Log Journal Project Theme 9

J.A. Busker (352905 BFV3)

EDA of yeast data: Protein Localization

The data has been retrived from uci. The dataset has been gathered of 1484 yeast sequences from SWISS-PROT using the annotations from YPD.

Attributes information:

- **Sequence Name**: Accession number for the SWISS-PROT database.
- **mcg**: McGeoch's method for signal sequence recognition.
- **gvh**: von Heijne's method for signal sequence recognition.
- **alm**: Score of the ALOM membrane spanning region prediction program.
- **mit**: Score of discriminant analysis of the amino acid content of the N-terminal region (20 residues long) of mitochondrial and non-mitochondrial proteins.
- **erl**: Presence of "HDEL" substring (thought to act as a signal for retention in the endoplasmic reticulum lumen). Binary attribute.
- **pox**: Peroxisomal targeting signal in the C-terminus.
- **vac**: Score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins.
- **nuc**: Score of discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins.
- **Class Distribution**: The class is the localization site. Consisting of (abbreviation (full name) the amount):

CYT	(cytosolic or cytoskeletal)	463
NUC	(nuclear)	429
MIT	(mitochondrial)	244
ME3	(membrane protein, no N-terminal signal)	163
ME2	(membrane protein, uncleaved signal)	51
ME1	(membrane protein, cleaved signal)	44
EXC	(extracellular)	37
VAC	(vacuolar)	30
POX	(peroxisomal)	20
ERL	(endoplasmic reticulum lumen)	5

Codebook

Since there is not a codebook. I created my own codebook.

```
# All the attributes name
attr_names <- c("seq.name", "mcg", "gvh", "alm", "mit", "erl",
                "pox", "vac","nuc", "loc.site" )

data_types <- c("str", "float", "float", "float", "float", "float",
                "bool", "float", "float", "factor")

# The description of the labels
data_labels <- c("Accession number for the SWISS-PROT database",
                 "McGeoch's method for signal sequence recognition",
                 "von Heijne's method for signal sequence recognition",
                 "Score of the ALOM membrane spanning region prediction program",
                 "Score of discriminant analysis of the amino
                 acid content of the N-terminal region",
                 "Presence of 'HDEL' substring",
                 "Peroxisomal targeting signal in the C-terminus",
                 "Score of discriminant analysis of the amino acid content
                 of vacuolar and extracellular proteins",
                 "Score of discriminant analysis of nuclear
                 localization signals of nuclear and non-nuclear proteins",
                 "The class is the localization site")

codebook <- data.frame(Name=attr_names,
                       Fullname=data_labels,
                       Datatypes=data_types)

pander(codebook)
```

Name	Fullname	Datatypes
seq.name	Accession number for the SWISS-PROT database	str
mcg	McGeoch's method for signal sequence recognition	float
gvh	von Heijne's method for signal sequence recognition	float
alm	Score of the ALOM membrane spanning region prediction program	float
mit	Score of discriminant analysis of the amino acid content of the N-terminal region	float
erl	Presence of 'HDEL' substring	float
pox	Peroxisomal targeting signal in the C-terminus	bool
vac	Score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins	float
nuc	Score of discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins	float
loc.site	The class is the localization site	factor

Here is the codebook. With the attributes abbreviation, explanation and data types. As you can see there are many float datatypes.

Load the data

Now we need to load in the data. And change the column names to the attributes abbreviation, since these are non-existent. Let's give all the columns a name. And let's take a quick look at the data.

```
# Read the file in ass a tibble
data <- as_tibble(read.table("yeast.data", sep = ""))
colnames(data) <- attr_names
head(data)
```

```
# A tibble: 6 x 10
  seq.name     mcg   gvh   alm   mit   erl   pox   vac   nuc loc.site
  <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
1 ADT1_YEAST  0.58  0.61  0.47  0.13   0.5   0    0.48  0.22 MIT
2 ADT2_YEAST  0.43  0.67  0.48  0.27   0.5   0    0.53  0.22 MIT
3 ADT3_YEAST  0.64  0.62  0.49  0.15   0.5   0    0.53  0.22 MIT
4 AAR2_YEAST  0.58  0.44  0.57  0.13   0.5   0    0.54  0.22 NUC
5 AATM_YEAST  0.42  0.44  0.48  0.54   0.5   0    0.48  0.22 MIT
6 AATC_YEAST  0.51  0.4   0.56  0.17   0.5   0.5  0.49  0.22 CYT
```

The yeast data set contains scores per cellular localization sites.

Clean the data

We can drop the first columns since it is not necessary. Since the sequence names contribute nothing to create a prediction model.

```
# Drop the first column
data <- data[, -1]
head(data)
```

```
# A tibble: 6 x 9
    mcg   gvh   alm   mit   erl   pox   vac   nuc loc.site
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
1  0.58  0.61  0.47  0.13   0.5   0    0.48  0.22 MIT
2  0.43  0.67  0.48  0.27   0.5   0    0.53  0.22 MIT
3  0.64  0.62  0.49  0.15   0.5   0    0.53  0.22 MIT
4  0.58  0.44  0.57  0.13   0.5   0    0.54  0.22 NUC
5  0.42  0.44  0.48  0.54   0.5   0    0.48  0.22 MIT
6  0.51  0.4   0.56  0.17   0.5   0.5  0.49  0.22 CYT
```

The first column has been succesfully dropped from the data.

Are there any missing values? Let's take a quick look.

```
# Count all the NA values
sum(is.na(data))
```

```
[1] 0
```

There are not any missing values. So nothing needs to be changed for this.

