Zadanie 4 - Heatmapa

Ksenia Kvitko

3 04 2020

1. Dane z etapu końca zadania 3

```
# Zadanie 1 -- start
raw_data <- read.csv("..\\source_files\\counts.txt", sep = "\t", skip = 1)
geneLengths <- raw_data[, c(1, 6:9)]
TPM_step1 <- geneLengths
TPM_step1$bam.flower.bam <- TPM_step1$bam.flower.bam / TPM_step1$Length
TPM_step1$bam.stem.bam <- TPM_step1$bam.stem.bam / TPM_step1$Length
TPM_step1$bam.leaf.bam <- TPM_step1$bam.leaf.bam / TPM_step1$Length
TPM_step2 <- TPM_step1
TPM_step2$bam.flower.bam <- TPM_step2$bam.flower.bam / (sum(TPM_step2$bam.flower.bam) / 1000000)
TPM_step2$bam.stem.bam <- TPM_step2$bam.stem.bam / (sum(TPM_step2$bam.stem.bam) / 1000000)
TPM_step2$bam.leaf.bam <- TPM_step2$bam.leaf.bam / (sum(TPM_step2$bam.leaf.bam) / 1000000)
dane_TPM <- TPM_step2[,c(1,3:5)]
colnames(dane_TPM)[2:4] <- c("liść_TPM", "pęd_TPM", "kwiat_TPM")
# Zadanie 1 -- koniec

# Zadanie 2 -- start
MYB <- read.csv("..\\source_files\\MYB.txt", sep = "\t")
library(dplyr)
dane_myb <- dane_TPM %>% filter(Geneid %in% MYB$Locus)
# Zadanie 2 -- koniec

# Zadanie 3 -- start
MYB_Zad3 <- read.csv("..\\source_files\\MYB_Zad3.txt", sep = "\t")
CEN <- read.csv("..\\source_files\\CEN.txt", sep = "\t")
dane_myb <- dane_TPM %>% filter(Geneid %in% MYB_Zad3$Locus)
dane_cen <- dane_TPM %>% filter(Geneid %in% CEN$Locus)
dane_myb_OR_cen <- rbind(dane_myb, dane_cen)
dane_myb <- dane_myb_OR_cen
# Zadanie 3 -- koniec
```

2. Instalacja niezbędnych bibliotek

```
# Instalacja ComplexHeatmap - potrzebny Bioconductor
# https://www.bioconductor.org/install/
```

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install(version = "3.10")

BiocManager::install(c("GenomicFeatures", "AnnotationDbi"))
BiocManager::install(c("ComplexHeatmap"))
```

```
library(ComplexHeatmap)
```

3. Rozwiązanie zadania

Przygotowanie danych

Wykorzystując skalowanie danych za pomocą funkcji log2()

