Data Ingestion and Selection

2024-03-28

Data Ingestion and Selection

Initial analysis of RNAseq data requires proper data ingestion as well as gene and sample selection for downstream analysis. This file provides a tutorial for the ingestion of RNAseq data provided by a source such as Azenta (formerly Genewiz) which has previously been provided as a series of excel files with raw count data for each sample in a given two group comparison. However, the gene and sample selection process presented here can be used with RNAseq data from any source. Following ingestion, genes must be filtered to remove lowly expressed genes, and samples must be inspected to determine significant outliers due to factors such as mislabeling or instrumentation error.

Set up the environment

```
rm(list = ls(all.names = TRUE)) # clears global environ.

# Load packages
library(tidyverse)
library(dplyr)
library(biomaRt)
library(broom)
library(tibble)
library(edgeR)
library(factoextra)
library(biomaRt)
library(dendextend)
```

Ingestion of data

This process assumes a multi-group RNAseq project rather than simple case control design. As such, count data may be provided as multiple excel files, with samples from a specific two group comparison included. If data is provided in a different format, such as all samples and genes in a single file, simply import that file and skip this section.

For example, assume a study design with four groups with one vehicle control and three treatment groups. Standard analysis, as provided by Azenta, may have conducted differential analysis comparing each group to the control group (T1 vs C, T2 vs C, T3 vs C) and provided a .csv file for each. This process imports each of these .csv files and removes the samples duplicated between comparisons (control samples). Simply create one folder and add the raw count data .csv from each comparison with a unique name.

```
# # Create a list of the files from your target directory
# file_list <- list.files(path = "C:/---")
#
```

```
# # Create a blank list for storage of .csv data
# list <- list()
#
# # Loop through the length of the files and read that .csv and save it at location i within the list
# for (i in 1:length(file_list)){
#
#   list[[i]] <- read.csv(paste0('C:/---/',
#                                file_list[i]))
#
# }
#
# # Merge all data frames together and remove duplicate samples
# RNAseq <- Reduce(function(x, y) merge(x, y, all = TRUE), list)
```

Select genes using edgeR function filterByExpr()

Not all genes will have meaningful expression within or across samples, and these genes will likely change based on numerous factors including source cell type and treatment. Here, the edgeR function filterByExpr() is employed to determine which genes possess meaningful expression within and across samples. Samples without meaningful expression can be removed from analysis to reduce the size of the data, reduce noise, and speed up downstream analysis.