Now we need to take a clearer look at the data set using `str()`.

```
str(data)
```

```
tibble [1,484 x 9] (S3: tbl_df/tbl/data.frame)
 $ mcg     : num [1:1484] 0.58 0.43 0.64 0.58 0.42 0.51 0.5 0.48 0.55 0.4 ...
 $ gvh     : num [1:1484] 0.61 0.67 0.62 0.44 0.44 0.4 0.54 0.45 0.5 0.39 ...
 $ alm     : num [1:1484] 0.47 0.48 0.49 0.57 0.48 0.56 0.48 0.59 0.66 0.6 ...
 $ mit     : num [1:1484] 0.13 0.27 0.15 0.13 0.54 0.17 0.65 0.2 0.36 0.15 ...
 $ erl     : num [1:1484] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
 $ pox     : num [1:1484] 0 0 0 0 0 0.5 0 0 0 0 ...
 $ vac     : num [1:1484] 0.48 0.53 0.53 0.54 0.48 0.49 0.53 0.58 0.49 0.58 ...
 $ nuc     : num [1:1484] 0.22 0.22 0.22 0.22 0.22 0.22 0.22 0.34 0.22 0.3 ...
 $ loc.site: chr [1:1484] "MIT" "MIT" "MIT" "NUC" ...
```

As you can see, the last column, loc.site, consists of characters, but this one needs to be converted to factors for clarity and complexity.

```
# Transform the last column to a factor
data$loc.site <- as.factor(data$loc.site)
str(data$loc.site)
```

```
 Factor w/ 10 levels "CYT","ERL","EXC",..: 7 7 7 8 7 1 7 8 7 1 ...
```

As you can see the data type of the column loc.site has been succesfully changed to factors.

There are alot of rows. Let's visualize the amount of each classification.

```
# A grid to fill in 28x53
df <- expand.grid(y = 1:28, x = 1:53)
# Sort the table
categ_table <- sort(table(data$loc.site), decreasing = T)
df$category <- factor(rep(names(categ_table), categ_table))

ggplot(df, aes(x = x, y = y, fill = category)) +
        geom_tile(color = "black", size = 0.5) +
        scale_x_continuous(expand = c(0, 0)) +
        scale_y_continuous(expand = c(0, 0), trans = 'reverse') +
        scale_fill_brewer(palette = "Set3") +
        labs(title="Waffle Chart", subtitle="'Class' of localization") +
        xlab(NULL) + ylab(NULL)
```

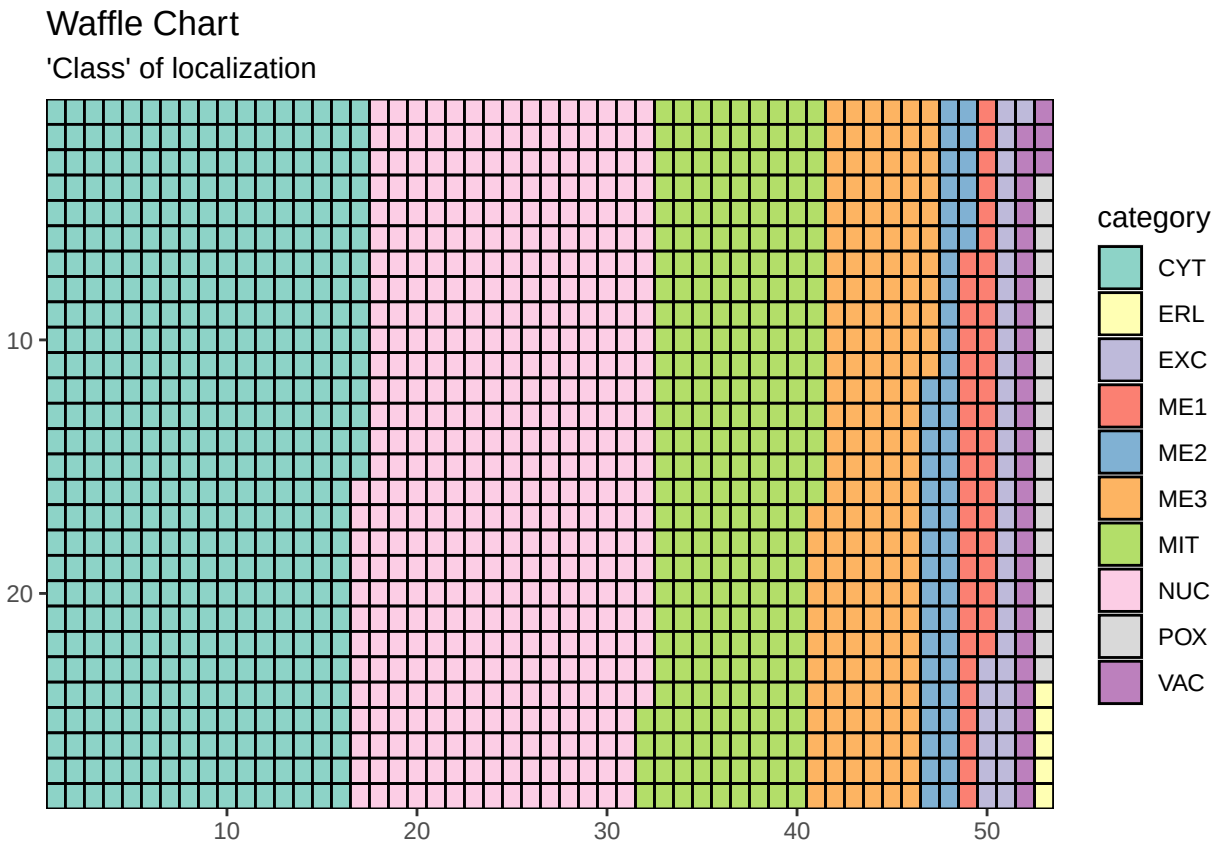

Figure 1: A waffle chart of the categorical composition

There are alot of CYT and NUC localization. This probably means that CYT has a greater variation in localisation than ERL.

