Unsupervised Analysis - Partitioning Methods

K Siegmund

6/3/2020

Libraries

```
library(ComplexHeatmap)
```

```
## Loading required package: grid

## ========================================
## ComplexHeatmap version 2.6.2
## 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.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(ComplexHeatmap))
## ========================================
```

```
library(matlab)   # this library let's us use blue-red color spectrum for heatmap  (jet.colors)
```

```
##
## Attaching package: 'matlab'

## The following object is masked from 'package:stats':
##
##     reshape

## The following objects are masked from 'package:utils':
##
##     find, fix

## The following object is masked from 'package:base':
##
##     sum
```

```
library(matrixStats)
library(stats)
library(ggplot2)
library(gg3D)
library(cluster)
```

Prostate Cancer Data

I'm going to apply the partioning algorithms to the same data that we used for the hierarchical clustering, specifically selecting the 500 most variable features and standardizing them.

```
load("data/JBC 2012/jbcdat.rda")
```

```
rowscale <- function(x) {
      (x - rowMeans(x))/matrixStats::rowMads(x)
}

fmad <- matrixStats::rowMads(jbcdat$E)
rfilt <- rank(-fmad)
fidx <- which( rfilt <= 500)

X <- t(rowscale(jbcdat$E)[fidx,])
dim(X)
```

```
## [1]  24 500
```

We have 24 samples, each with 500 gene expression measurements.

K-means

Kmeans generates starting values for the algorithm using a random number generator. In order to reproduce our results later, we have to set the random number seed using the set.seed() command.

Here's the command for kmeans, with a table comparing the cluster assignments to the actual treatments.

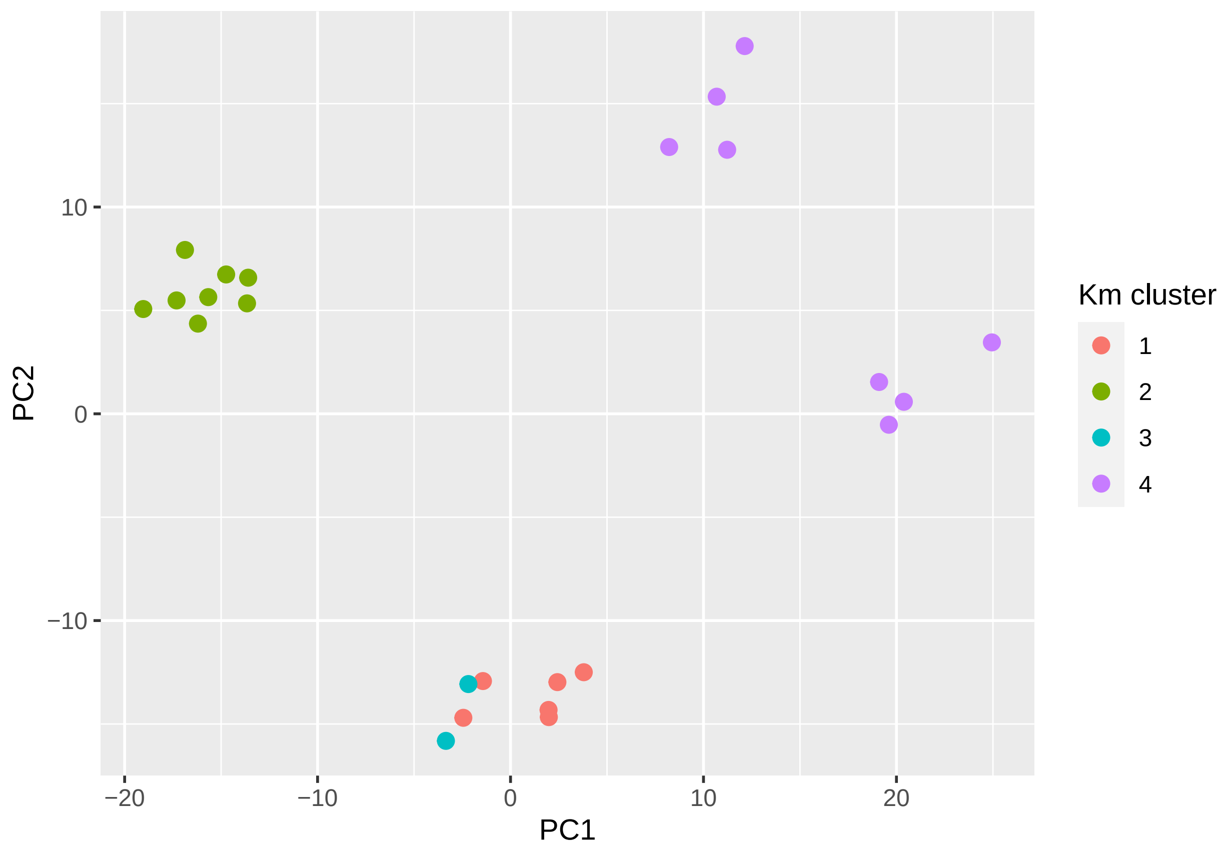
```
set.seed(46)
km4 <- stats::kmeans(X,4)
table(jbcdat$targets$type,km4$cluster)
```

```
##
##                1 2 3 4
##   siCBP_0hr    4 0 0 0
##   siCBP_16hr   0 4 0 0
##   siNS_0hr     2 0 2 0
##   siNS_16hr    0 4 0 0
##   sip300_0hr   0 0 0 4
##   sip300_16hr  0 0 0 4
```

Let's visualize these cluster assignments by coloring the samples on the 2-D PCA plot.

```
my.pca <- prcomp(X,retx=TRUE)
dfx <- as.data.frame(my.pca$x)
```

```
ggplot(dfx,  aes(x=PC1, y=PC2, color = factor(km4$cluster))) +
           geom_point(size=2.5) +
  labs(color="Km cluster")
```

