# Clastrowanie genów - wizualizacja genów o podobnym profilu ekspresji

*Bartosz Kozak*

*26 10 2019*

## Normalizacja TPS

- Dzielimy ilość odczytów przez długość transkryptu (kpz, nie pz).
- Obliczamy sumę transkryptów dla każdej biblioteki - próbki.
- Dzielimy liczbę transkryptów przez sumę transkryptów uzyskaną dla biblioteki.

Analizy wykonamy dla odczytów z pliku counts.txt. Chcemy wytypować czynniki transkrypcyjne, które są charakterystyczne (ulegają ekspresji) dla określonej tkance - kwiat.

### Wczytanie danych

```
dane_RNAseq <- read.csv("counts.txt", sep = "\t", skip = 1)
dim(dane_RNAseq)
```

```
## [1] 37834     9
```

### Normalizacja biblioteki

**Ćwiczenie 1:**

Znormalizować biblioteki dla tkanek: liść, kwiat, pęd za pomocą algorytmu TPM. Dane w pliku `counts.txt`.

- utworzyć wektory `liść_TPM, ped_TPM, kwiat_TPM`
- utworzyć tabele `dane_TPM`.

```
dim(dane_TPM)
```

```
## [1] 37834     4
```

```
colnames(dane_TPM)[1:3] <- c("liść_TPM", "pęd_TPM", "kwiat_TPM")
head(dane_TPM)
```

```
##     liść_TPM   pęd_TPM kwiat_TPM        GenID
## 1 11.8751036 5.7486431 1.2743461 LOC109343272
## 2  0.6243926 0.8898839 0.4307470 LOC109343320
## 3  0.0000000 0.0000000 0.0000000 LOC109343262
## 4  0.0000000 0.0000000 0.0000000 LOC109343339
## 5  0.0000000 0.0000000 0.1775501 LOC109343296
## 6  0.0000000 0.1137621 5.4638060 LOC109343328
```

Z znormalizowanej biblioteki wybieramy transkrypty czynników transkrypcyjnych MYB. Gdzie można znaleźć dane odnośnie interesujących nas genów?

- NCBI
- ENSEMBL

### Wczytanie nazw genów

Korzystamy z pliku `MYB.txt`

```
myb <- read.csv("MYB.txt", sep = "\t")
myb_list <- myb$Locus
```

### Ćwiczenie 2

Przefiltrować dane `dane_TPM` wybierając tylko geny z grupy MYB (na podstawie wektora `myb_list`)

- utworzyć tabele `dane_myb` z informacją o ekspresją genów MYB w tkankach: liść, pęd, kwiat.

```
colnames(dane_myb) <- c("liść", "pęd", "kwiat", "GenID")
head(dane_myb)
```

```
##            liść         pęd        kwiat        GenID
## 379   0.0000000   0.0000000   0.00000000 LOC109346957
## 1553 81.6047414 169.6305964 134.93608009 LOC109357172
## 1581  0.6250208   0.5688484   0.08942999 LOC109362577
## 1600  0.0000000   0.1707920   1.04717370 LOC109357663
## 2196  0.0000000   0.0000000   2.11521846 LOC109362532
## 2202  0.5220762   0.1218348   0.12450058 LOC109339520
```

```
dim(dane_myb)
```

```
## [1] 111   4
```