The ERL variable also should be changed. As you can see, it is now a num data type but I needs to be a bool/binary datatype. Let's take a closer look at th ERL column.

```
summary(data$erl)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.5000  0.5000  0.5000  0.5047  0.5000  1.0000
```

This looks weird because the value should be 0 or 1.

There are alot of 0.5 values. But should not exist in this column. The variable need to be a bool, or in other words it must be a 0 or a 1. So all the 0.5 values needs to be changed to a 0.

```
table(data[, "erl"])
```

```
 0.5    1
1470   14
```

Before the data transformation there are 1470 counts of the value 0.5 and 14 of 1.

```
# Change every 0.5 value to a 0
data$erl[data$erl == 0.5] <- 0
table(data[,"erl"])
```

```
   0    1
1470   14
```

The data has been succesfully transformed. Now the datatype has to be changed to a bool.

```
# Change the datatype to a logical
data$erl <- as.logical(data$erl)
str(data$erl)
```

```
 logi [1:1484] FALSE FALSE FALSE FALSE FALSE FALSE ...
```

Now every column has the right datatype.

Univariate Analysis

Let's take a quick look at the data using `summary()`. We can drop column 5 and 9 for this. Column 5, ERL, is a logical datatype and column 9, loc.site, is a string datatype.

```
summary(data[, -c(5,9)])
```

```
      mcg              gvh              alm              mit        
 Min.   :0.1100   Min.   :0.1300   Min.   :0.21   Min.   :0.0000  
 1st Qu.:0.4100   1st Qu.:0.4200   1st Qu.:0.46   1st Qu.:0.1700  
 Median :0.4900   Median :0.4900   Median :0.51   Median :0.2200  
 Mean   :0.5001   Mean   :0.4999   Mean   :0.50   Mean   :0.2612  
 3rd Qu.:0.5800   3rd Qu.:0.5700   3rd Qu.:0.55   3rd Qu.:0.3200  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.00   Max.   :1.0000  
      pox              vac              nuc        
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.4800   1st Qu.:0.2200  
 Median :0.0000   Median :0.5100   Median :0.2200  
 Mean   :0.0075   Mean   :0.4999   Mean   :0.2762  
 3rd Qu.:0.0000   3rd Qu.:0.5300   3rd Qu.:0.3000  
 Max.   :0.8300   Max.   :0.7300   Max.   :1.0000  
```

As you can see all the datapoints are between 0 and 1. Se the data already has been transformed with a min-max normalization.

Let's visualise this with ggplot. Using jitterpoints and a violing plot.

```
p1 <- ggplot(data, mapping = aes(x = "", y = mcg)) + geom_violin(alpha=0.2) +
  geom_jitter(width = 0.2, alpha = 0.25, height = 0, color = "red") + xlab(NULL)
p2 <- ggplot(data, mapping = aes(x = "", y = gvh)) + geom_violin(alpha=0.2) +
  geom_jitter(width = 0.2, alpha = 0.25, height = 0, color = "blue") + xlab(NULL)
p3 <- ggplot(data, mapping = aes(x = "", y = alm)) + geom_violin(alpha=0.2) +
  geom_jitter(width = 0.2,alpha = 0.25, height = 0, color = "purple") + xlab(NULL)
p4 <- ggplot(data, mapping = aes(x = "", y = mit)) + geom_violin(alpha=0.2) +
  geom_jitter(width = 0.2, alpha = 0.25, height = 0, color = "brown") + xlab(NULL)
p5 <- ggplot(data, mapping = aes(x = "", y = pox)) + geom_violin(alpha=0.2) +
  geom_jitter(width = 0.2,alpha = 0.25, height = 0, color = "orange") + xlab(NULL)
p6 <- ggplot(data, mapping = aes(x = "", y = vac)) + geom_violin(alpha=0.2) +
  geom_jitter(width = 0.2, alpha = 0.25, height = 0, color = "green") + xlab(NULL)
p7 <- ggplot(data, mapping = aes(x = "", y = nuc)) + geom_violin(alpha=0.2) +
  geom_jitter(width = 0.2,alpha = 0.25, height = 0, color = "orange") + xlab(NULL)

plot <- ggarrange(p1, p2, p3, p4, p5, p6, p7, nrow = 4, ncol = 2)
annotate_figure(plot, top = text_grob("Boxplots", face = "bold", size = 14))
```

