Class-5-Summary

Jenna G. Tichon

15/10/2019

5 Clustering

5.3 How do we measure similarity?

- **Euclidean**:
$$d(A,B) = \sqrt{(a_1 - b_1)^2 + (a_2 + b_2)^2 + ... + (a_p - b_p)^2}$$
- **Manhattan (L_1)**
$$d(A,B) = |a_1 - b_1| + |a_2 - b_2| + ... + |a_p - b_p|$$
- **Weighter Euclidean Distance**
- **Minkowski**
$$d(A,B) = ((a_1 - b_1)^m + (a_2 - b_2)^m + ... + (a_p - b_p)^m)\frac{1}{m}$$
- **BinARY Bit** For vectors of binary components, proportion of features with exactly one bit on against those with at least one on.
- **Jacard Distance** f_{11} number of time a feature co-occurs in S and T and f_{10}/f_{01} the number of times it occurs in exactly one.
$$J(S,T) = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$
$$d_J(S.T) = 1 - J(S,T) = \frac{f_{01} + f_{10}}{f_{01} + f_{10} + f_{11}}$$
- **COrrelation Based Distance**
$$d(A,B) = \sqrt{2(1 - cor(A,B))}$$

5.3.1 Computations RElated to Distances in R

```
mx  = c(0, 0, 0, 1, 1, 1)
my  = c(1, 0, 1, 1, 0, 1)
mz  = c(1, 1, 1, 0, 1, 1)
mat = rbind(mx, my, mz)
dist(mat)
```

```
##          mx       my
## my 1.732051
## mz 2.000000 1.732051
```

We can specify the distance method.

```
dist(mat, method = "binary")
```

```
##           mx        my
## my 0.6000000
## mz 0.6666667 0.5000000
```

Showing that this agrees with euclidean distance

```
load(here("data","Morder.RData"))
sqrt(sum((Morder[1, ] - Morder[2, ])^2))
```

```
## [1] 5.593667
```

```
as.matrix(dist(Morder))[2, 1]
```

```
## [1] 5.593667
```

Now let's compare HIV mutation data using the Jacard distance (using the `vegdist` function in the pacakge `vegan`) and the correlation based distance.

```
mut = read.csv(here("data", "HIVmutations.csv"))
mut[1:3, 10:16]
```

```
##   p32I p33F p34Q p35G p43T p46I p46L
## 1    0    1    0    0    0    0    0
## 2    0    1    0    0    0    1    0
## 3    0    1    0    0    0    0    0
```

```
#Jacard Distance
library("vegan")
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.5-6
```

```
mutJ = vegdist(mut, "jaccard")
mutJ
```

```
##           1         2         3         4
## 2 0.8000000
## 3 0.7500000 0.8888889
## 4 0.9000000 0.7777778 0.8461538
## 5 1.0000000 0.8000000 0.8888889 0.9000000
```

```
#Correlation based distance
mutC = sqrt(2*(1-cor(t(mut))))
as.dist(mutC)
```

```
##          1        2        3        4
## 2 1.186342
## 3 1.104026 1.302931
## 4 1.318368 1.133893 1.298780
## 5 1.452966 1.186342 1.302931 1.318368
```

5.4 Nonparametic Mixture Detection

4.4.1 k-methods: m-means, k-medoids and PAM

PAM (partitioning around medoids)

1. Matrix with p features on n observations

2. Randomly pick k distinct *cluster centers* from the n observations (these are our initial seeds)
3. Assign each other observation to closest center
4. Find the observation in each group that minimizes the sum of distances to that observation in the group. Called the *medoid*
5. Repeat 3 & 4 until groups stabilize.

This is implemented in the `PAM` package.

k-means makes the arithmetic mean the medoids which may or may not be an actual observation. This is in the bases `stats` package that come with R

5.4.2 Tight clusters with resampling

Tight clusters are points that are almost always grouped together. This is the example vignette from the package `clusterExperiment` using `clusterMany` as the clustering function and `pam` to run the individual clusters.

Change the choice of genes at 60, 100, 150 and the number of clusters k for 4 to 9.

```
library("clusterExperiment")
```

```
## Loading required package: SingleCellExperiment
## Loading required package: SummarizedExperiment
## Loading required package: GenomicRanges
## Loading required package: stats4
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter,
##     Find, get, grep, grepl, intersect, is.unsorted, lapply, Map,
##     mapply, match, mget, order, paste, pmax, pmax.int, pmin,
##     pmin.int, Position, rank, rbind, Reduce, rownames, sapply,
##     setdiff, sort, table, tapply, union, unique, unsplit, which,
##     which.max, which.min
## Loading required package: S4Vectors
```

```
##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## Loading required package: GenomeInfoDb

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: DelayedArray

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians

## The following object is masked from 'package:dplyr':
##
##     count

## Loading required package: BiocParallel

##
## Attaching package: 'DelayedArray'

## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges

## The following objects are masked from 'package:base':
##
##     aperm, apply, rowsum

## Warning: namespace 'bigmemory' is not available and has been replaced
## by .GlobalEnv when processing object 'Morder'

## Warning: namespace 'bigmemory' is not available and has been replaced
## by .GlobalEnv when processing object 'Morder'
```

```
data("fluidigm", package = "scRNAseq")
```