## Standaryzacja danych, globalnie czy w obrębie genu?

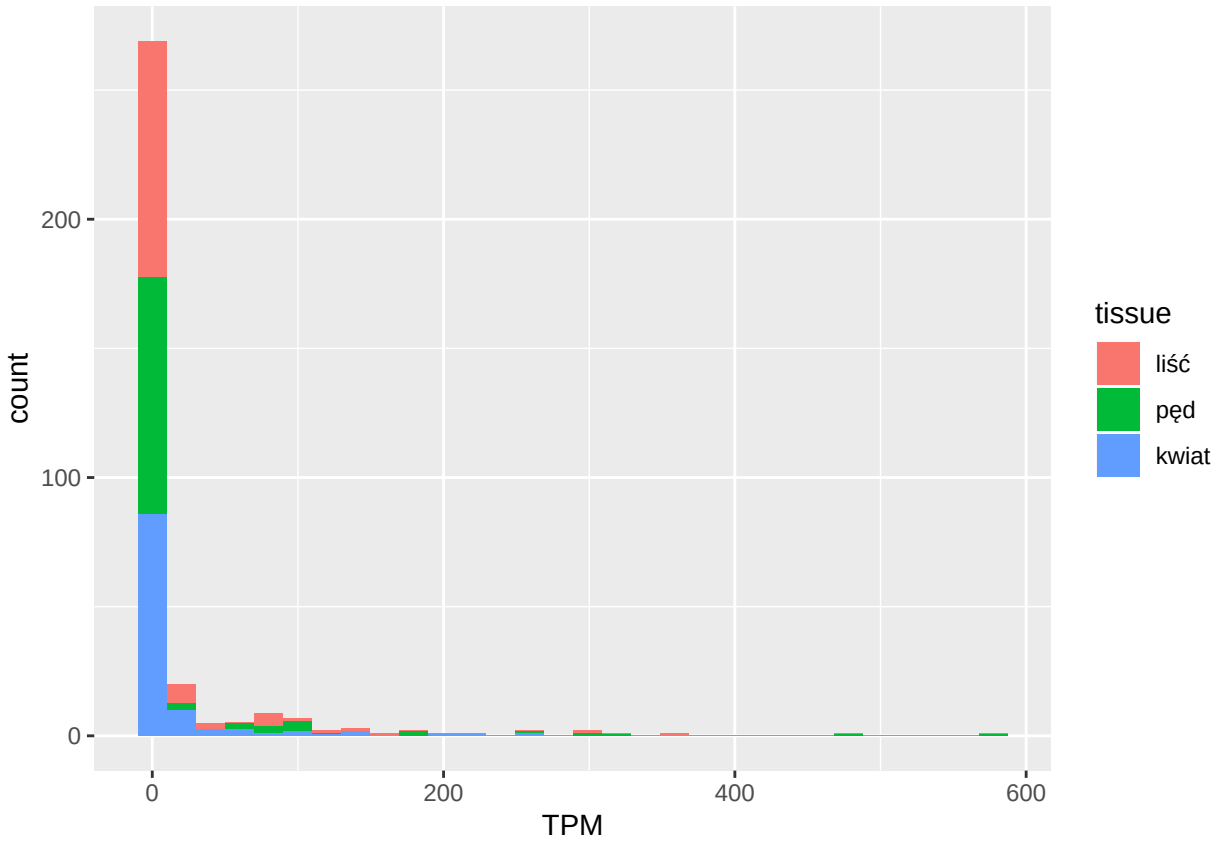
Zanim odpowiemy na to pytanie zobaczmy w jakim zakresie (ilość transkryptów) są nasze dane w poszczególnych bibliotekach (tkankach).

```
dane_myb_s <- stack(dane_myb)
```

```
## Warning in stack.data.frame(dane_myb): non-vector columns will be ignored
```

```
colnames(dane_myb_s) <- c("TPM","tissue")
library(ggplot2)
ggplot(dane_myb_s, aes(x = TPM, fill = tissue))+geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




Do standaryzacji danych wykorzystamy funkcję R scale()