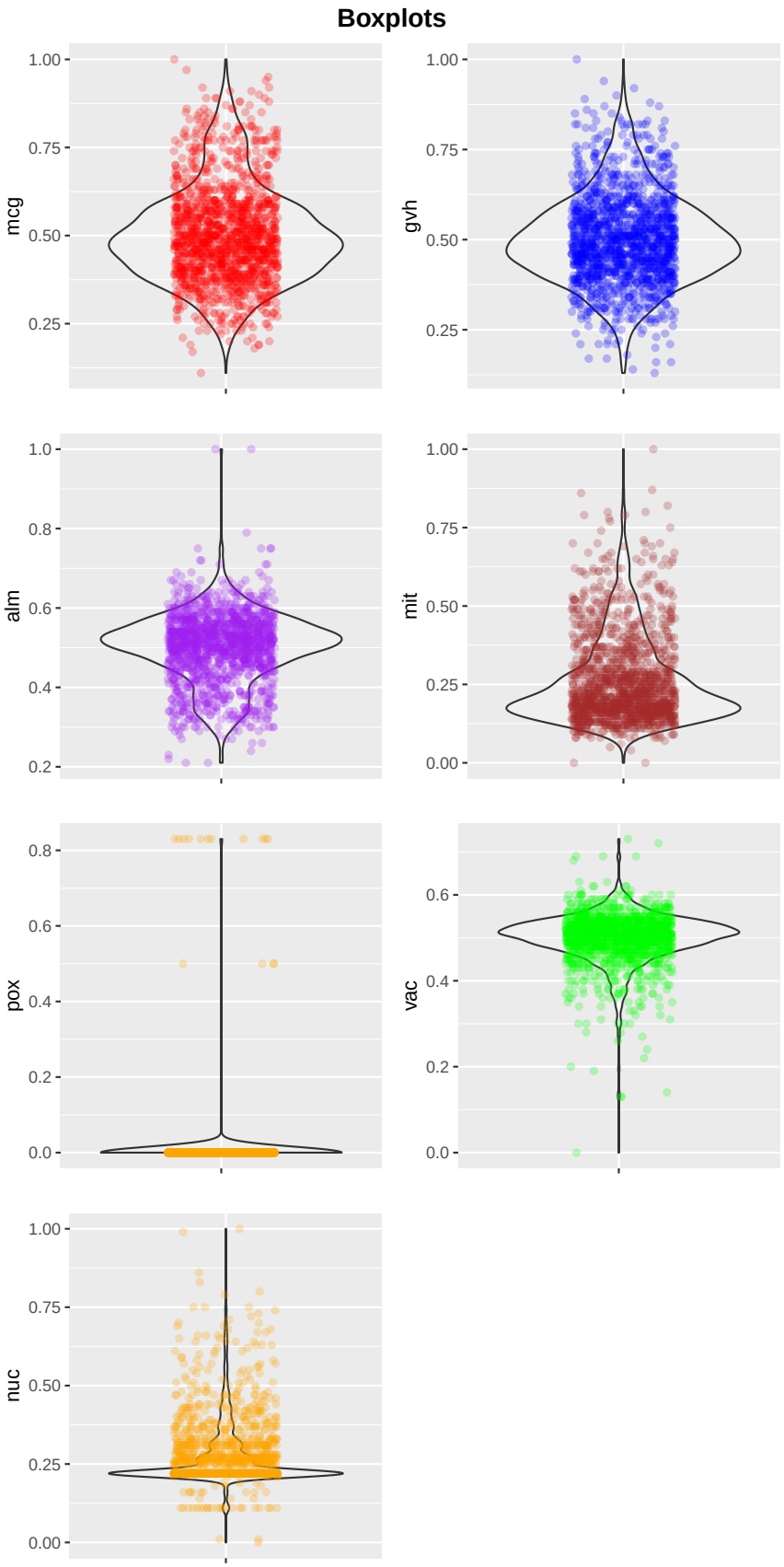

Figure 2: Boxplot comparing basic statistic for all columns

Mcg and gvh are normally distributed. Alm and mit are skewed but nothing crazy to worry about. Vac and nuc are really skewed however, not yet bad enough to worry about it. Pox is looks weird let's take a detailed look at pox.

```
table(data[, "pox"])
```

```
   0  0.5 0.83
1469    4   11
```

As you can see almost all the numbers are 0. This does not mean that we just discord this column. Maybe it is very significant if the number is not 0.

```
# Subset the data where pox is larger than 0
subset(data, pox > 0, select=loc.site)
```

```
# A tibble: 15 x 1
   loc.site
   <fct>
 1 CYT
 2 MIT
 3 POX
 4 POX
 5 POX
 6 POX
 7 POX
 8 POX
 9 POX
10 MIT
11 POX
12 MIT
13 POX
14 POX
15 POX
```

If the number is non-zero then the probability is very high that the protein is localised in peroxisomal region.

Bivariate Analysis

With a heatmap, you can easily see where there are correlations between variables. First let's create a correlation matrix

```
# Create a cor matrix
cor_matrix <- cor(data[, -c(5,9)])
cor_matrix <- as_tibble(cor_matrix)
# Add a column with the varibale names
(cor_matrix <- cor_matrix %>% mutate(varnames = all_of(attr_names)[-c(1,6,10)]))
```

```
# A tibble: 7 x 8
       mcg       gvh      alm      mit       pox     vac     nuc varnames
     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>   <dbl>   <dbl> <chr>
```

```
1  1        0.582    -0.164    0.158    0.00560   0.0750 -0.125  mcg
2  0.582    1        -0.272    0.140    0.000392  0.0888 -0.103  gvh
3 -0.164   -0.272     1        0.0597   0.00938  -0.186  -0.0220 alm
4  0.158    0.140     0.0597   1       -0.00904  -0.104  -0.0548 mit
5  0.00560  0.000392  0.00938 -0.00904  1         0.0209 -0.0357 pox
6  0.0750   0.0888   -0.186   -0.104    0.0209    1       0.0897 vac
7 -0.125   -0.103    -0.0220  -0.0548  -0.0357    0.0897  1      nuc
```

The calculated correlation matrix. This needs to be tranformed to a long matrix.

```
# Create a long matrix
(cor_matrix_long <- pivot_longer(data = cor_matrix,
                                 cols = all_of(attr_names)[-c(1,6,10)],
                                 names_to = "variable", values_to = "cor"))
```

```
# A tibble: 49 x 3
   varnames variable      cor
   <chr>    <chr>       <dbl>
 1 mcg      mcg       1
 2 mcg      gvh       0.582
 3 mcg      alm      -0.164
 4 mcg      mit       0.158
 5 mcg      pox       0.00560
 6 mcg      vac       0.0750
 7 mcg      nuc      -0.125
 8 gvh      mcg       0.582
 9 gvh      gvh       1
10 gvh      alm      -0.272
# ... with 39 more rows
```

The long calculated correlation matrix.

```
ggplot(data = cor_matrix_long, aes(x=varnames, y=variable, fill=cor)) +
    geom_tile() +
    labs(x=NULL, y=NULL, title="Heatmap Correlation") +
    scale_fill_gradient(high = "purple", low = "white" )
```