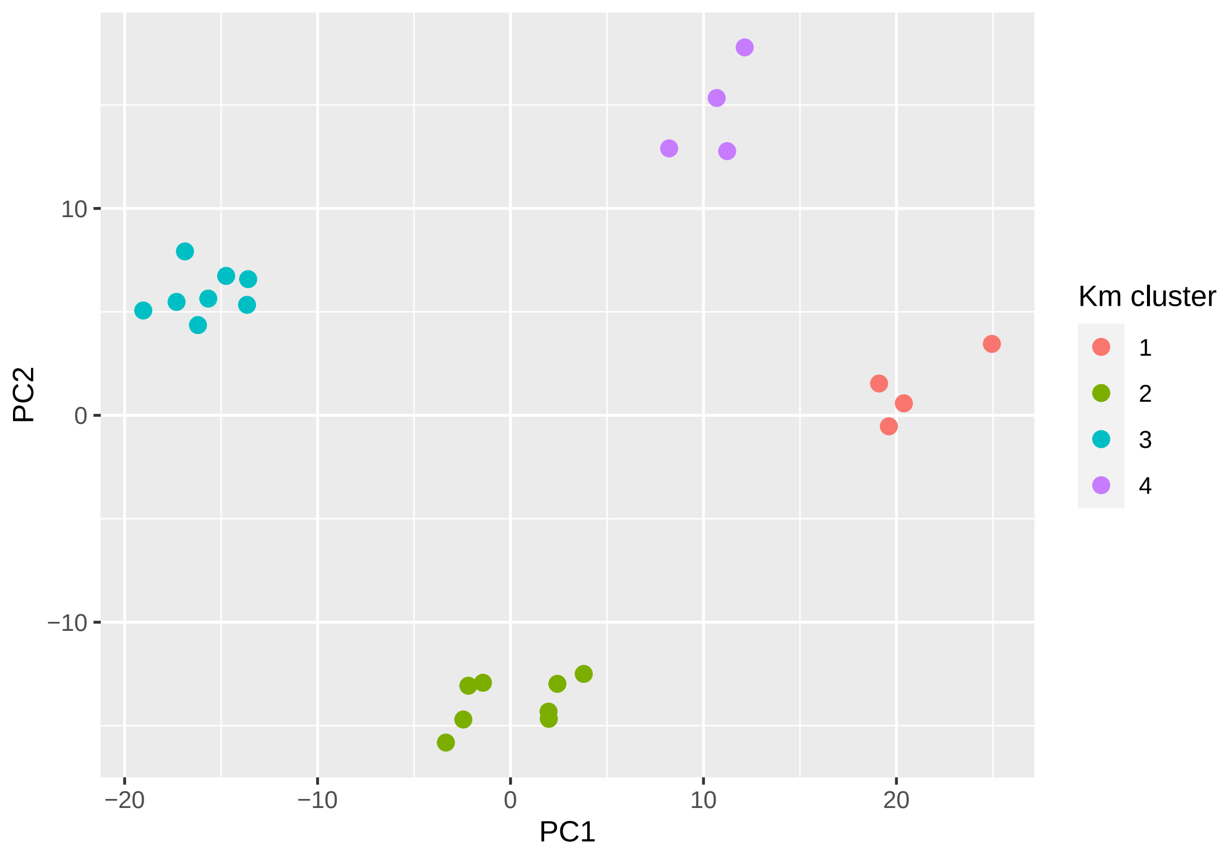


We see cluster 3 (aqua) does not correctly identify a complete treatment group. But, with different starting values, the kmeans algorithm may find a different solution.

Let's repeat the analysis using 200 random starting values and select the solution with the smallest within-cluster sum of squares.

```
set.seed(99)
kmx <- stats::kmeans(X,centers = 4,nstart = 200)
#table(jbcdat$targets$type,kmx$cluster)
```

```
ggplot(dfx,  aes(x=PC1, y=PC2, color = factor(kmx$cluster))) +
           geom_point(size=2.5) +
  labs(color="Km cluster")
```



A-ha! When we pick the best solution from multiple starts (lowest within-cluster sum of squares from 200 different starts) we separate the samples perfectly into 4 treatment groups. But why 4?

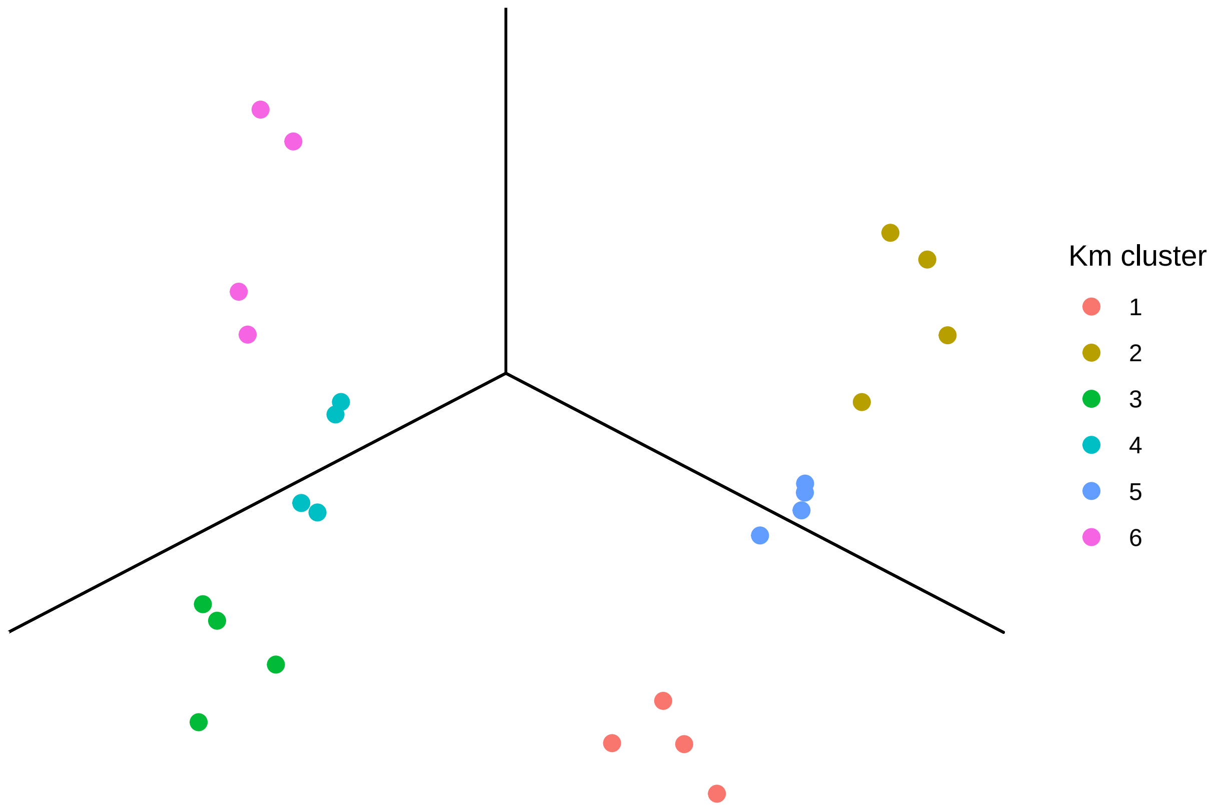
Can we recover all 6 treatment groups?

```
set.seed(99)
km6 <- stats::kmeans(X,centers = 6,nstart = 200)
table(jbcdat$targets$type,km6$cluster)
```

```
##
##                1 2 3 4 5 6
##   siCBP_0hr    0 0 0 4 0 0
##   siCBP_16hr   0 0 0 0 4 0
##   siNS_0hr     0 0 0 0 0 4
##   siNS_16hr    0 4 0 0 0 0
##   sip300_0hr   0 0 4 0 0 0
##   sip300_16hr  4 0 0 0 0 0
```

Yes! These are the 6 treatment groups. We need a 3-D PCA plot to show this.

```
ggplot(dfx, aes(x=PC1, y=PC2, z=PC3,
                    color=factor(km6$cluster))) +
  theme_void() +
  axes_3D() +
  stat_3D(size=2.5) +
  labs(color="Km cluster")
```



How might we indicate the different treatment labels on the figure to show the concordance?