```
dane_myb_s <- scale(t(as.matrix(dane_myb[,c(1:3)])))
dane_myb_s <- t(dane_myb_s)
row.names(dane_myb_s) <- dane_myb$GenID
dane_myb_s1 <- na.omit(dane_myb_s)
```

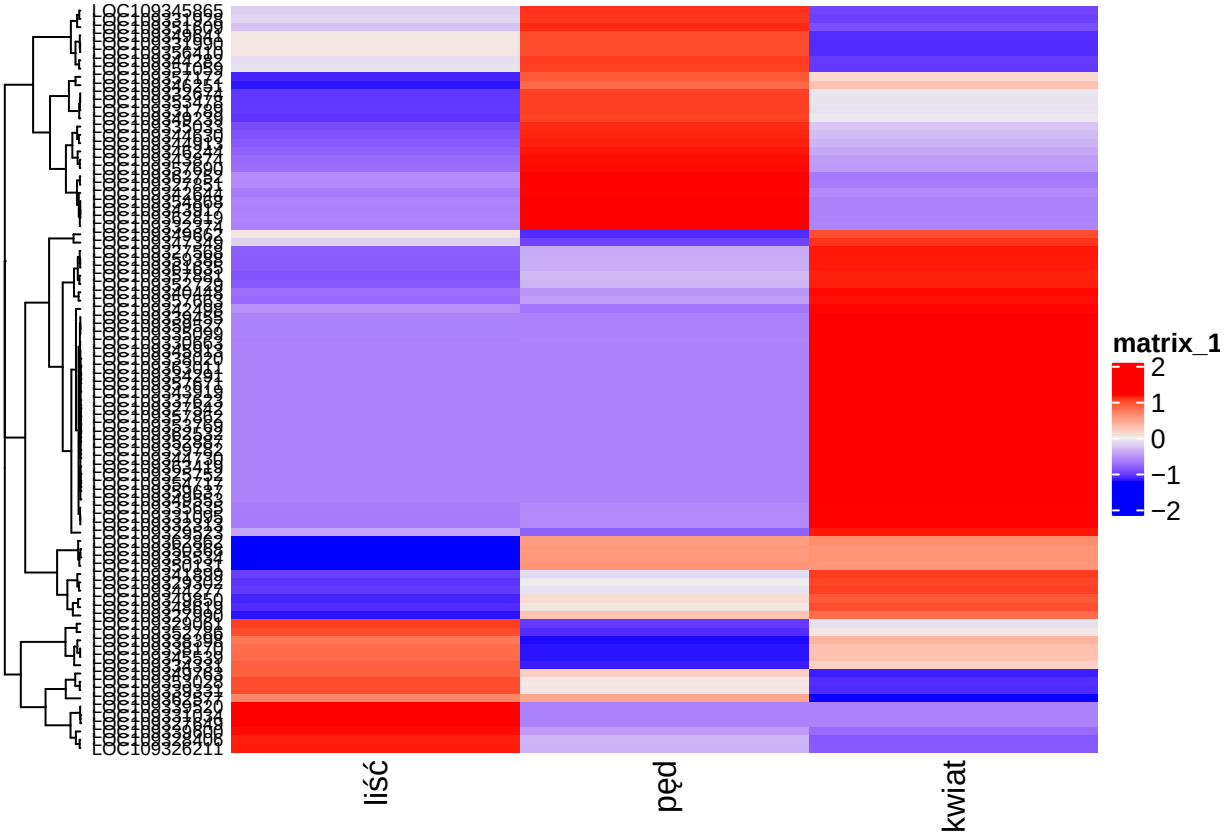
**Tworzymy heatmape**

```
library(ComplexHeatmap)
```

```
## Loading required package: grid
```

```
## ========================================
## ComplexHeatmap version 2.0.0
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite:
## Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##   genomic data. Bioinformatics 2016.
## ========================================
```

```
Heatmap(dane_myb_s1, cluster_columns=FALSE,
        row_names_side = "left",
        row_dend_sid = "left",
        row_names_gp=gpar(cex=0.6))
```




## Poszukiwanie czynników transkrypcyjnych MYB o zbliżonym profilu ekspresji do genów CEN (CEN-like 2 LOC109350801)

W pierwszej kolejności musimy z naszych danych RNAseq wybrać transkrypty odpowiadające genom MYB oraz CEN (plik `CEN.txt`).

### Ćwiczenie 3

Przefiltrować dane z tabeli `dane_TPM`, aby uzyskać tabele z genami CEN-like oraz MYB.

- Utworzyć tabelę `dane_myb` z informacją o ekspresją genów MYB oraz CEN w tkankach: liść, pęd, kwiat.

