Analiza RNASeq dla bioprojektu PRJNA313294

Izabela Szymkowiak

Spis treści

1. Cel projektu 3
2. Pobieranie danych 3
3. Kontrola jakości oraz filtrowanie odczytów 3
4. Mapowanie odczytów 4
4.1. Pobieranie genomu hg19 5
4.2. Mapowanie odczytów 5
5. Zliczanie odczytów 6
6. Analiza DE 6
7. PCA oraz HeatMapa 7
8. Wnioski 12

1. Cel projektu

Celem projektu było wykonanie analizy RNAseq dla 8 bibliotek, w tym 4 zawierających odczyty typu pair end z urządzenia MiSeq oraz 4 zwierających odczyty single end z urządzenia NextSeq. Analizę przeprowadzono dla bioprojektu PRJNA313294.

	Pair-end	Single-end
MOCK	SRR3191542	SRR3194428
	SRR3191543	SRR3194429
ZIKV	SRR3191544	SRR3194430
	SRR3191545	SRR3194431

2. Pobieranie danych

Do pobrania danych z bazy ncbi utworzono następujący skrypt z wykorzystaniem narzędzia *Entrez Direct*:

```
#!/bin/bash

arrRun=( $(esearch -db sra -query PRJNA313294 | efetch -format
runinfo -mode xml | xtract -pattern SraRunInfo -element Run) )
arrType=( $(esearch -db sra -query PRJNA313294 | efetch -format
runinfo -mode xml | xtract -pattern SraRunInfo -element
LibraryLayout) )
for i in "${!arrType[@]}"
do
  if [[ "${arrType[i]}" == "PAIRED" ]]
   then
    echo "Pobieranie: ${arrRun[i]} typ: ${arrType[i]}"
    fastq-dump --split-files --gzip ${arrRun[i]}
  else
    echo "Pobieranie: ${arrRun[i]} typ: ${arrType[i]}"
    fastq-dump --gzip ${arrRun[i]}
  fi
done
```

Skrypt na początku pobiera informacje dla każdej z 8 bibliotek, czy znajdują się w niej odczyty typu „paired" czy „single" a następnie zapisuje tą informację w tablicy arrType. W następnym kroku, również dla 8 bibliotek, zapisuje numer Run w tablicy arrRun. Kolejno w pętli sprawdzany jest warunek, czy i-ta obserwacja jest typu paired, a jeśli ten warunek jest spełniony, odczyt musi zostać podzielony na lewy i prawy, poleceniem *–split-files*. W przeciwnym wypadku mamy do czynienia z odczytami typu „single" i dane zostają pobierane, bez podziału na lewy i prawy.

3. Kontrola jakości oraz filtrowanie odczytów

Kolejnym etapem było wykonanie analizy jakości dla każdej z 8 bibliotek. W tym celu wykorzystano program *FastQC*. Poniżej przedstawiono skrypt do tego służący:

```
#!/bin/bash

mkdir QC
for file in $(ls fastq-gzip/*)
 do
```

```
 echo "Kontrola i raport dla ${file}"
 fastqc ${file} --noextract --nogroup --outdir QC
done
```

W sposób wizualny oceniono wyniki oraz zauważono że, dane są dość dobrej jakości. Zdecydowano się zastosować jednak program *Trimmomatic* w celu poprawy ich jakości oraz ustawiono następujące parametry:

- SLIDINGWINDOW:4:20, co oznacza zastosowanie okna do skanowania odczytów o szerokości 4 oraz usunięcie odczytów, których jakość była mniejsza niż 20
- MINLEN:60, co oznacza zachowanie odczytów o długości minimum 60

Parametry zostały zastosowane dla odczytów typu pair-end oraz single-end

Skrypt z użyciem narzędzia trimmomatic wyglądają następująco:

i. pair-end

```
#!/bin/bash

for num in `seq 42 45`
do
  echo SRR31915$num
  java -jar /home/iza/miniconda3/share/trimmomatic-0.39-
1/trimmomatic.jar PE -phred33 /home/iza/Pulpit/fastq-
gzip/SRR31915${num}_1.fastq.gz /home/iza/Pulpit/fastq-
gzip/SRR31915${num}_2.fastq.gz /home/iza/Pulpit/fastq-
gzip/TRIMM/PAIRED/paired_1_SRR31915${num}.fq.gz
/home/iza/Pulpit/fastq-
gzip/TRIMM/PAIRED/unpaired_1_SRR31915${num}.fq.gz
/home/iza/Pulpit/fastq-
gzip/TRIMM/PAIRED/paired_2_SRR31915${num}.fq.gz
/home/iza/Pulpit/fastq-
gzip/TRIMM/PAIRED/unpaired_2_SRR31915${num}.fq.gz SLIDINGWINDOW:4:20
MINLEN:60
     done
```