Heatmap

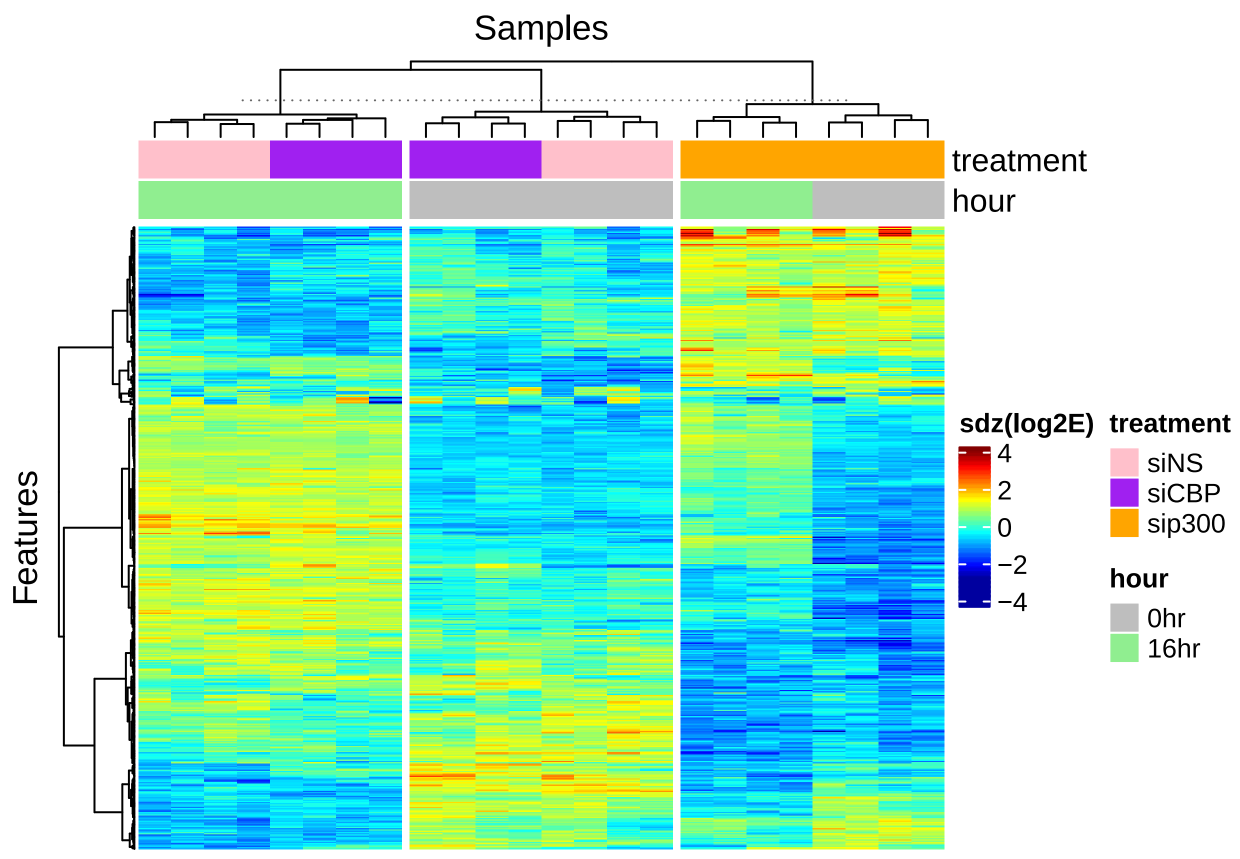
Another way to show the results is to use a heatmap. Here are the color annotations for sample treatments again.

```
jbcdat$targets$treatment <- factor(jbcdat$targets$treatment,
                                   levels=c("siNS","siCBP","sip300"))

# column heatmap annotation
colha <- ComplexHeatmap::HeatmapAnnotation(df =
                       jbcdat$targets[,c("treatment","hour")],
                col = list(treatment = c(siNS = "pink",
                                         siCBP = "purple",
                                         sip300 = "orange"),
                                 hour = c('0hr' = "grey",
                                         '16hr' = "lightgreen")
                          ),
                which = "column")
```

And the k-means clustering can be called directly from the heatmap function in the ComplexHeatmap library.

```
set.seed(200)
htc <- ComplexHeatmap::Heatmap(t(X),
           clustering_distance_rows = "pearson",
           clustering_method_rows = "ward.D2",
           column_km = 4,
           column_km_repeats = 100,
              column_title = "Samples",
              row_title = "Features",
              name = "sdz(log2E)",
              col = jet.colors(32),
              top_annotation = colha,
              show_column_names = FALSE,
              show_row_names = FALSE)
draw(htc)
```