```
dim(dane_myb)
```

```
## [1] 114   4
```

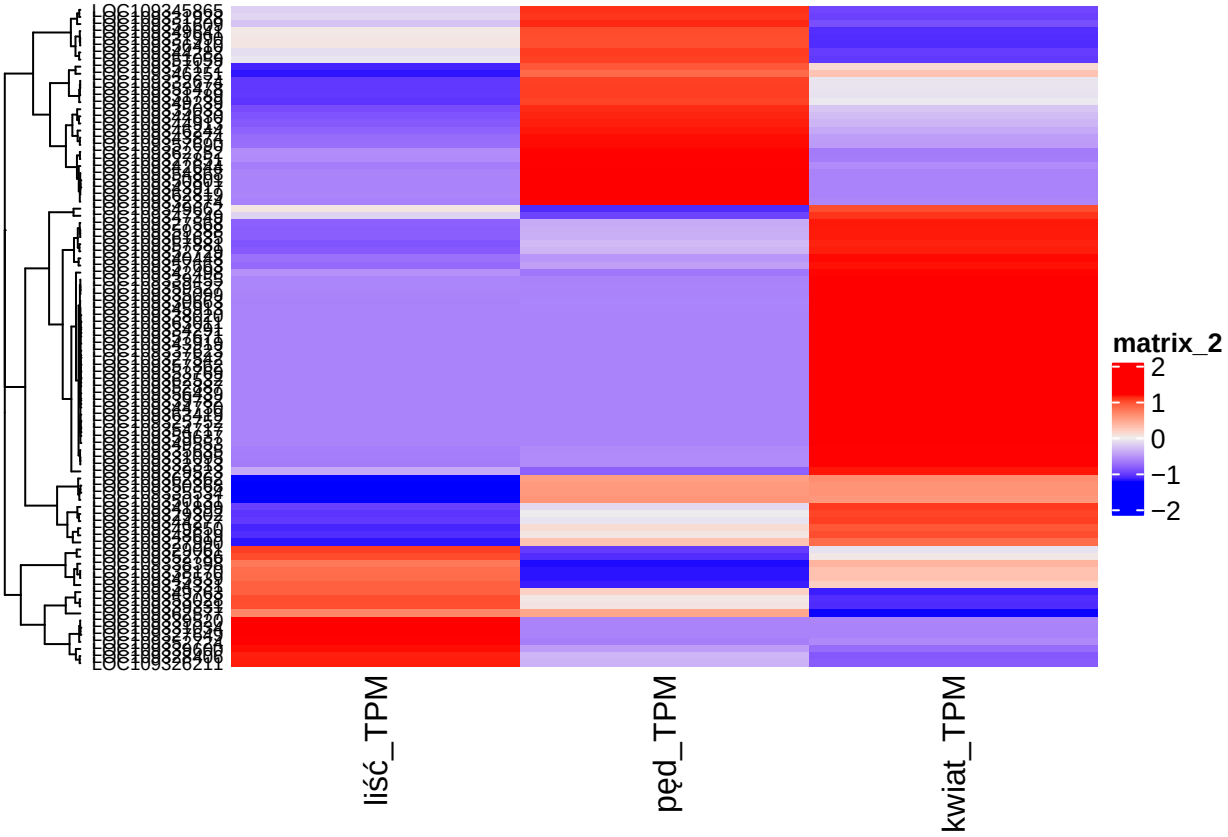
Następnie możemy stworzyć kolejną heet mapę.

### Ponowne skalowanie danych

Musimy powtórzyć skalowanie danych, bo mamy inne dane w obiekcie dane_myb.

```
dane_myb_s <- scale(t(as.matrix(dane_myb[,c(1:3)])))
dane_myb_s <- t(dane_myb_s)
row.names(dane_myb_s) <- dane_myb$GenID
dane_myb_s1 <- na.omit(dane_myb_s)
Heatmap(dane_myb_s1, cluster_columns=FALSE,
        row_names_side = "left",
```

```
        row_dend_sid = "left",
        row_names_gp=gpar(cex=0.6))
```




**Jak możemy znaleźć czynniki MYB o najbardziej zbliżonym profilu ekspresji do naszego genu zainteresowania (LOC109350801)?**

Do tego celu wykorzystamy dystans Euklidesowy (lub innego).