A mock data set is introduced here for downstream analysis. Remove this data and replace it with your own as needed.

```
# Set seed
set.seed(1234)

# Create a synthetic data set with 100 genes, 5 samples per group, and 400/500 genes differentially exp
RNAseq <- compcodeR::generateSyntheticData(dataset = "mydata",
                                           n.vars = 500,
                                           samples.per.cond = 20,
                                           n.diffexp = 400)@count.matrix
```

```
## Registered S3 method overwritten by 'rmutil':
##   method         from
##   print.response httr
```

```
# Shuffle around the synthetic data to create 4 random groups with 10 samples each
Control <- RNAseq[sample(nrow(RNAseq)), sample(ncol(RNAseq))][, 1:10] %>%
  `rownames<-` (paste0('g', seq(1, 500, 1)))

Treatment_1 <- data.frame(RNAseq[sample(nrow(RNAseq)), sample(ncol(RNAseq))][, 1:10]) %>%
  `rownames<-` (paste0('g', seq(1, 500, 1)))

Treatment_2 <- data.frame(RNAseq[sample(nrow(RNAseq)), sample(ncol(RNAseq))][, 1:10]) %>%
  `rownames<-` (paste0('g', seq(1, 500, 1)))

Treatment_3 <- data.frame(RNAseq[sample(nrow(RNAseq)), sample(ncol(RNAseq))][, 1:10]) %>%
  `rownames<-` (paste0('g', seq(1, 500, 1)))

# Combine synthetic data groups into one data frame
```

```
RNAseq <- data.frame(Control,
                     Treatment_1,
                     Treatment_2,
                     Treatment_3) %>% t()

# Lets get a list of real Ensembl IDs to randomly assign them to our synthetic data
Ensembl_ID <- read.csv('Ensembl_IDs.csv')

# Randomly sample genes to be included into the data set and set samples from 1 to 40
colnames(RNAseq) <- Ensembl_ID[sample(nrow(Ensembl_ID)), ][1:ncol(RNAseq)]
rownames(RNAseq) <- paste0('Sample_', seq(1, nrow(RNAseq), 1))

# Lets create 5 more dummy genes with lower counts to demonstrate filtering
RNAseq <- data.frame(Biological_Sample = factor(rep(seq(1, 10, 1), 4)),
                     Treatment = factor(rep(c('Control',
                                              'Treatment_1',
                                              'Treatment_2',
                                              'Treatment_3'),
                                            each = 10)),
                     RNAseq,
                     ENSG00000000061 = sample(1000, size = 40, replace = TRUE),
                     ENSG00000000062 = sample(100, size = 40, replace = TRUE),
                     ENSG00000000063 = sample(10, size = 40, replace = TRUE),
                     ENSG00000000064 = sample(1, size = 40, replace = TRUE),
                     ENSG00000000065 = sample(0, size = 40, replace = TRUE))

head(RNAseq[1:5, 1:5])
```

```
##          Biological_Sample Treatment ENSG00000214107 ENSG00000252743
## Sample_1                 1   Control           10141             313
## Sample_2                 2   Control            7805             370
## Sample_3                 3   Control            4425             292
## Sample_4                 4   Control           20079              36
## Sample_5                 5   Control           22609             387
##          ENSG00000126353
## Sample_1             410
## Sample_2             172
## Sample_3              33
## Sample_4             172
## Sample_5               0
```

```
# Create DGEList object
# DGElist accepts a matrix of nrow genes and ncol samples
# The group designation must be an equal length to ncol samples
d0 <- DGEList(as.matrix(t(RNAseq[3:ncol(RNAseq)])),
              group = RNAseq$Biological_Sample)

# Calculate normalization factor
d0 <- calcNormFactors(d0)

# Use EdgeR function to automatically filter genes
# This gives logical values to keep or remove
keep.exprs <- filterByExpr(d0,
```

```
                            group = RNAseq$Biological_Sample)

# Removed genes due to insufficient counts across samples
remove <- colnames(RNAseq[, !keep.exprs])

dim(RNAseq)
```

```
## [1]  40 507
```

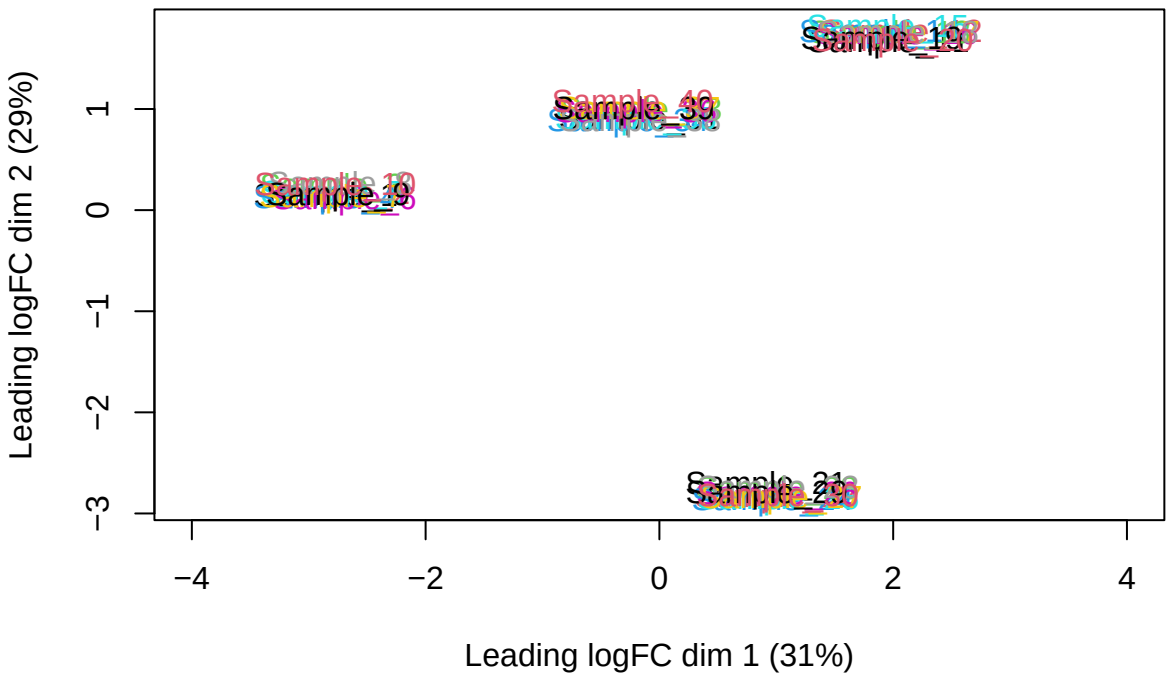
```
# Remove filtered genes from the dataset
RNAseq <- RNAseq[, keep.exprs]

dim(RNAseq)
```

```
## [1]  40 505
```