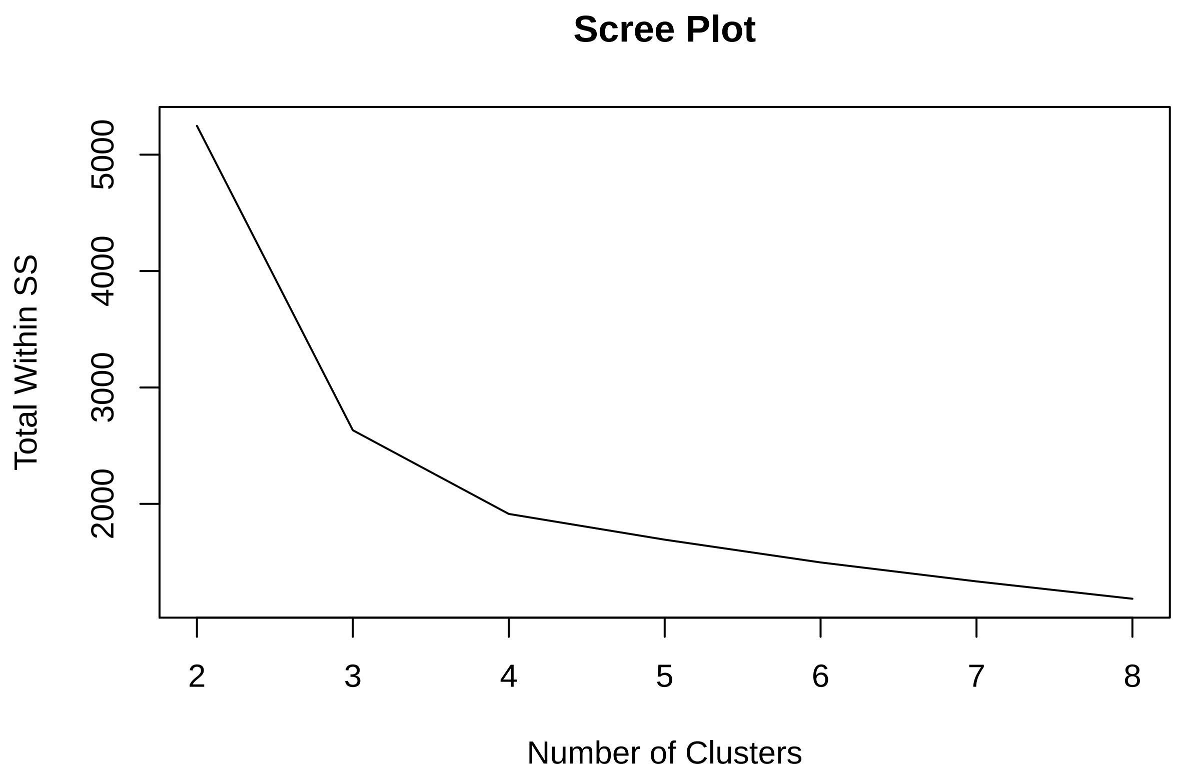


Notice how it only draws 3 groups when we ask for 4. This has to do with how it summarizes the results from the 100 random starting points. It reports a "consensus" k-means cluster result that agrees as much as possible with each of the 100 cluster results, instead of picking the single result that minimizes the clustering criterion. There are different methods for picking a criterion to measure cluster agreement.

Number of clusters

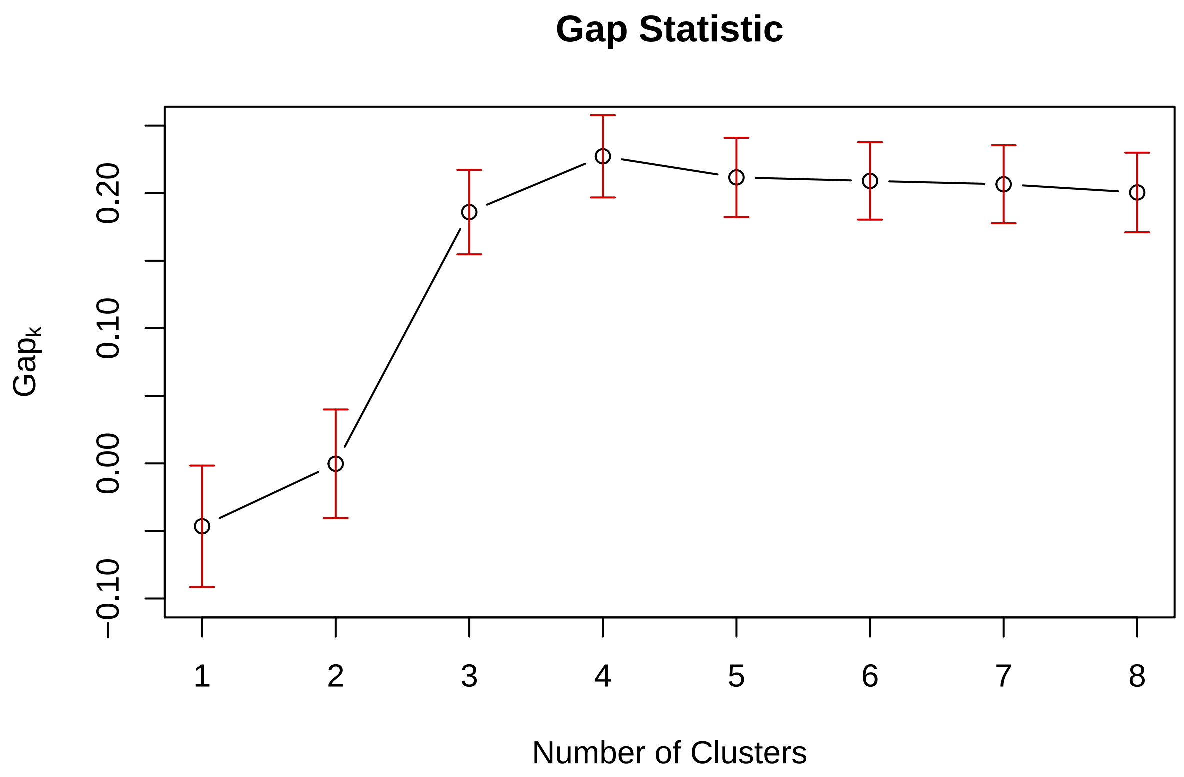
Here are 2 approaches for estimating the number of clusters.

```
set.seed(101)
twss <- rep(NA,7)
for (i in 1:7) {
  km <- stats::kmeans(X,i+1,nstart=200)
  twss[i] <- km$tot.withinss
  }
plot(2:8,twss,main="Scree Plot",xlab="Number of Clusters", ylab="Total Within SS",type="l")
```



The above plot is called a scree plot. One picks the number of clusters where there is an 'elbow' (bend). It is a very old approach, and many more modern ones have been proposed. For the above plot the answer is either 3 or 4, but which one?

Here's one more recent approach, the Gap statistic. It measures a goodness of clustering.



One picks the first k such that: Gap(k) > Gap(k+1) - SE(k+1)