```
library(reshape2)
dyst <- dist(dane_myb_s1)
temp<-t(as.matrix(dyst))
temp<-temp[,ncol(temp):1]
dyst_m <- melt(temp)
dyst_m_myb <- dyst_m[dyst_m$Var1=="LOC109350801",]
summary(dyst_m_myb$value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.7834  2.2765  1.7492  2.4495  2.8165
```

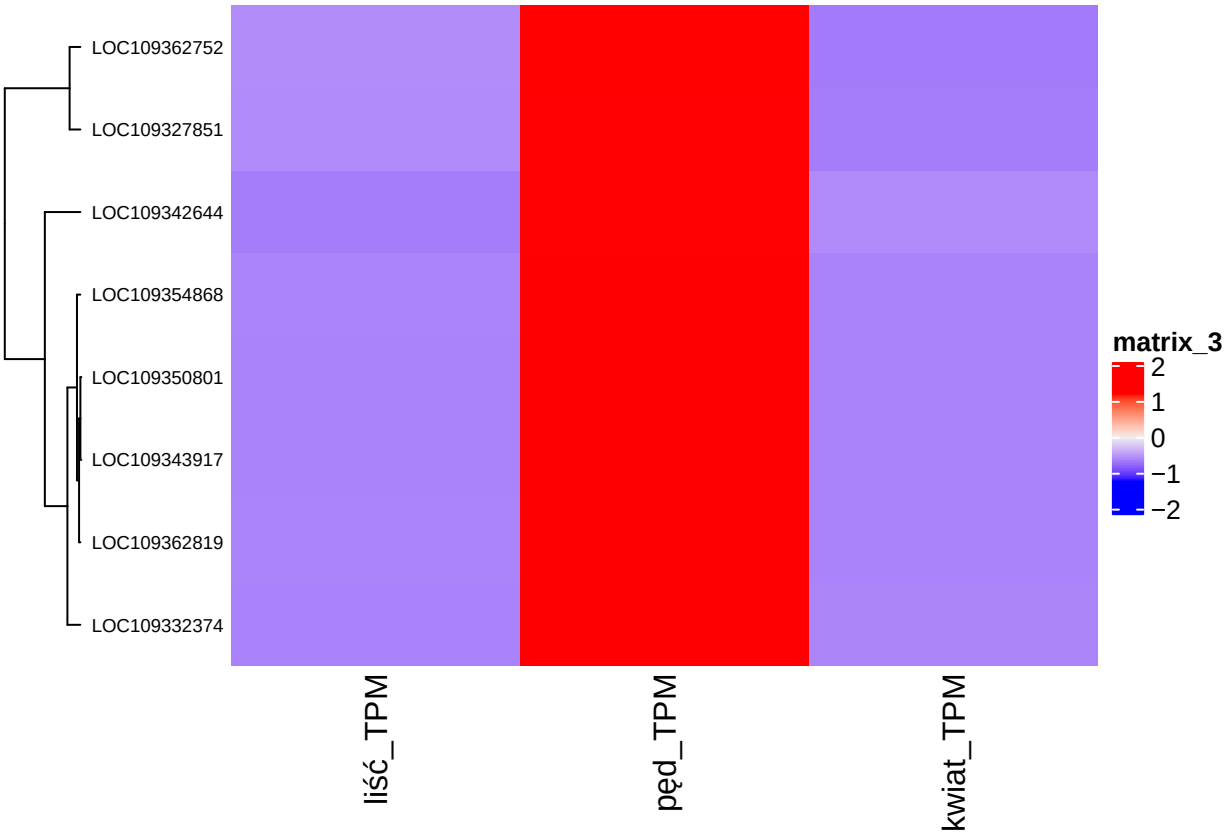
```
dyst_m_myb_b <- dyst_m_myb[dyst_m_myb$value<0.1,]
dyst_m_myb_b
```

```
##              Var1         Var2        value
## 1431 LOC109350801 LOC109332374 1.623732e-02
## 2547 LOC109350801 LOC109327851 5.544118e-02
## 3291 LOC109350801 LOC109362752 7.365312e-02
## 4500 LOC109350801 LOC109354868 5.750113e-03
## 5337 LOC109350801 LOC109350801 0.000000e+00
```

```
## 7104 LOC109350801 LOC109342644 5.444172e-02
## 7476 LOC109350801 LOC109343917 2.220446e-16
## 8034 LOC109350801 LOC109362819 2.174335e-03
```

Możemy stworzyć nową heat mapę dla wybranych genów.

```
myb_list <- as.vector(dyst_m_myb_b$Var2)
dane_myb <- dane_TPM[dane_TPM$GenID %in% myb_list,]
dane_myb_s <- scale(t(as.matrix(dane_myb[,c(1:3)])))
dane_myb_s <- t(dane_myb_s)
row.names(dane_myb_s) <- dane_myb$GenID
dane_myb_s1 <- na.omit(dane_myb_s)
Heatmap(dane_myb_s1, cluster_columns=FALSE,
        row_names_side = "left",
        row_dend_sid = "left",
        row_names_gp=gpar(cex=0.6))
```




## Korelacja.

- wracamy do danych wyjściowych - wszystkie geny MYB
- ponownie skalujemy dane

```
library(Hmisc)
```

```
## Loading required package: lattice
## Loading required package: survival
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
myb_list <- myb$Locus
dane_myb <- dane_TPM[dane_TPM$GenID %in% myb_list,]
dane_myb_s <- scale(t(as.matrix(dane_myb[,c(1:3)])))
dane_myb_s <- t(dane_myb_s)
row.names(dane_myb_s) <- dane_myb$GenID
dane_myb_s1 <- na.omit(dane_myb_s)
dane_myb_s1_t <- t(dane_myb_s1)
cor_dane_myb <- cor(dane_myb_s1_t)
```

Tworzymy pomocniczą funkcję, która pozwoli nam odpowiednio uszeregować nasze obiekty.