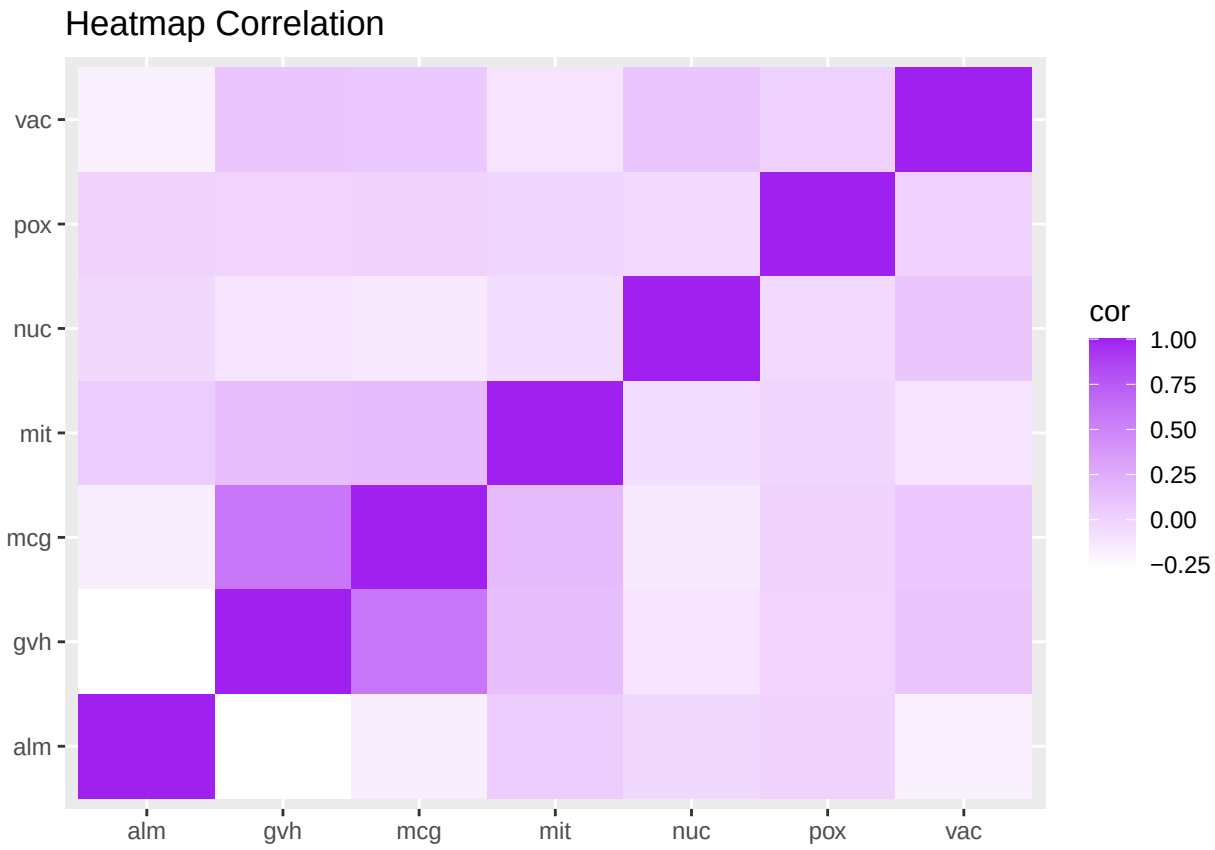

Figure 3: A heatmap pairwise correlation of selected numeric variables

As seen from the heatmap there is some correlation between mcg and gvh. Let's visualise this.

```
ggplot(data, aes(x=mcg, y=gvh, color=loc.site)) +
        labs(x="MCG", y="GVH") +
    geom_jitter(mapping = aes(color=loc.site),
                na.rm=T, width=0.2, height=0.2,
                alpha=0.5, shape=16, size=0.8) +
    ylim(0,1) + labs(title="Scatterplot and trendline") +
    geom_smooth(formula = y ~ x, method = "loess")
```