Check for outlier samples

Samples can be identified as outliers for a number of reasons. At this point, the biggest concern is major technical failure, such as mislabeling of samples or defects in the sequencer. If a sample is identified as a likely outlier in this analysis, it may be worth applying more robust statistical methods of outlier detection to better rationalize exclusion of a sample.

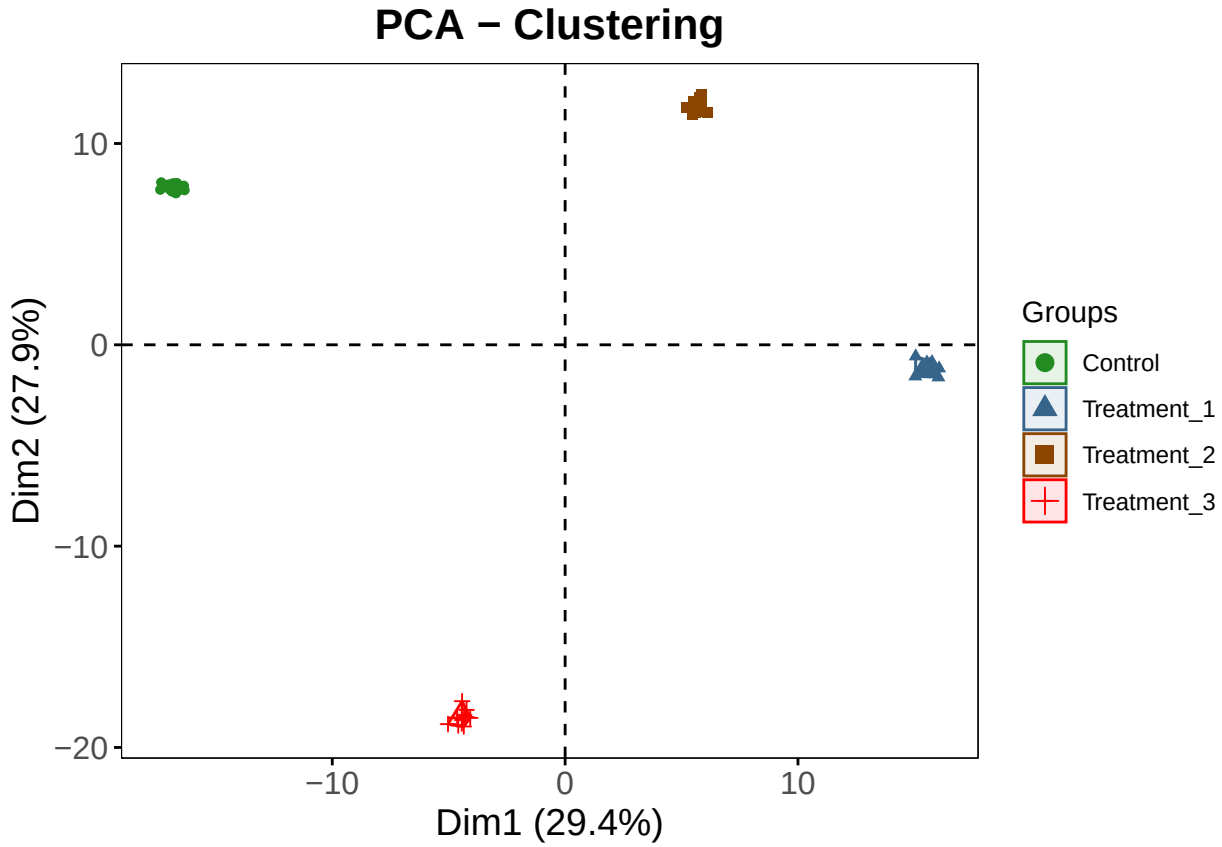
```
# Convert into a matrix
counts <- as.matrix(t(RNAseq[3:ncol(RNAseq)]))

# Create DGEList object
dataset <- DGEList(counts,
                   group = RNAseq$Biological_Sample)

# Calculate normalization factor
dataset <- calcNormFactors(dataset)

# Calculate logCPM of filtered gene set
lcpm <- cpm(dataset, log = TRUE)