ii. single-end

```
#!/bin/bash
for num in `seq 28 31`
 do
  echo SRR31944${RUN}
  java -jar /home/iza/miniconda3/share/trimmomatic-0.39-
1/trimmomatic.jar SE -phred33 /home/iza/Pulpit/fastq-
gzip/SRR31944${num}_1.fastq.gz /home/iza/Pulpit/fasq-
gzip/TRIMM/SINGLE/sample_ SRR31944${num}.fq.gz SLIDINGWINDOW:4:20
MINLEN:60
done
```

Następnie przeprowadzono ponownie analizę jakości za pomocą programu FastQ, która była zadowalająca.

4. Mapowanie odczytów

Kolejnym etapem analizy jest przeprowadzenie mapowania odczytów do genomu referencyjnego hg19.

4.1. Pobieranie genomu hg19

Do pobrania genomu wykorzystano następujący skrypt:

```
#!/bin/bash

wget --timestamping
'ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/*'
mkdir hg19

for chr in *.gz
 do
   gunzip -c $chr >  hg19/`echo $chr | sed s/.gz//`
 done
cd hg19
cat *.fa > hg19_ref_genome.fa
```

Wszystkie chromosomy zostały pobrane w osobnych plikach, które następnie zostały rozpakowane i połączone w jeden plika o rozszerzeniu fa. Kolejnym etapem było utworzenie indeksu dla genomu referencyjnego za pomocą polecenia hisat2-build. W ten sposób otrzymano 8 plików o rozszerzeniu ht2.

4.2. Mapowanie odczytów

Do mapowania odczytów zdecydowano się wykorzystać program hisat2. Poniżej przedstawiono skrypty, służące do tej części analizy:

i. pair-end

```
#!/bin/bash

for num in `seq 42 45`
 do
  echo SRR31915${num}

  hisat2 -q -x /home/iza/Pobrane/index/index -1
/home/iza/Pulpit/fastq-
gzip/TRIMM/PAIRED/paired_1_SRR31915${num}fg.gz -2
paired_2_SRR31915${num}.fg.gz -S wynik_SRR31915${num}.hisat.sam
  done
```

ii. single-end

```
#!/bin/bash
for num in `seq 28 31`
 do
  echo SRR31944${num}
  hisat2 -q -x /home/iza/Pobrane/index/index -U
/home/iza/Pulpit/fasq-gzip/TRIMM/SINGLE/sample_SRR31944${num}.fq.gz
-S wynik_SRR31944${num}.hisat.sam
done
```

W kolejnym etapie wykonano kompresje otrzymanych plików SAM w pliki BAM. W tym celu wykorzystano program *samtools*.

```
for i in $(ls *.sam)
 do
  echo "Kompresja dla pliku: $i"
  j=${i::-3}
```

```
  samtools view -Sb -@ 2 ${i} > ${j}bam
 done
```

5. Zliczanie odczytów

W celu zliczenia odczytów wykorzystano program *featureCounts* oraz pakiet z biblioteki *Bioconductor* (Rsubread). W pierwszym etapie pobrano plik z adnotacją genomową w formacie GTF poleceniem:

```
wget
'http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/genes/hg19.r
efGene.gtf.gz'
```

i. pair-end

```
featureCounts -a hg19.refGene.gtf -g gene_name -o
counts_refGene_pair.txt *4[2345].bam
```

ii. single-end

```
featureCounts -a hg19.refGene.gtf -g gene_name -o
counts_refGene_single.txt *[23][8901].bam
```