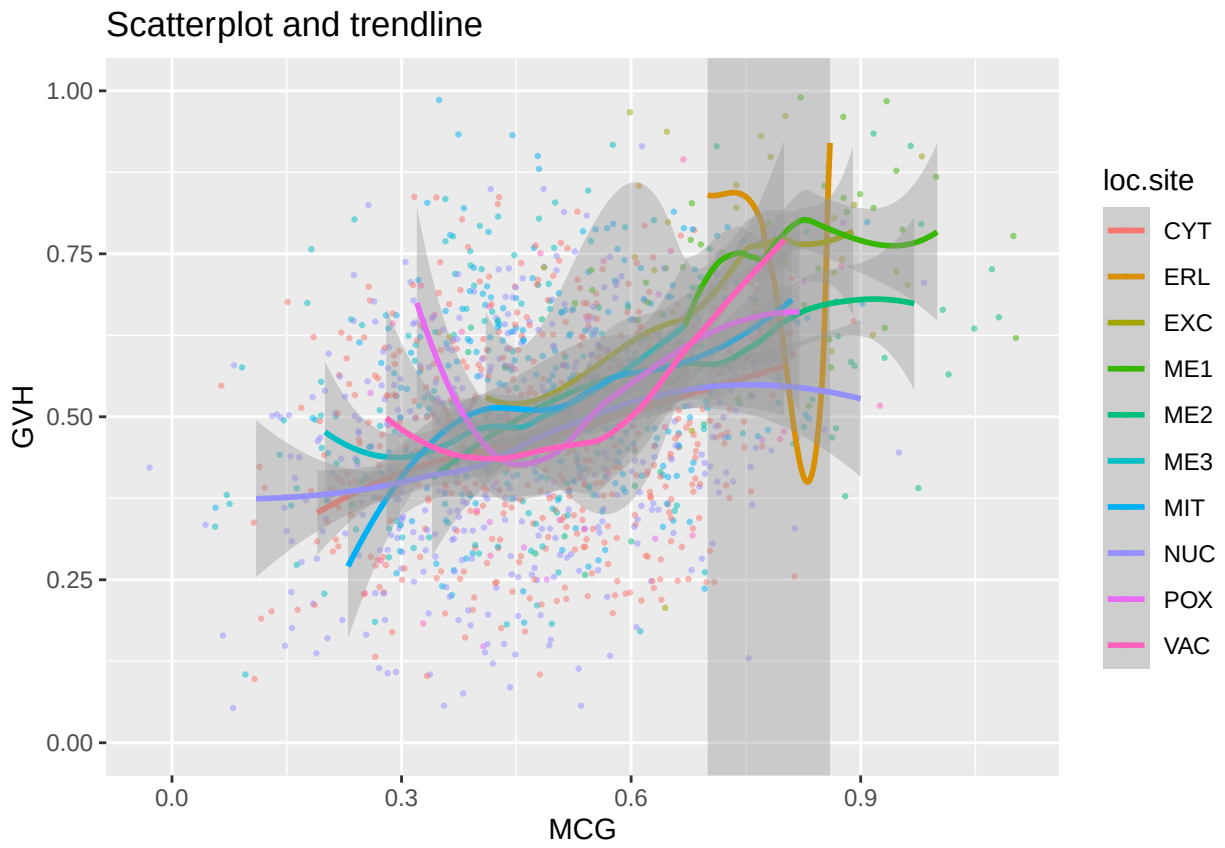

Figure 4: Scatterplot with trendline with the dependent variables

Every variable goes with a slow upward trend except erl this one has a weird curve.

Class labels

Now we need to look at how the data correlates with the classes. The heigth of the peak doesn't matter much, shifted peaks do.

```
p1 <- ggplot(data, aes(x=mcg)) + geom_density(aes(color=loc.site))
p2 <- ggplot(data, aes(x=gvh)) + geom_density(aes(color=loc.site))
p3 <- ggplot(data, aes(x=alm)) + geom_density(aes(color=loc.site))
p4 <- ggplot(data, aes(x=mit)) + geom_density(aes(color=loc.site))
p5 <- ggplot(data, aes(x=pox)) + geom_density(aes(color=loc.site))
```

```
p6 <- ggplot(data, aes(x=vac)) + geom_density(aes(color=loc.site))
p7 <- ggplot(data, aes(x=nuc)) + geom_density(aes(color=loc.site))
plot <- ggarrange(p1, p2, p3, p4, p5, p6, p7, ncol=4, nrow=2,
          common.legend=TRUE, legend="right")
annotate_figure(plot, top = text_grob("Density plots", face = "bold", size = 14))
```