# Get count of number of samples in each group
counts <- RNAseq %>% group_by(Biological_Sample) %>%
  summarise(total_count = n(),
            .groups = 'drop')

# Set group variable to a factor
Biological_Sample <- factor(RNAseq$Biological_Sample)

# Create a variable setting group to a number for MDS plot
col <- as.numeric(Biological_Sample)

# Create an MDS plot of log CPM data, labeled based on the group number set in col above
plotMDS(lcpm,
        col = col,
        xlim = c(-4, 4))
```




```
# Run PCA
pca.res <- prcomp(t(lcpm), center = TRUE, scale = TRUE)

# Set theme
theme_set(theme_bw())

# Set colors object to the length of colors you need for below
colors <- c("forestgreen",
            "steelblue4",
            "darkorange4",
            "red")

# Look up this table if you want different/additional symbols
shapes <- c(18, 9, 15, 12, 17)

# All groups cluster plot
fviz_pca_ind(pca.res,
             label = "none",
             habillage = RNAseq$Treatment, # What are ellipses drawn around?
             palette = colors, # Set palette to colors defined above
             addEllipses = TRUE, # Set to FALSE if no ellipse desired
             ellipse.type = 'convex' # Look up variable for options
) +

  ggtitle("PCA - Clustering") + # Set title
```

```
  # Define axis information
  theme(axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        plot.title = element_text(hjust = 0.5, face = "bold", size = 16),
        panel.border = element_rect(fill = NA, color = "black", size = 0.3),
        panel.grid.minor = element_blank(),
        panel.grid.major = element_blank())
```

```