6. Analiza DE

Dalsza analiza została przeprowadzona w programie RStudio.

```
library(DESeq2)

DE_analysis = function(path){
  Data = read.csv(path, sep = "\t", skip = 1)
  countData = Data[, 7:10]
  samples = names(countData)
  rownames(countData) <-Data$Geneid
  cond_1 = rep("cond1", 2)
  cond_2 = rep("cond2", 2)
  condition = factor(c(cond_1, cond_2))
  colData = data.frame(samples=samples, condition=condition)
  dds = DESeq2::DESeqDataSetFromMatrix(countData=countData,
        colData=colData, design = ~condition)

  dds <- DESeq(dds)

  res <- results(dds)
  r <- res[res$baseMean!=0,]
  r <- r[r$log2FoldChange > 1 | r$log2FoldChange < -1, ]
  x <- !is.na(r$padj)
  r <- r[x, ]
  r <- r[r$padj<0.05, ]
  print(paste("ilosc genow istotnych statystycznie:",nrow(r)))
  return(r)
}

r_pair = DE_analysis(path = "C:/Users/Izabela/Documents/counts_refGene_pair
.txt")
```

```
## [1] "ilosc genow istotnych statystycznie: 1068"
```

```
r_single = DE_analysis("C:/Users/Izabela/Documents/counts_refGene_single.tx
t")
```

```
## [1] "ilosc genow istotnych statystycznie: 1574"
```

Powyższy kod pozwala na przeprowadzenie analizy różnicowej, za pomocą funkcji DESeq, z podziałem danych na dwie grupy (tj. dwie próbki zika i dwie mock). Wyniki filtrowane są poprzez usunięcie ekspresji równej 0, a także poprzez wybranie tylko genów których zmiana ekspresji jest dwa razy większa lub dwa razy mniejsza. Usuwane są także geny, których ta wartość wynosi NA, a następnie wybierane tylko istotne statystycznie – na podstawie wartości padj.

Za pomocą diagramu Venna przedstawiono wspólną ilość genów istotnych statystycznie w przypadku sekwencjonowania pair-end oraz single-end:

```
library(VennDiagram)
diag = venn.diagram(list(pair_end= r_pair@rownames, single_end= r_single@ro
wnames),fill = c("yellow", "red"), filename = NULL)
grid.newpage()
grid.draw(diag)
```

Figure 1: Diagram Venna dla wyników analizy DE

Na podstawie powyższych wyników można zauważyć, że ilość genów istotnych statystycznie w przeprowadzonej analizie jest zdecydowanie większa w przypadku sekwencjonowania single-end. Diagram Venna uwidocznił także, że wspólna ilość genów istotnych statystycznie dla przeprowadzonej analizy wynosi 833.

7. PCA oraz HeatMapa

Kolejnym etapem było wykonanie Heatmapy dla uzyskanych wyników. Heatmapa przedstawia tylko geny o wysokiej ekspresji, powyżej 10.

```
library(DESeq2)
library(ComplexHeatmap)
library(dplyr)
heatMap = function(path){
  countData = read.csv(path,
                       sep = "\t", skip = 1)
 countData = countData[, 7:10]
 samples = names(countData)
 rownames(countData) <-countData$Geneid

 cond_1 = rep("cond1", 2)
 cond_2 = rep("cond2", 2)
 condition = factor(c(cond_1, cond_2))
 colData = data.frame(samples=samples, condition=condition)
 dds = DESeqDataSetFromMatrix(countData=countData, colData=colData, design
= ~condition)

 log_data <- rlog(dds)
 norm_data<-assay(log_data)
 norm_data <- as.data.frame(norm_data)
 norm_data_10 <- norm_data %>% filter(!rowSums(. <10))
 norm_data_10
 dane_heatMap <- as.matrix(norm_data_10)
 dane_heatMap <- na.omit(dane_heatMap)
 Heatmap(dane_heatMap, cluster_columns=FALSE, row_names_side = "left",
 row_dend_sid = "left",
 row_names_gp=gpar(cex=0.6))
}

heatMap(path = "C:/Users/Izabela/Documents/counts_refGene_pair.txt")
```