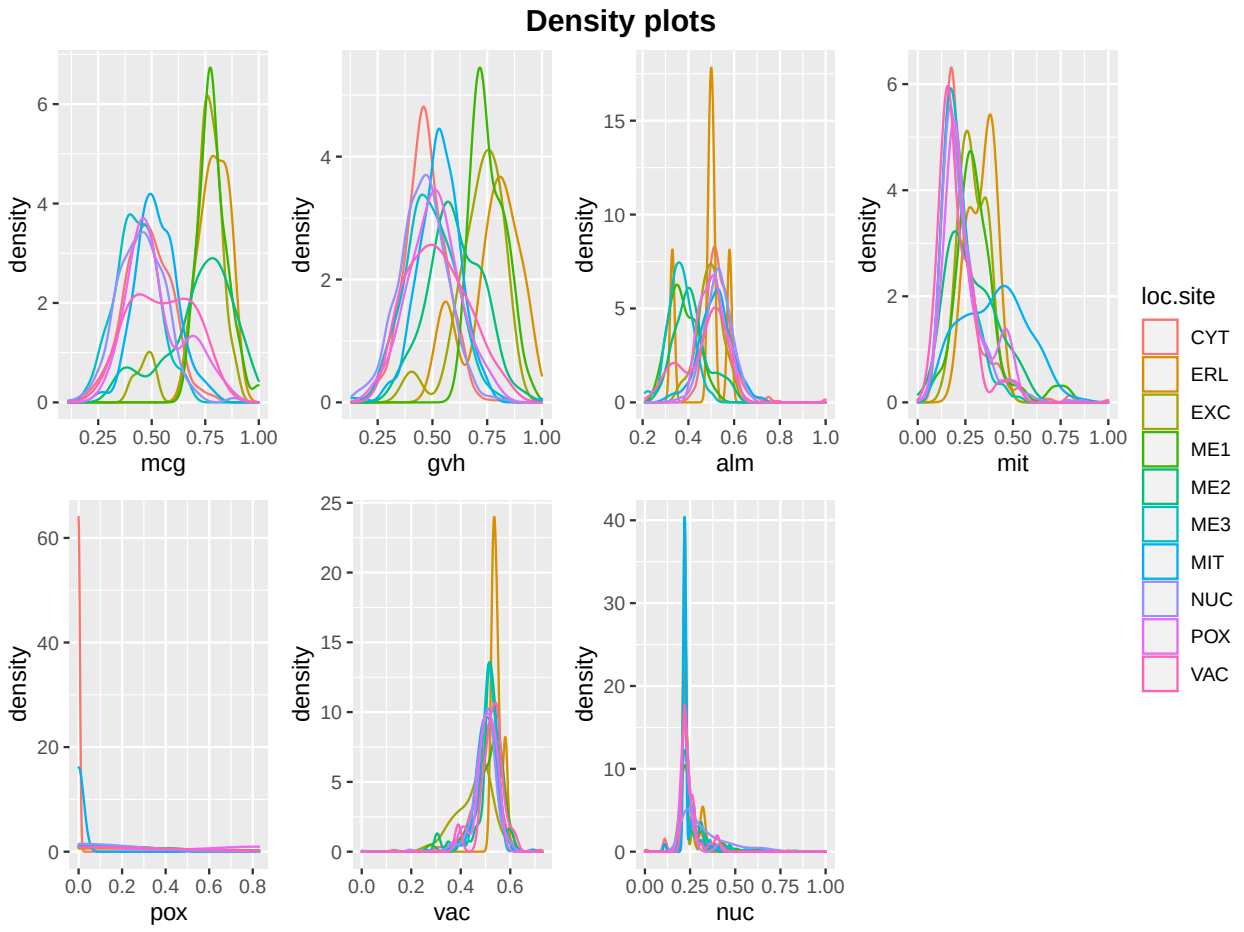

Figure 5: Density plots shows class distinction

At pox, vac and nuc, you don't see shifted peaks. At mcg and gvh you really see shifted peaks this shows a distribution of the different classes. There are small shifted peaks at alm en mit, this shows that there is difference but not so much. Let's look at mcg and gvh using ridge lines plot.

```
p1 <- ggplot(data, aes(x=mcg, y=loc.site, fill = loc.site)) +
  geom_density_ridges2(rel_min_height = 0.005) + theme_minimal() +
  coord_cartesian(clip = "off")
p2 <- ggplot(data, aes(x=gvh, y=loc.site, fill = loc.site)) +
  geom_density_ridges2(rel_min_height = 0.005) + theme_minimal() +
  coord_cartesian(clip = "off")
plot <- ggarrange(p1, p2, common.legend = TRUE)
annotate_figure(plot, top = text_grob("Ridge line plot", face = "bold", size = 14))
```

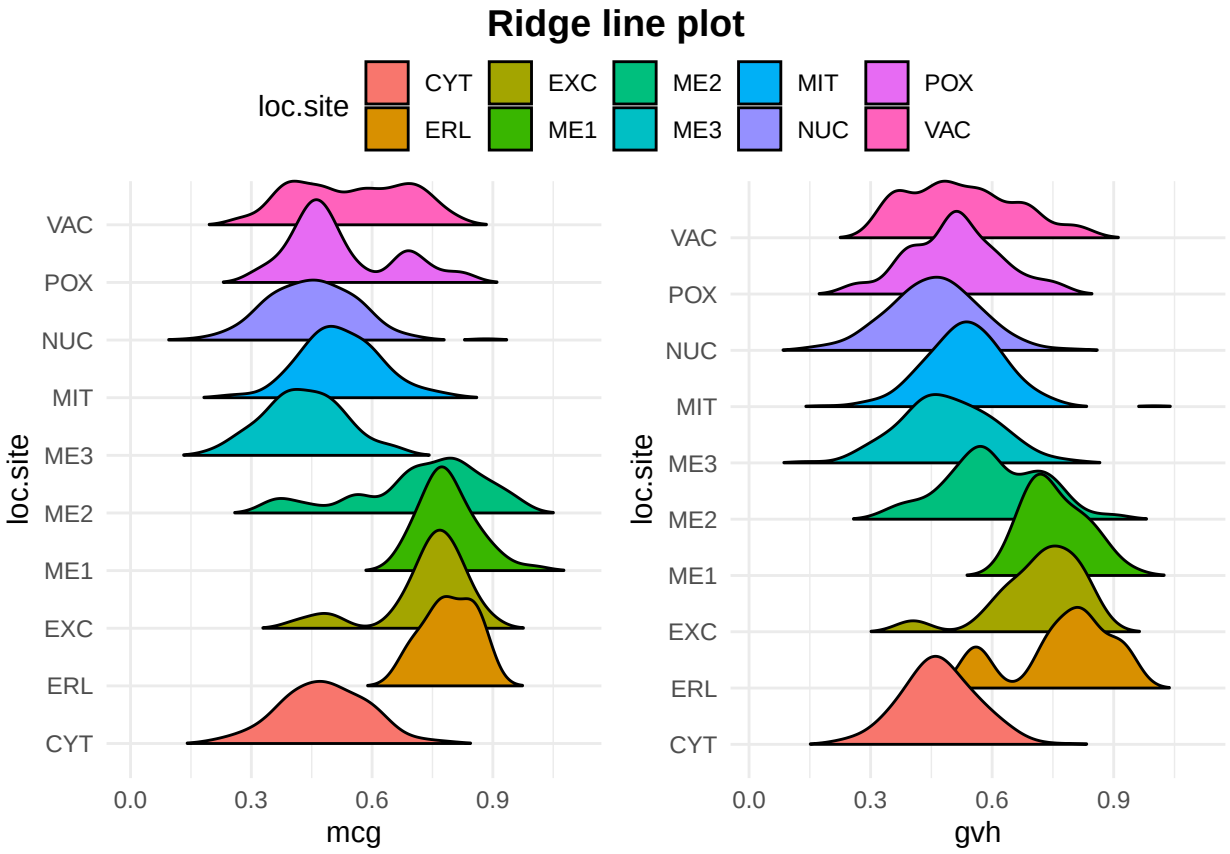

Figure 6: Ridge line plot between mcg and gvh

Now you can really see that the peaks are shifted.

We need to test the data to see if there is significant difference between it. Using a 1-way ANOVA test.

```
# Perform a one way anova test
res.aov <- summary(aov(mcg ~ loc.site, data = data))
res.aov[[1]]$`Pr(>F)`[1]
```

```
[1] 1.203676e-146
```

The P-value is < 0.05. So there is a significant difference.