## Warning: The ‘size‘ argument of ‘element_rect()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



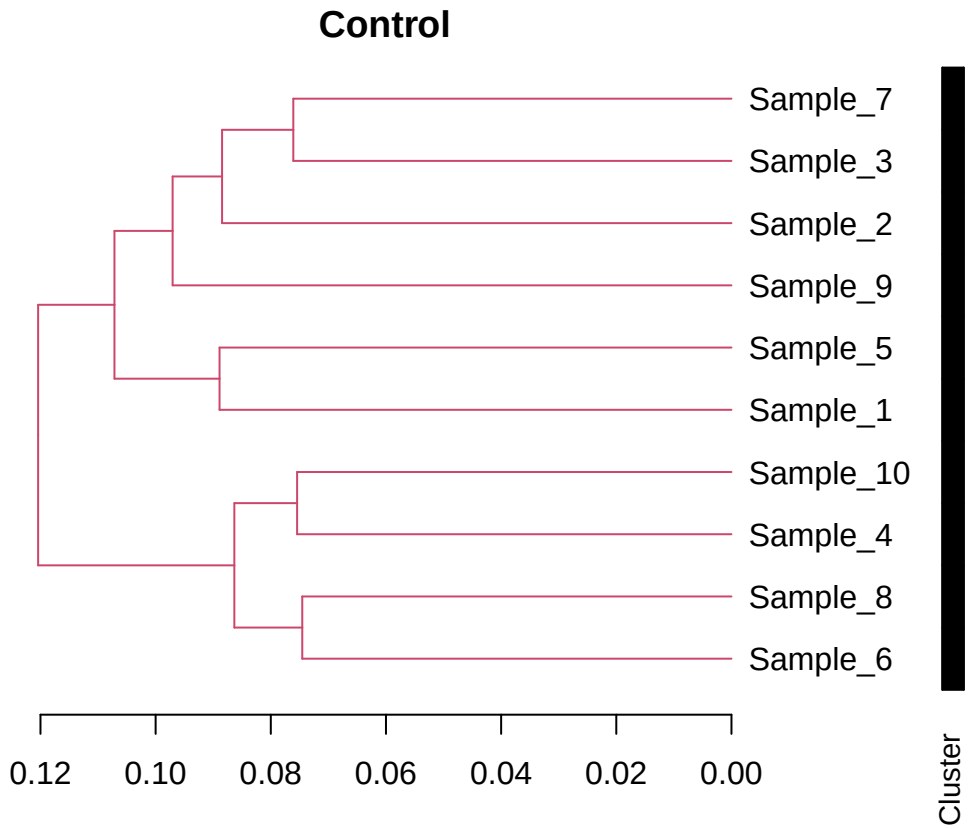
Determine outliers through dendrogram clustering

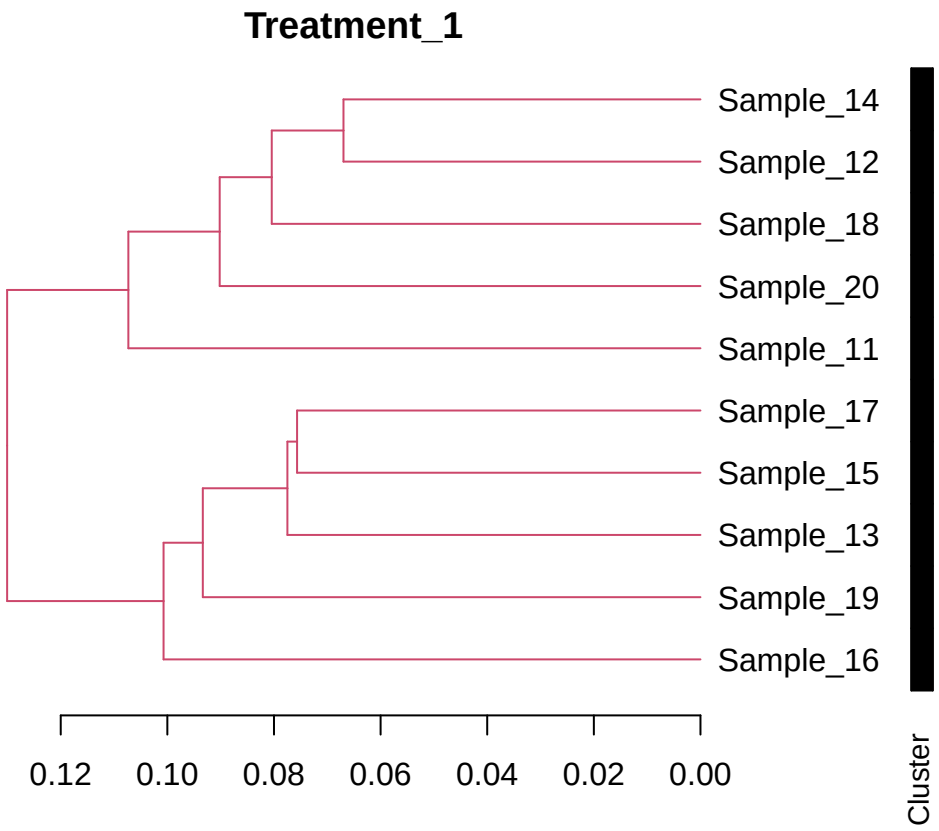
One method to determine outliers is to perform dendrogram clustering. Here, samples from each group are measured for within treatment group correlations and dendrograms are made using these correlations. Here, a cut height of 0.25 (i.e. pearson correlation of 0.75) is employed. Any samples with correlations below 0.75 are selected for removal.

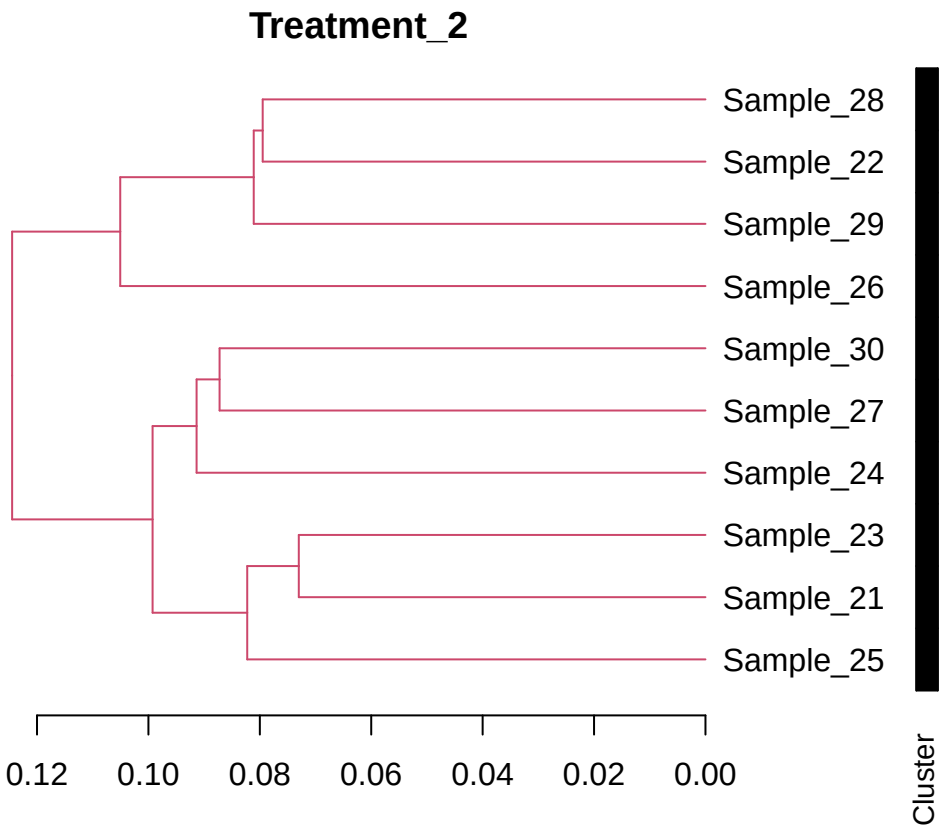
```
# Add sample information back to log cpm data
lcpm <- data.frame(RNAseq[1:2], t(lcpm))
```

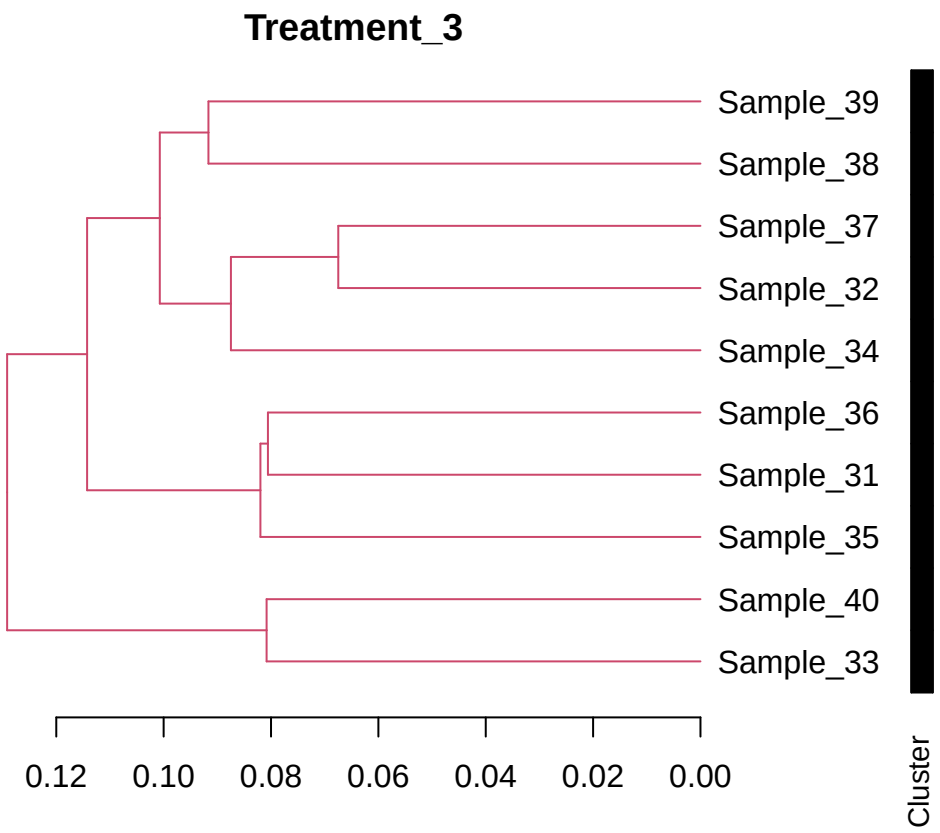
```
list <- as.list(unique(as.character(lcpm$Treatment)))