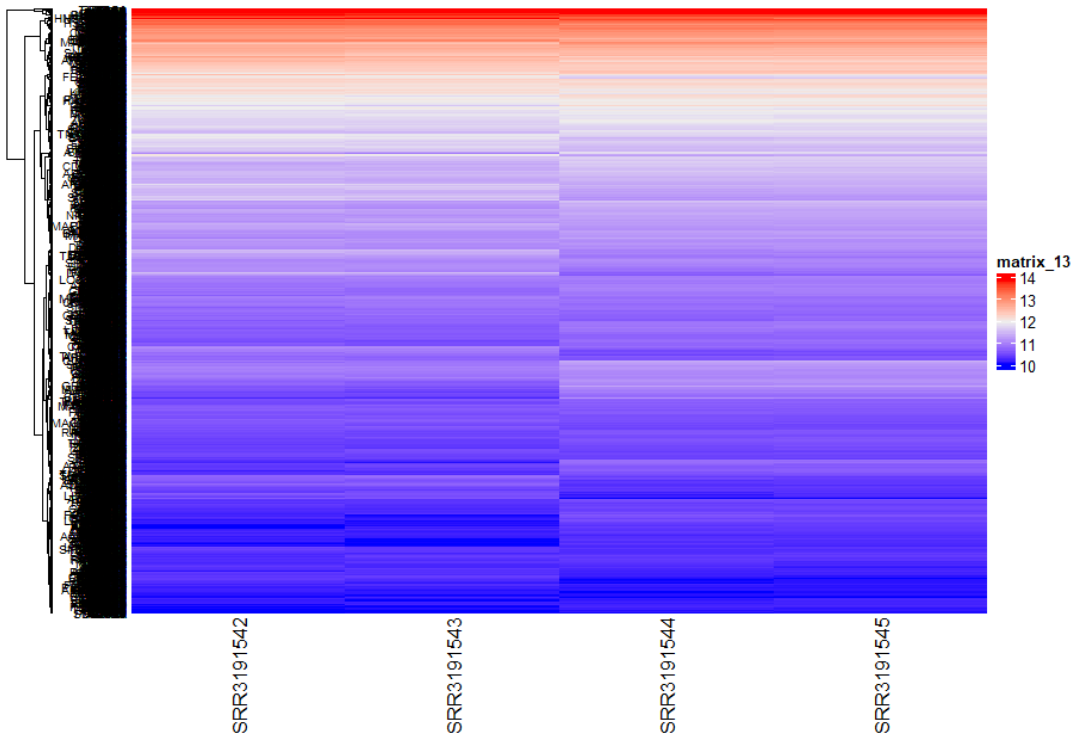

Figure 2: Heatmapa dla genów o wysokiej ekspresji, sekwencjonowanie pair-end

```
heatMap(path = "C:/Users/Izabela/Documents/counts_refGene_single.txt")
```

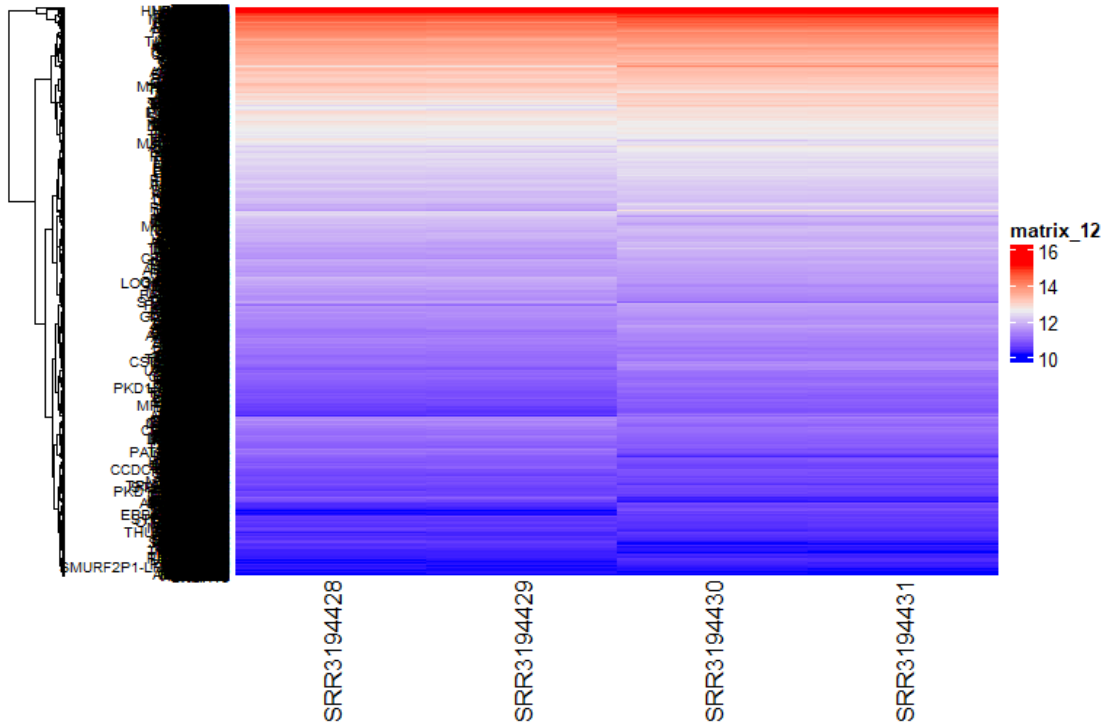

Figure 3: Heatmapa dla genów o wysokiej ekspresji, sekwencjonowanie single-end

Wyniki te ukazują nieco wyższą ekspresje genów w przypadku sekwencjonowania single-end, w porównaniu do pair-end. Dodatkowo na podstawie Fig3 można zauważyć niewiele większą ekspresje genów dla próbek Zika, w porównaniu z próbkami Mock.

