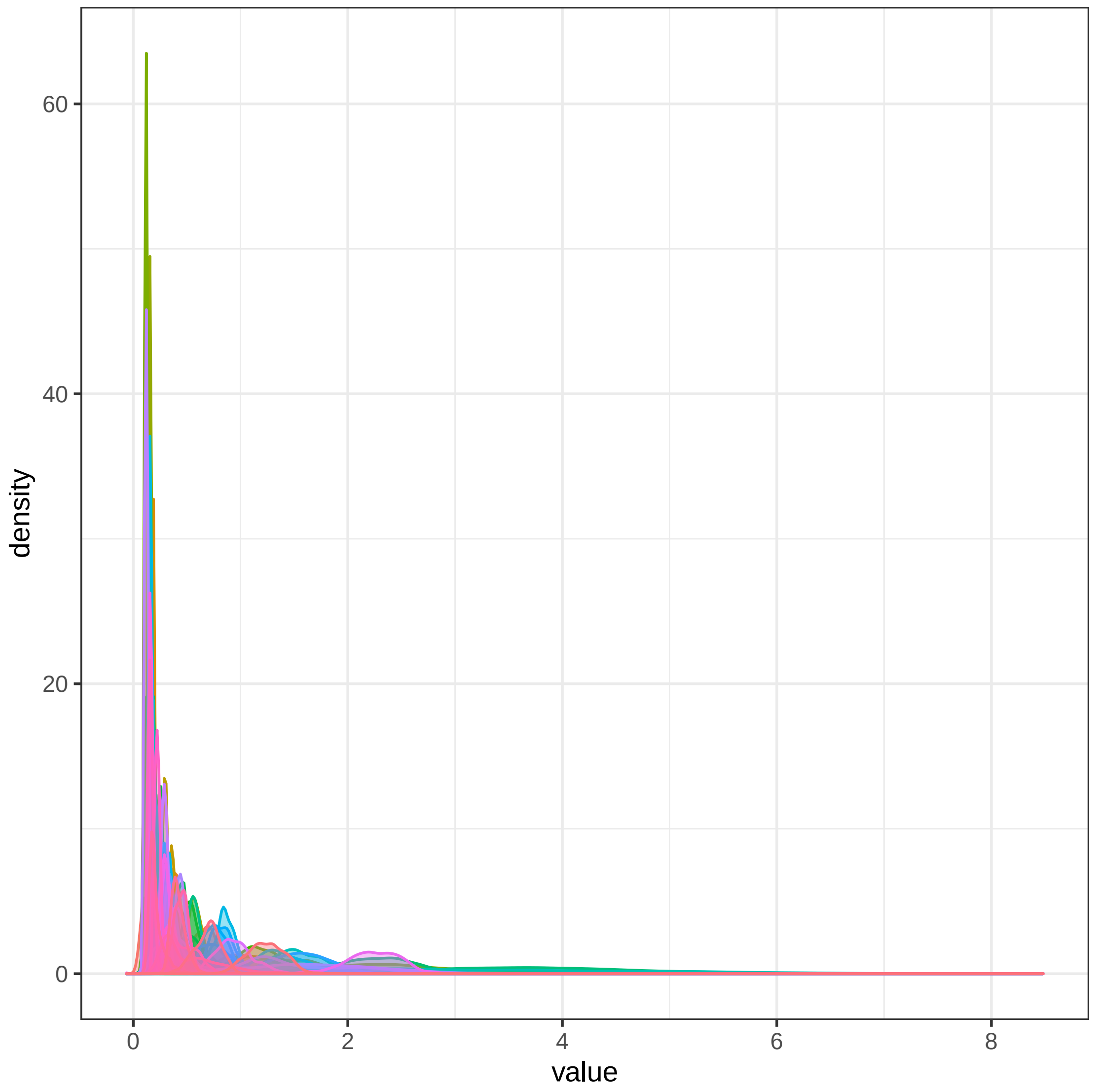
```
DataExplorer::plot_correlation(down.df, type = "continuous")
```



```
down.df[, -1] %>%
ggcorrplot(., type = "upper",
               outline.col = "white",
               colors = c("#FDE725FF", "white", "#440154FF"),
               lab = FALSE) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 10),  # Order: top, right, bottom, lef
        axis.text.y = element_text(angle = 0, vjust = 1, size = 10),
        plot.title = element_text(hjust = 0.5))
```



```r
down.df %>%
  gather(variable, value, -c(id, down_syndrome)) %>%
  ggplot(aes(x = value, colour = variable, fill = variable)) +
  geom_density(alpha = 0.4) +
  theme(legend.position = "none")
```


density
value
0
20
40
60
0
2
4
6
8