# Create dendrograms based on sample correlations from each group
dend <- lapply(list, function(x){

  tmp <- lcpm %>% subset(Treatment == x)
  names <- rownames(tmp)
  tmp_series <- tmp$series_id

  tmp <- tmp %>%
    subset(select = -c(1:2)) %>%
    sapply(as.numeric) %>%
    as.data.frame(row.names = names)

  tmp_dend <- cor(t(tmp))
  tmp_dend <- as.dendrogram(hclust(as.dist(1 - tmp_dend)))

  useries = unique(tmp_series)

  # Match unique series to series list and record row location of each series
  series_match = useries[match(tmp_series, useries)]

  # Set colors
  colos <- colorspace::rainbow_hcl(length(tmp_series),
                                   c = 160,
                                   l  = 50)

  # Set colors to series ID
  names(colos) = tmp_series

  # Set matched colors
  series_color <- colos[series_match]

  # Create clusters
  clu = cutree(tmp_dend,
               h = 0.25) # Height for cut (1 - correlation coefficient)

  # Set colors of labels
  labels_colors(tmp_dend) <- series_color[order.dendrogram(tmp_dend)]

  # Create dendrograms
  tmp_dend <- color_branches(tmp_dend,
                             h = 0.25) # Height of cut (1 - correlation coefficient)

  par(mar = c(4,1,1,12))

  plot(tmp_dend,
       main = as.character(x),
       horiz = TRUE) +

    abline(v = 0.25, lty = 2)

  colored_bars(cbind(clu,
```

```
                    series_color),
                tmp_dend,
                rowLabels = c("Cluster", "Series"),
                horiz = TRUE)

  tmp <- lcpm %>% subset(Treatment == x)

  tmp <- cbind(clu, tmp)
  colnames(tmp)[1] <- 'cluster'
  tmp <- tmp %>% dplyr::select(Treatment, everything())

  largest_cluster = names(rev(sort(table(tmp$cluster))))[1]
  ww = which(tmp$cluster == largest_cluster)
  tmp[ww, ]

})
```




Treatment_1
Sample_14
Sample_12
Sample_18
Sample_20
Sample_11
Sample_17
Sample_15
Sample_13
Sample_19
Sample_16
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Cluster


Treatment_2
Sample_28
Sample_22
Sample_29
Sample_26
Sample_30
Sample_27
Sample_24
Sample_23
Sample_21
Sample_25
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Cluster




```
# Combine all selected samples into one place and remove dendrogram cluster information
Filtered_df <- do.call(rbind, dend) %>% subset(select = -c(2))

# Remove filtered samples from raw count data frame
RNAseq <- RNAseq[rownames(Filtered_df), ]
```

Export selected data

```
save(RNAseq,
     file = 'Selected_Raw_Data.RDATA')

save(Filtered_df,
     file = 'Selected_lcpm_Data.RDATA')
```