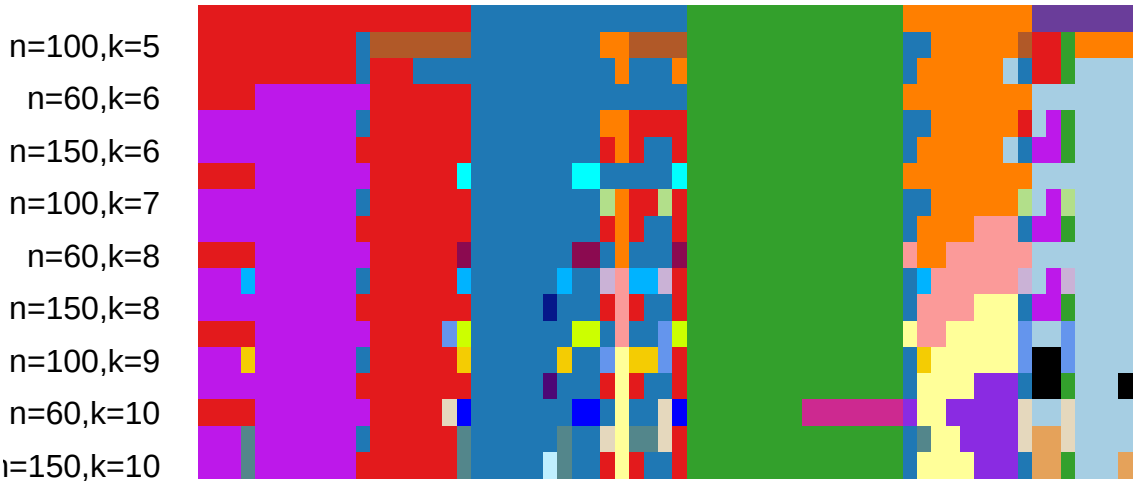
```
## Warning: 'data(fluidigm)' is deprecated.
## Use ReprocessedFluidigmData() instead.
```

```
## adding rname 'https://github.com/LTLA/scRNAseq/raw/80b3d4dde812b72406499922c5cd157e7b8d3b45/data/flu
```

```
se = fluidigm[, fluidigm$Coverage_Type == "High"]
assays(se) = list(normalized_counts =
   round(limma::normalizeQuantiles(assay(se))))
ce = clusterMany(se, clusterFunction = "pam", ks = 5:10, run = TRUE,
  isCount = TRUE, reduceMethod = "var", nFilterDims = c(60, 100, 150))
```

```
## Note: Not all of the methods requested in 'reduceMethod' have been calculated. Will calculate all the
```

```
clusterLabels(ce) = sub("FilterDims", "", clusterLabels(ce))
plotClusters(ce, whichClusters = "workflow", axisLine = -1)
```



5.5 Clustering Examples: flow cytometry and mass cytometry

5.5.1 Flow cytometry and mass cytometry

*__Bio tip__: "At different stages of their development, immune cells express unique combinations of proteins on their surfaces. These protein-markers are called CDs (clusters of differentiation) collected using flow or mass cytometry

Some CD4 is protein expressed by helper t-cells (Called CD4+) but some cells express this without being helper t cells

ALERT!!: I've given up on installing flowViz but got flowCore installed.

```
library("flowCore")
```

```
##
## Attaching package: 'flowCore'
```

```
## The following object is masked from 'package:BiocGenerics':
##
```

```
##     normalize
```

```
library("flowViz")
fcsB = read.FCS("../data/Bendall_2011.fcs")
slotNames(fcsB)
```