Kolejny etap przedstawia PCA:

```
PCA = function(path){
  countData = read.csv(path,
                       sep = "\t", skip = 1)
 countData = countData[, 7:10]
 samples = names(countData)
 rownames(countData) <-countData$Geneid

 cond_1 = rep("mock", 2)
 cond_2 = rep("zika", 2)
 condition = factor(c(cond_1, cond_2))

 colData = data.frame(samples=samples, condition=condition)
 dds = DESeqDataSetFromMatrix(countData=countData, colData=colData, design
= ~condition)
 rld <- rlog(dds, blind=TRUE)
 plotPCA(rld)
}
```

```
PCA(path = "C:/Users/Izabela/Documents/counts_refGene_pair.txt")
```

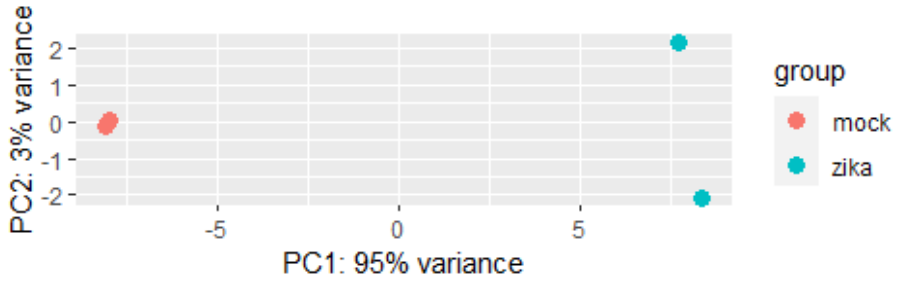

Figure 4: PCA pair-end

```
PCA(path = "C:/Users/Izabela/Documents/counts_refGene_single.txt")
```

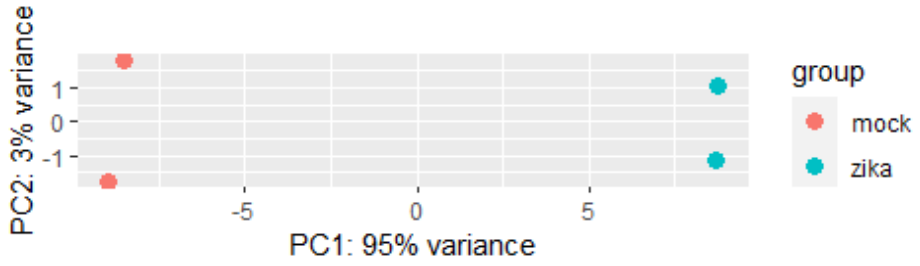

Figure 5: PCA single-end

Na podstawie Fig4 oraz Fig5 można zauważyć, że próbki kontrolne (mock) dla bibliotek pair-end są bardziej podobne do siebie, niż w przypadku bibliotek single-end. Zarówno dla biblioteki pair-end jak i single-end próbki kontrole różniły się od próbek zika.

Poniżej przedstawiono wspólny wykres PCA dla wszystkich ośmiu bibliotek, zachowując podział na próbki mock i zika, a także heatmape korelacji pomiędzy próbkami:

```
library(DESeq2)
library(pheatmap)
DataPE = read.csv("C:/Users/Izabela/Documents/counts_refGene_pair.txt", sep
= "\t", skip = 1)
countDataPE = DataPE[, 7:10]
DataSE = read.csv("C:/Users/Izabela/Documents/counts_refGene_single.txt", s
ep = "\t", skip = 1)
countDataSE = DataSE[, 7:10]
countData = cbind(countDataPE, countDataSE)
colnames(countData) = c("SRR3191542", "SRR3191543", "SRR3191544", "SRR31915
45", "SRR3194428", "SRR3194429", "SRR3194430", "SRR3194431")
 samples = names(countData)
 rownames(countData) <-countData$Geneid

 cond_1 = rep("mock", 2)
 cond_2 = rep("zika", 2)
 condition = factor(c(cond_1, cond_2))
 colData = data.frame(samples=samples, condition=condition)
 colData

##       samples condition
## 1 SRR3191542      mock
```

```
## 2 SRR3191543      mock
## 3 SRR3191544      zika
## 4 SRR3191545      zika
## 5 SRR3194428      mock
## 6 SRR3194429      mock
## 7 SRR3194430      zika
## 8 SRR3194431      zika
```

```
 dds = DESeqDataSetFromMatrix(countData=countData, colData=colData, design
= ~condition)
 rld <- rlog(dds, blind=TRUE)
 plotPCA(rld, intgroup = c("samples", "condition"), ntop = 500, returnData
= FALSE)
```