```
dim(dane_myb)
```

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

```
dane_myb_s <- log2(t(as.matrix(dane_myb[,c(2:4)])))
dane_myb_s <- t(dane_myb_s)
row.names(dane_myb_s) <- dane_myb$GenID
dane_myb_s1 <- na.omit(dane_myb_s)
```

Utworzenie heatmapy

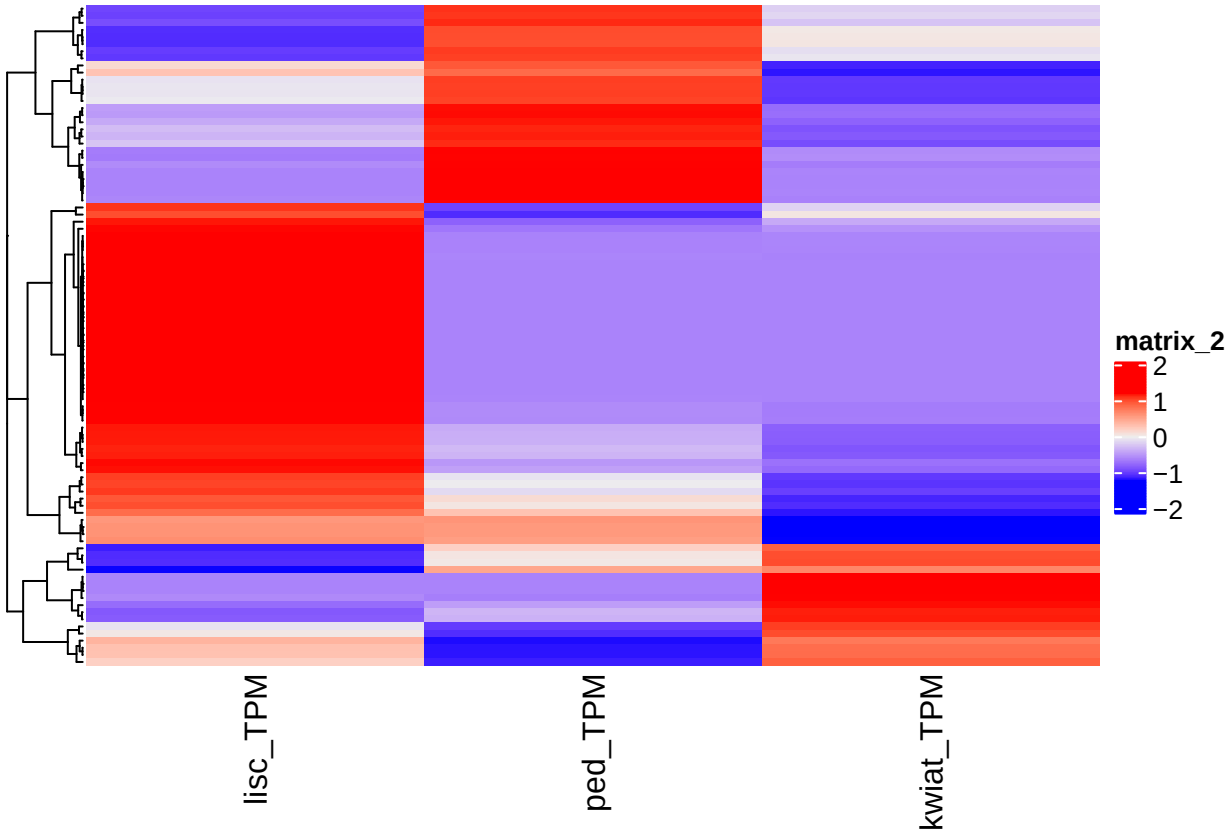
```
Heatmap(dane_myb_s1, cluster_columns=FALSE,
        row_names_side = "left",
        row_dend_sid = "left",
        row_names_gp=gpar(cex=0.6))
```

```
## Error in grid.pretty(range(breaks)): nieskończone długości osi '[GEPretty(-inf,inf,5)]'
```

Komenda nie działa po zmianie skalowania danych z funkcji scale() na log2()

Dla porównania - komenda ze skalowaniem funkcją scale() - jak w instrukcji do ćwiczeń

```
dane_myb_s <- scale(t(as.matrix(dane_myb[,c(2:4)])))
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))
```



Znaleziona na stronie pomocy Bioconductor'a informacja sugeruje, że problemem są liczby nieskończone https://support.bioconductor.org/p/57354/

```
# Przywrócenie zmiennych przeskalowanych za pomocą funkcji log2() dla potrzeb dalszych działań
dane_myb_s <- log2(t(as.matrix(dane_myb[,c(2:4)])))
dane_myb_s <- t(dane_myb_s)
row.names(dane_myb_s) <- dane_myb$GenID
dane_myb_s1 <- na.omit(dane_myb_s)
```

Przetworzona tabela rzeczywiście zawiera dane nieskończone (jak zasugerowano na forum Bioconductor'a)

```
sum(!is.finite(dane_myb_s1))
```

```
## [1] 133
```

Spróbowano usunąć dane nieskończone

```
dane_myb_s1_finite_only <- dane_myb_s1
dim(dane_myb_s1_finite_only)
```

```
## [1] 114   3
```

```
dane_myb_s1_finite_only[!is.finite(dane_myb_s1_finite_only)] <- NA
dane_myb_s1_finite_only <- na.omit(dane_myb_s1_finite_only)
dim(dane_myb_s1_finite_only)
```

```
## [1] 46  3
```

```
sum(!is.finite(dane_myb_s1_finite_only))
```

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

Po pozostawieniu jedynie liczb skończonych funkcja Heatmap() działa. Czy taki był cel zadania? Czy heatmapa wygenerowana w oparciu o ograniczoną ilość danych jest równie wartościowa?

```
Heatmap(dane_myb_s1_finite_only, cluster_columns=FALSE,
        row_names_side = "left",
        row_dend_sid = "left",
        row_names_gp=gpar(cex=0.6))
```