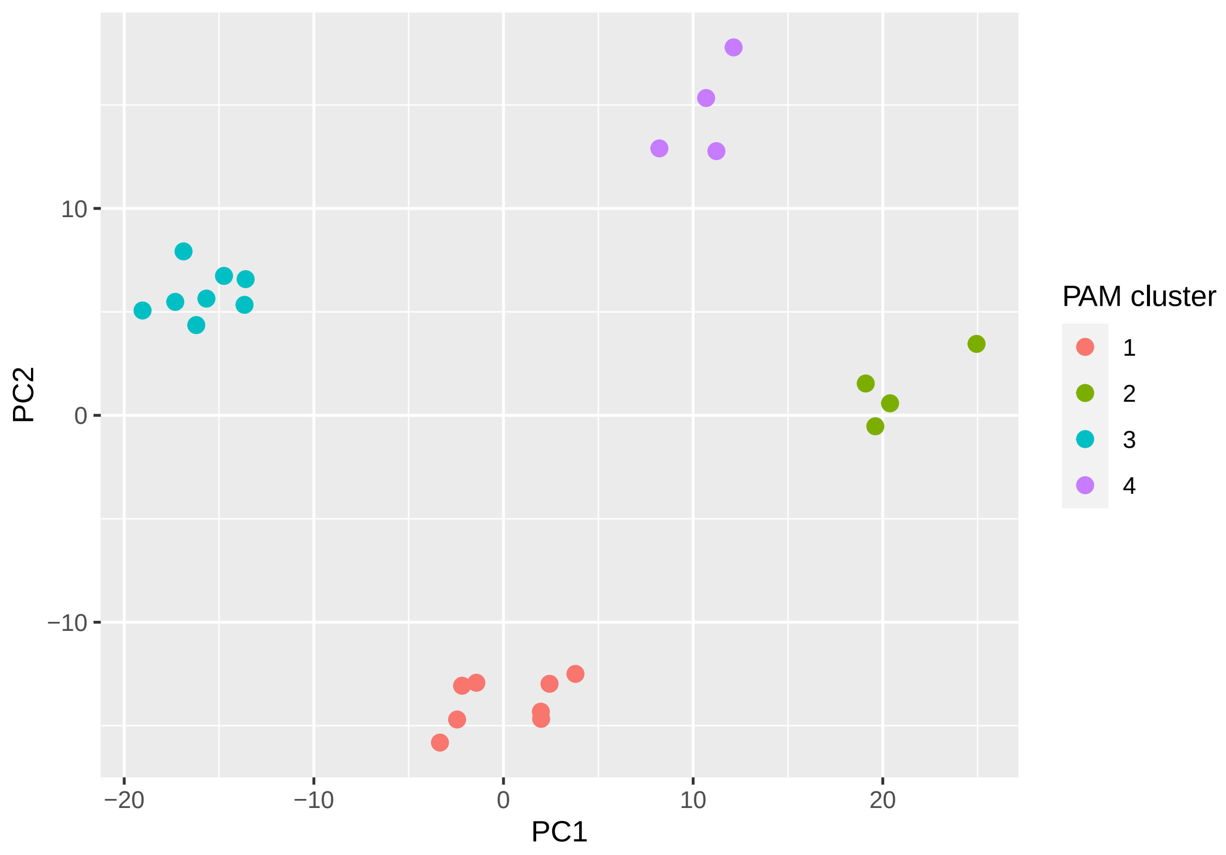
PAM

The function partioning around medoids (PAM) is available in cluster library.

```
p4=cluster::pam(X,4)
table(jbcdat$targets$type,p4$cluster)
```

```
##
##                1 2 3 4
##    siCBP_0hr   4 0 0 0
##    siCBP_16hr  0 0 4 0
##    siNS_0hr    4 0 0 0
##    siNS_16hr   0 0 4 0
##    sip300_0hr  0 4 0 0
##    sip300_16hr 0 0 0 4
```

```
ggplot(dfx,  aes(x=PC1, y=PC2, color = factor(p4$cluster))) +
           geom_point(size=2.5) +
  labs(color="PAM cluster")
```

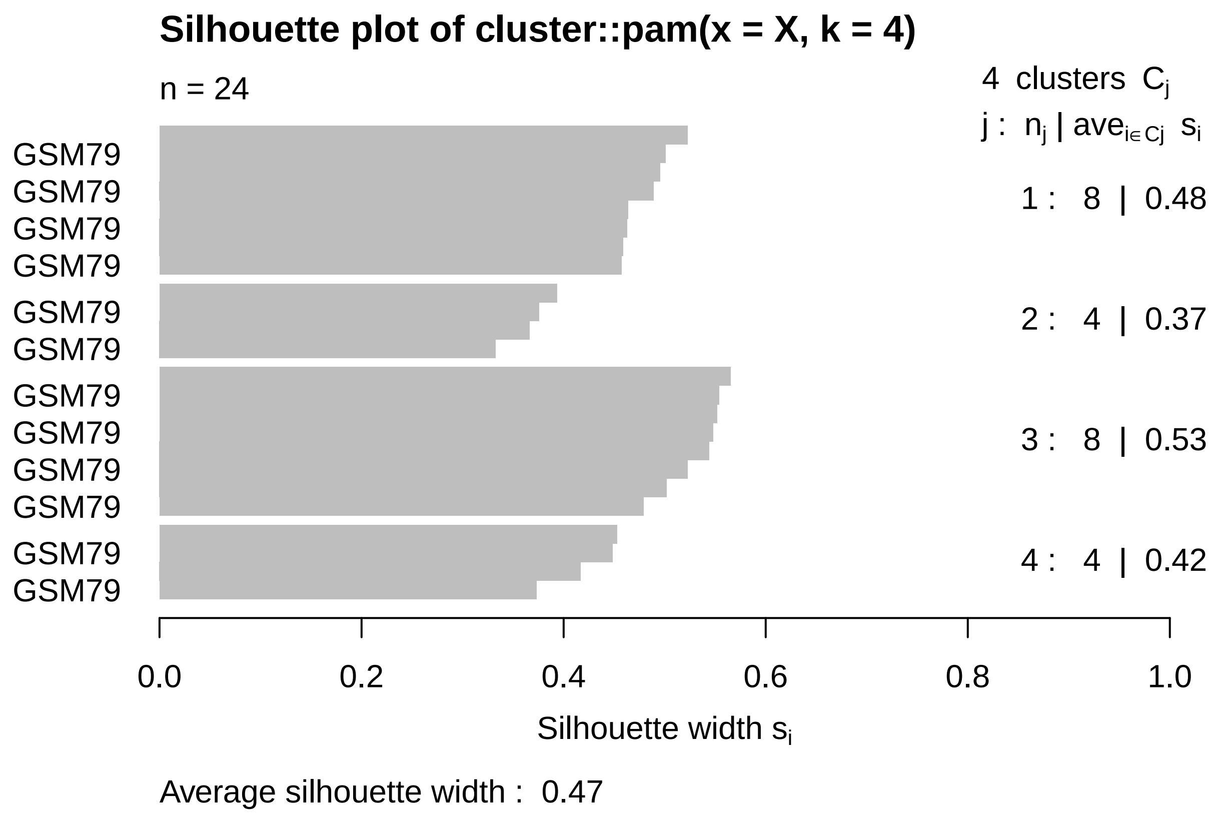


Nice! This robust technique separated the 4 treatment groups without multiple starts.