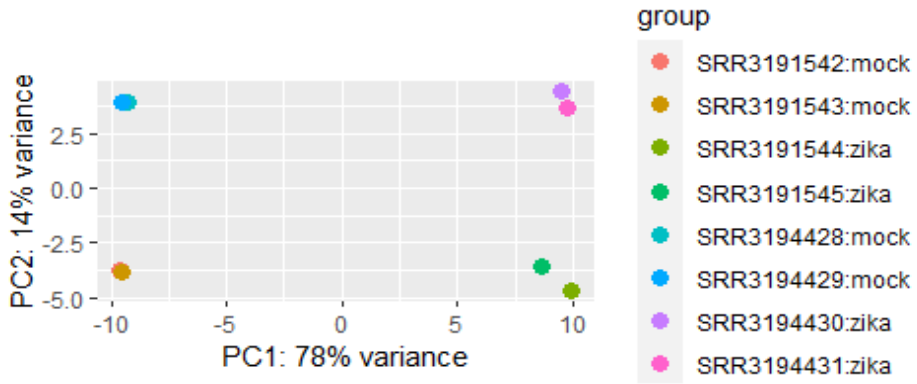

Figure 6: PCA dla całej analizy

```
rld_mat <- assay(rld)
 rld_cor <- cor(rld_mat)
 pheatmap(rld_cor)
```

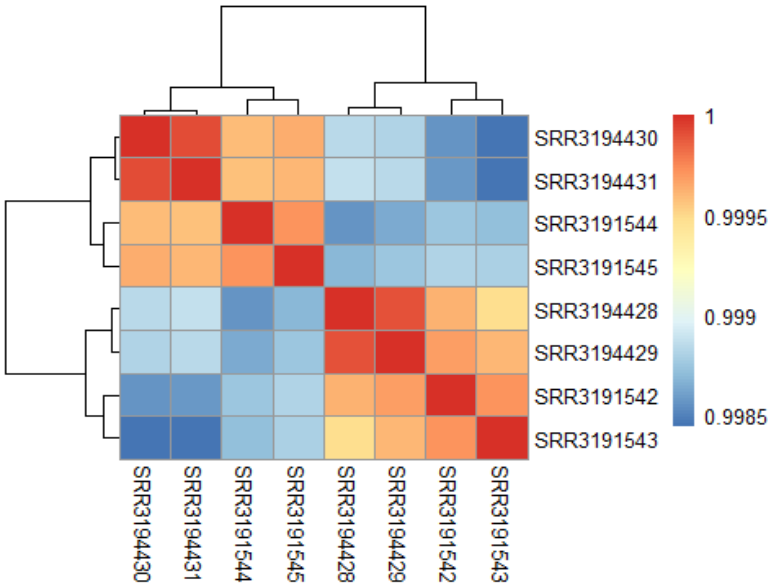

Figure 7: Heatmapa korelacji pomiędzy próbkami

Na podstawie analizy PCA (Fig.6) można zauważyć, że próbki grupują się ze względu na sposób sekwencjonowania (pair/single) oraz rodzaj próbki (badana/kontrolna). Ze względu na rozmieszczenie próbek na wykresie, widać także, że bardziej zbliżone są do siebie próbki mock w danym typie sekwencjonowania. Odpowiadające sobie próbki zika są od siebie oddalone, zarówno w sekwencjonowaniu pair-end jak i single-end, co oznacza mniejsze podobieństwo.

Na podstawie Heatmapy (Fig. 7) można stwierdzić wysoką korelacje (> 0.9985) dla wszystkich próbek oraz brak odstających wyników. Można zauważyć ponownie większą wzajemną korelację pomiędzy odczytami pair-end, a także oddzielnie pomiędzy odczytami single-end. Jedną z większych wartości korelacji wyróżniają się biblioteka SRR314428 z biblioteką SRR314429.

8. Wnioski

Otrzymane wyniki, a także sporządzone na ich podstawie wykresy, sugerują poprawność wykonanej analizy, a tym samym dane dobrej jakości.