```
reorder_cormat <- function(cormat){
  dd <- as.dist((1-cormat)/2)
  hc <- hclust(dd)
  cormat<-cormat[hc$order, hc$order]
}
```

Ustawiamy dane w odpowiedniej kolejności.

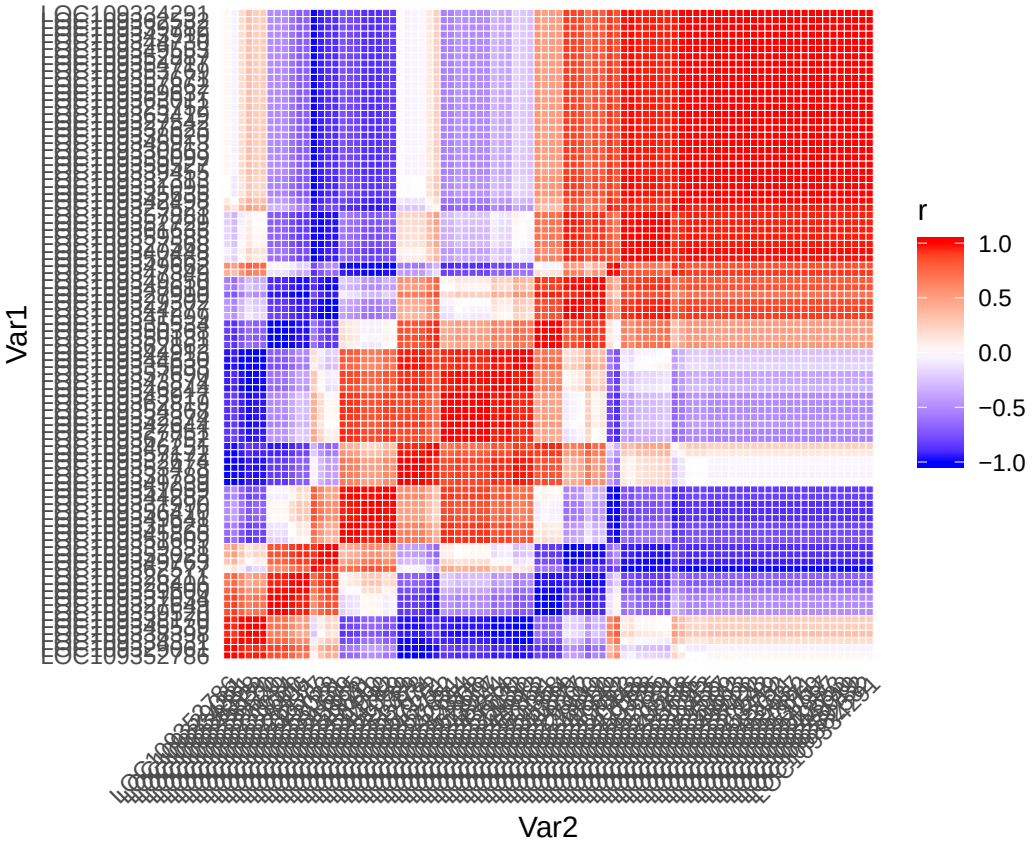
```
cor_dane_myb <- reorder_cormat(cor_dane_myb)
acol <- colnames(cor_dane_myb)

dane_myb_s1_t <- as.data.frame(dane_myb_s1_t)
dane_myb_s1_t <- dane_myb_s1_t[acol]
cor_mp <- cor(as.matrix(dane_myb_s1_t))
cor_m<-cor_mp
```

Tworzymy nową tabelę dla funkcji gg.

```
melted_cormat <- melt(cor_m)
melted_cormat[is.na(melted_cormat)] <- ""
```

Tworzymy obraz z wykorzystaniem funkcji `ggplot`.

```
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name=expression(r)) +
  theme_minimal()+ # minimal theme
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))+
  coord_fixed()
print(ggheatmap)
```


Var1
Var2
r
1.0
0.5
0.0
−0.5
−1.0