Number of clusters

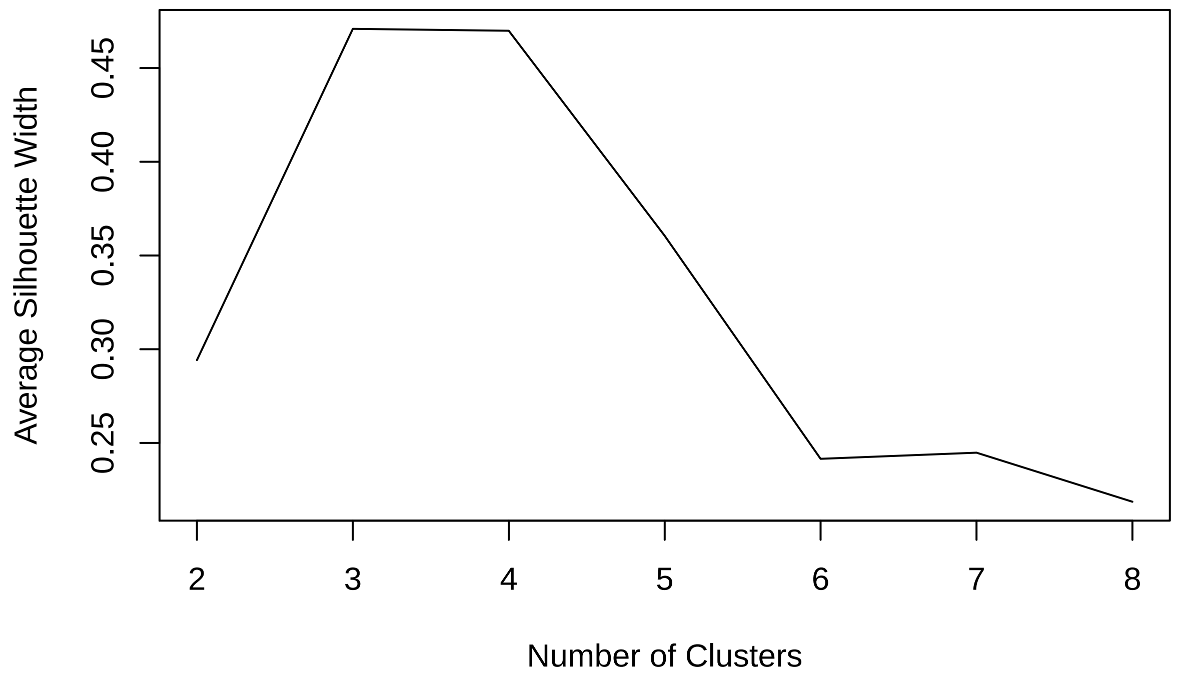
The Silhouette plot is recommended for selecting the number of clusters.

```
silpam4=silhouette(p4)
```



Let's check how many groups the data support.

```
sw <- rep(NA,7)
for (i in 1:7) {
  pm <- pam(X,i+1)
  sw[i] <- summary(pm)$silinfo$avg.width
  }
plot(2:8,sw,xlab="Number of Clusters", ylab="Average Silhouette Width",type="l")
```



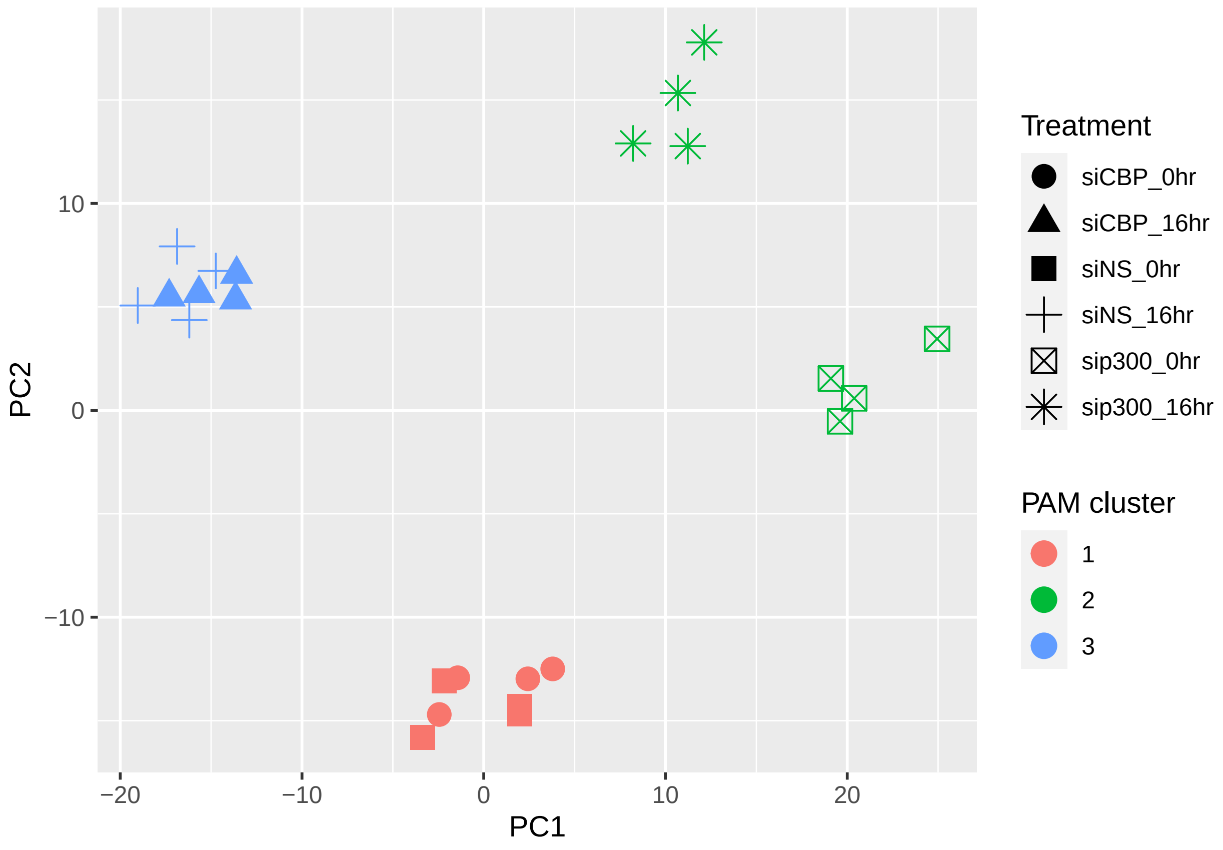
The Silhouette width is oh-so-close for 3 and 4 groups. Let's see how our partitioning around medoids algorithm sumarizes the data with only 3 groups.

```
p3 <- pam(X,3)
table(jbcdat$targets$type,p3$cluster)
```

```
##
##                1 2 3
##   siCBP_0hr    4 0 0
##   siCBP_16hr   0 0 4
##   siNS_0hr     4 0 0
##   siNS_16hr    0 0 4
##   sip300_0hr   0 4 0
##   sip300_16hr  0 4 0
```

```
#summary(p3)$silinfo$avg.width
```

```
ggplot(dfx,  aes(x=PC1, y=PC2)) +
           geom_point(size=4,aes(shape=jbcdat$targets$type,
                          color = factor(p3$cluster))) +
  labs(shape = "Treatment", color = "PAM cluster")
```