Multivariate analysis

One way to see if groups are clustered is to MDS plot the groups and calculated the distance matrix.

```
# Create a matrix
matrix <- with(data, rbind(mcg, gvh, alm, mit, pox, vac, nuc))
(distmat <- dist(matrix))
```

```
          mcg       gvh       alm       mit       pox       vac
gvh  4.623700
alm  6.699455  6.524822
mit 11.476977 11.321051 11.025910
pox 19.910020 19.776585 19.479594 11.495734
vac  5.580672  5.083611  4.342165 10.946100 19.312255
nuc 11.161711 10.858508 10.144550  6.884693 11.545921  9.715282
```

Here is the distance matrix.

```
autoplot(cmdscale(distmat, eig = TRUE), shape = FALSE, label = TRUE, label.size = 4)
```

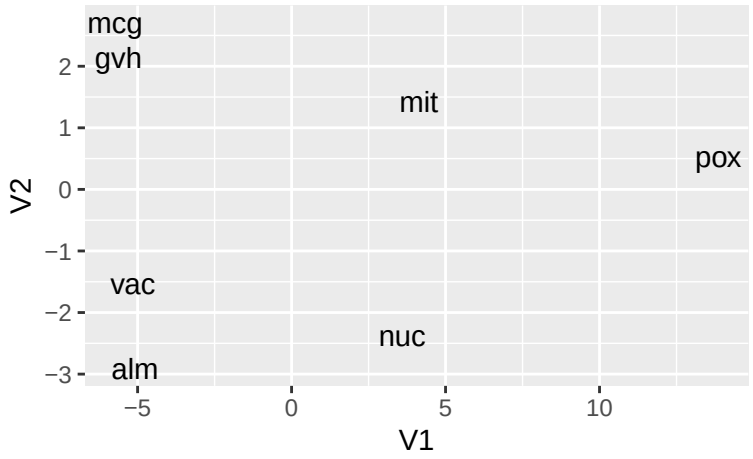

Figure 7: Classical (Metric) Multidimensional Scaling

As seen from above mcg and gvh are the clustered group.

A other plot to show is using PCA.

```r
df <- data[-c(5,9)]

pca_res <- prcomp(df, scale. = TRUE)

autoplot(pca_res, data = data, colour = 'loc.site', loadings = TRUE, loadings.label = TRUE)
```

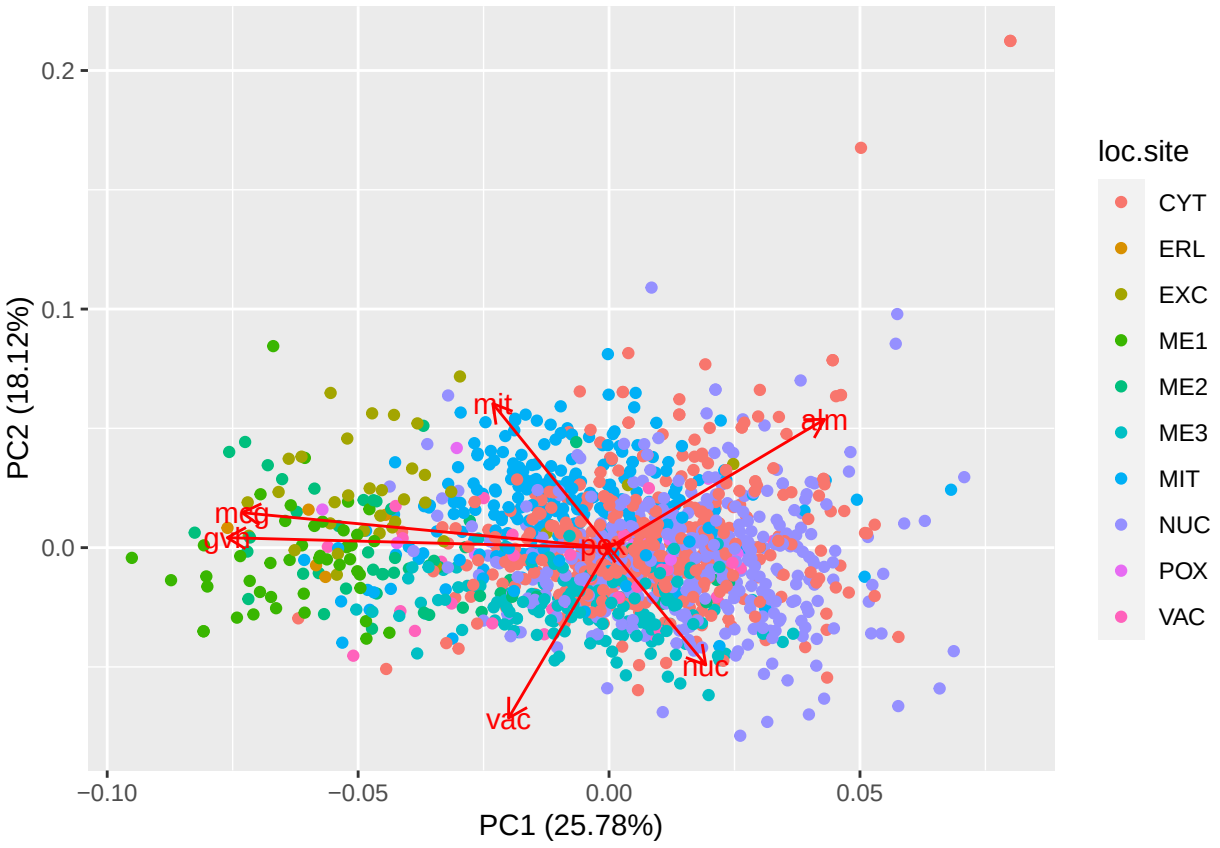

Figure 8: PCA plot showing the groups

As earlier shown mcg and gvh are grouped together.