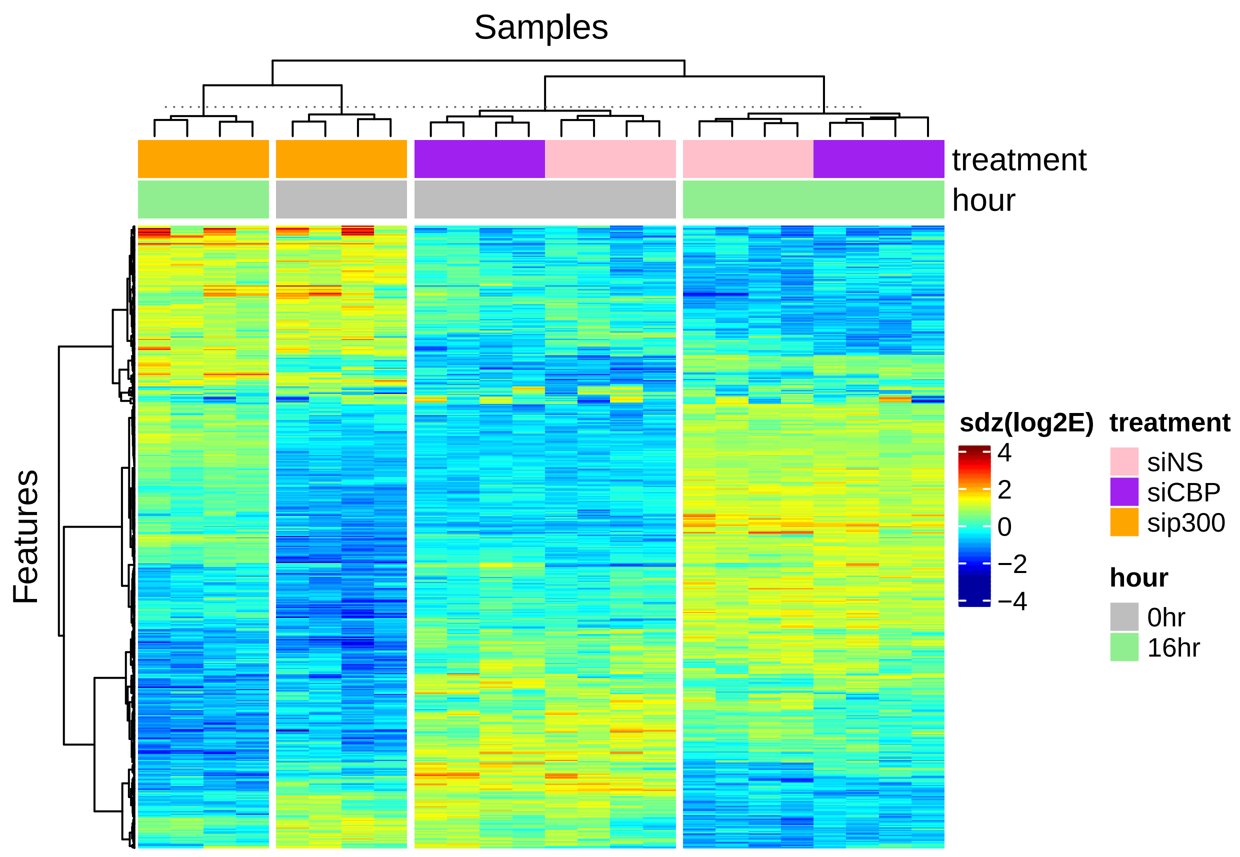
The two sip300 treatments (0h, 16h) are different from the rest, but similar to each other.

Heatmap

We can present the data matrix using a heatmap, with the columns ordered by the PAM cluster assignments.

```
dfr <- jbcdat$targets[,c("treatment","hour")]
colha <- ComplexHeatmap::HeatmapAnnotation(df = dfr,
                col = list(treatment = c(siNS = "pink",
                                         siCBP = "purple",
                                         sip300 = "orange"),
                                 hour = c('0hr' = "grey",
                                         '16hr' = "lightgreen")
                           ),
                which = "column")
```

```
# split columns by clust3 variable
#clust3 <- as.character(p3$clustering)
clust4 <- as.character(p4$clustering)
htc <- ComplexHeatmap::Heatmap(t(X),
           clustering_distance_rows = "pearson",
           clustering_method_rows = "ward.D2",
           column_split = clust4,
              column_title = "Samples",
              row_title = "Features",
              name = "sdz(log2E)",
              col = jet.colors(32),
              top_annotation = colha,
              show_column_names = FALSE,
              show_row_names = FALSE)
draw(htc)
```

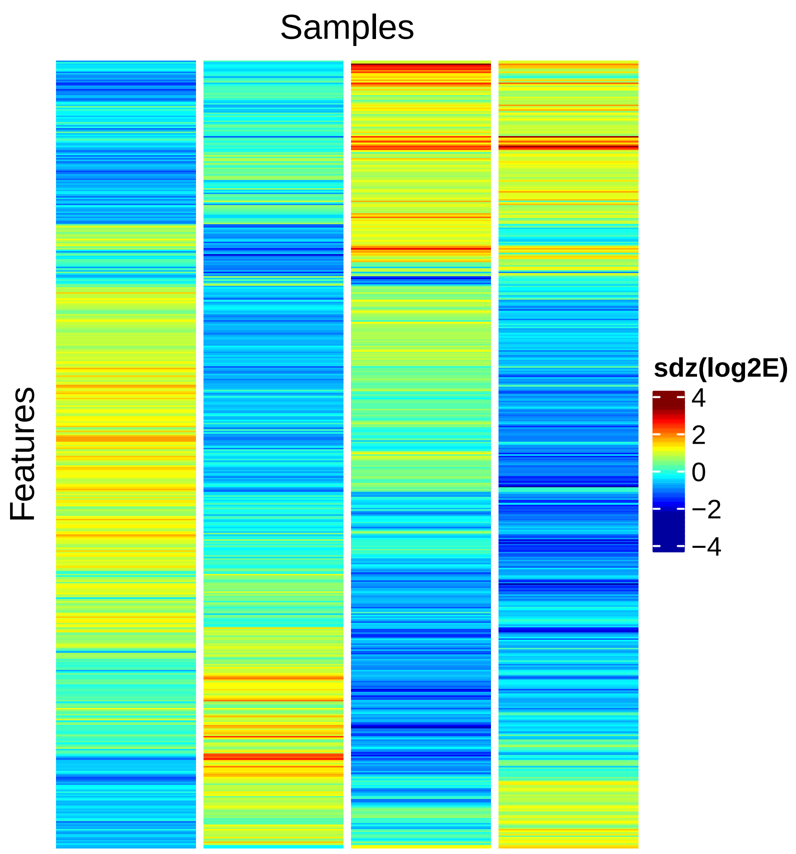


Alternatively, we may want to just plot the cluster means/medoids. I will use the same row order as from our clustering of all 24 samples.

```
#some exploration of the plot above led me to find out the clusters were ordered: 2,4,1,3
clust4 <- as.character(c(2,4,3,1))
htm <- ComplexHeatmap::Heatmap(t(p4$medoids)[,c(2,4,3,1)],
            row_order = row_order(htc),
            column_split = clust4,
                column_title = "Samples",
                row_title = "Features",
                name = "sdz(log2E)",
                col = jet.colors(32),
                show_column_names = FALSE,
                show_row_names = FALSE)
```

```
## Warning: The heatmap has not been initialized. You might have different results
## if you repeatedly execute this function, e.g. when row_km/column_km was
## set. It is more suggested to do as `ht = draw(ht); row_order(ht)`.
```

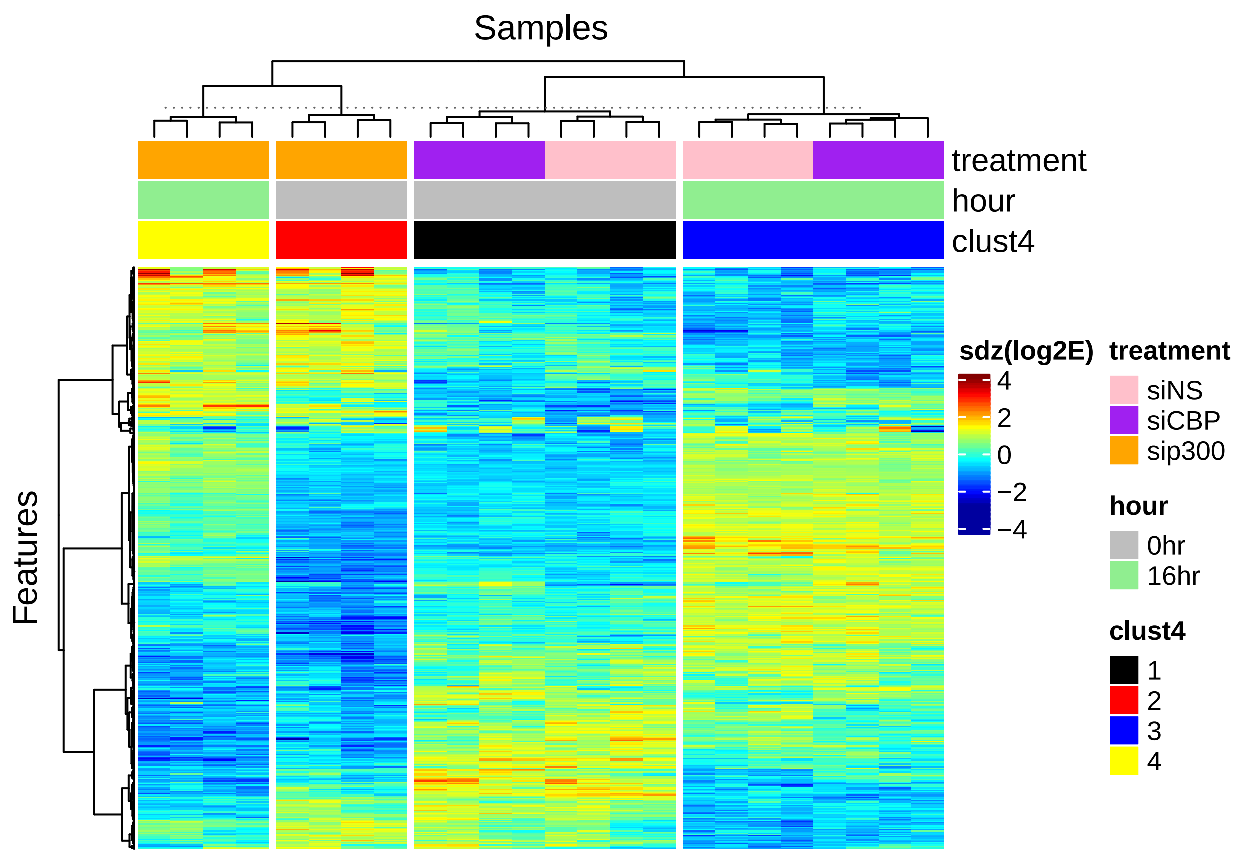
```
draw(htm, padding = unit(c(2, 60, 2, 2), "mm"))
```



Here is how I figured out the cluster order:

```
# column heatmap annotation
clust4 <- as.character(p4$clustering)
dfr <- data.frame(jbcdat$targets[,c("treatment","hour")],
                  clust4 = as.character(p4$clustering))
colha4 <- ComplexHeatmap::HeatmapAnnotation(df = dfr,
                col = list(treatment = c(siNS = "pink",
                                         siCBP = "purple",
                                         sip300 = "orange"),
                           hour = c('0hr' = "grey",
                                    '16hr' = "lightgreen"),
                     clust4 = c("1" = "black",
                            "2" = "red",
                            "3" = "blue",
                            "4" = "yellow")
                    ),
                which = "column")
```

```
htc4 <- ComplexHeatmap::Heatmap(t(X),
           clustering_distance_rows = "pearson",
           clustering_method_rows = "ward.D2",
           column_split = clust4,
              column_title = "Samples",
              row_title = "Features",
              name = "sdz(log2E)",
              col = jet.colors(32),
              top_annotation = colha4,
              show_column_names = FALSE,
              show_row_names = FALSE)
draw(htc4)
```



Other Packages

There is a wealth of packages that have been developed to perform unsupervised data analysis.

https://cran.r-project.org/web/views/Cluster.html

A package cclust advertises doing both kmeans and hard-clustering. Their hard-clustering sounds similar to the method Partioning around medoids above. The authors of the package provide a different approach, bagged clustering, to address the instability of a single solution. Their Bagged clustering approach can be a topic for a class presentation or class project. Is it analogous to consensus clustering?

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
```

```
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] cluster_2.1.2       gg3D_0.0.0.9000     ggplot2_3.3.3
## [4] matrixStats_0.59.0  matlab_1.0.2        ComplexHeatmap_2.6.2
##
## loaded via a namespace (and not attached):
##  [1] circlize_0.4.12     shape_1.4.6         GetoptLong_1.0.5
##  [4] tidyselect_1.1.1    xfun_0.23           purrr_0.3.4
##  [7] tcltk_4.0.3         colorspace_2.0-1    vctrs_0.3.8
## [10] generics_0.1.0      htmltools_0.5.1.1   stats4_4.0.3
## [13] yaml_2.2.1          utf8_1.2.1          rlang_0.4.11
## [16] pillar_1.6.1        glue_1.4.2          withr_2.4.2
## [19] DBI_1.1.1           BiocGenerics_0.36.1 plot3D_1.4
## [22] RColorBrewer_1.1-2  lifecycle_1.0.0     stringr_1.4.0
## [25] munsell_0.5.0       gtable_0.3.0        GlobalOptions_0.1.2
## [28] evaluate_0.14       labeling_0.4.2      misc3d_0.9-0
## [31] knitr_1.33          IRanges_2.24.1      Cairo_1.5-12.2
## [34] parallel_4.0.3      fansi_0.5.0         highr_0.9
## [37] scales_1.1.1        S4Vectors_0.28.1    farver_2.1.0
## [40] rjson_0.2.20        png_0.1-7           digest_0.6.27
## [43] stringi_1.6.2       dplyr_1.0.6         clue_0.3-59
## [46] tools_4.0.3         magrittr_2.0.1      tibble_3.1.2
## [49] crayon_1.4.1        pkgconfig_2.0.3     ellipsis_0.3.2
## [52] assertthat_0.2.1    rmarkdown_2.8       R6_2.5.0
## [55] compiler_4.0.3
```