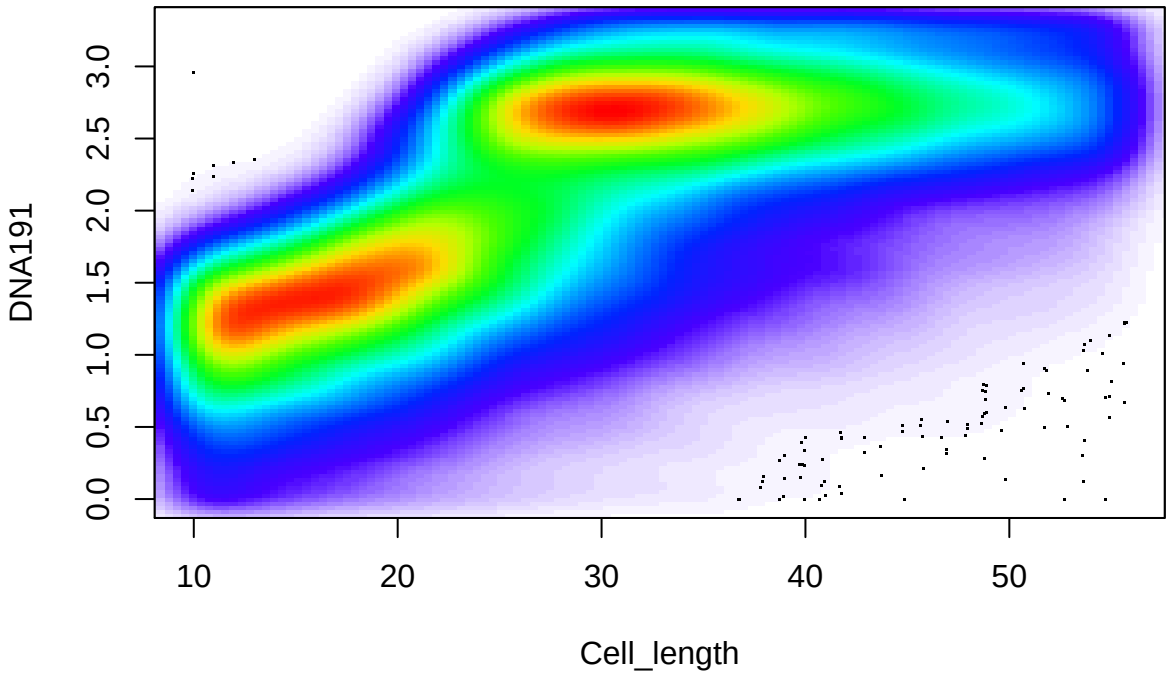
```
## [1] "exprs"       "parameters"  "description"
```

5.5.2 Data processing

```
fcsB = read.FCS(here("data", "Bendall_2011.fcs"))
library("flowCore")
markersB = readr::read_csv(here("data","Bendall_2011_markers.csv"))
```

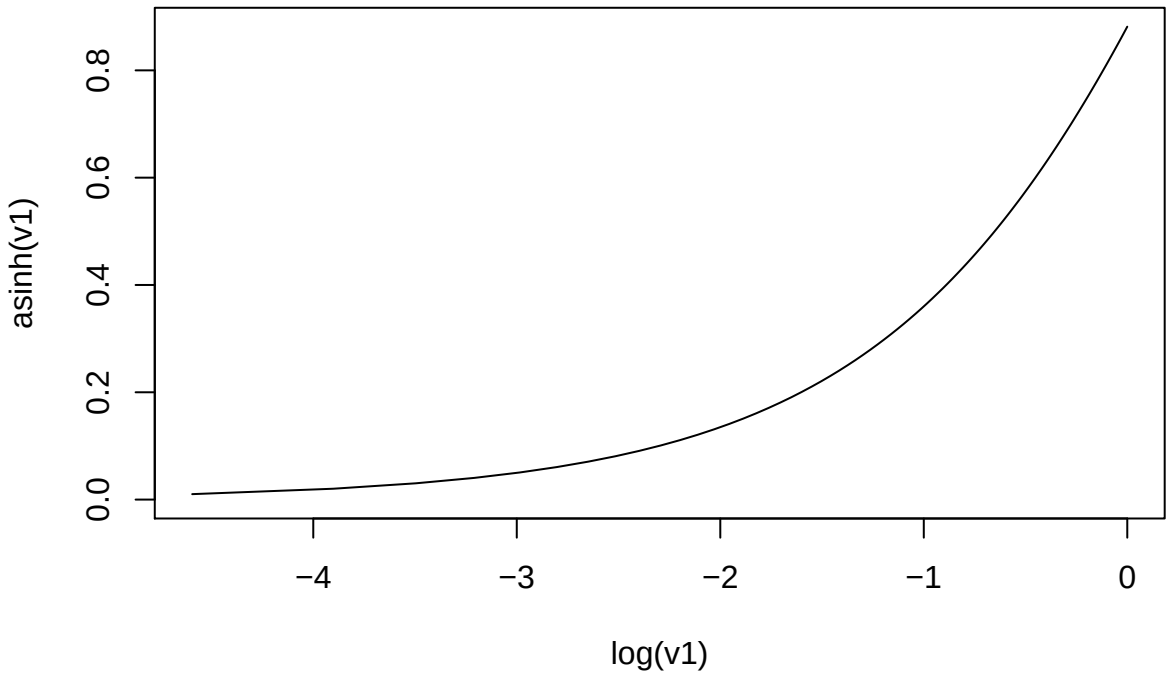
```
## Parsed with column specification:
## cols(
##   isotope = col_character(),
##   marker = col_character()
## )
```

```
#Replace columns names with marker names
mt = match(markersB$isotope, colnames(fcsB))
stopifnot(!any(is.na(mt)))
colnames(fcsB)[mt] = markersB$marker
flowPlot(fcsB, plotParameters = colnames(fcsB)[2:3], logy = TRUE)
```

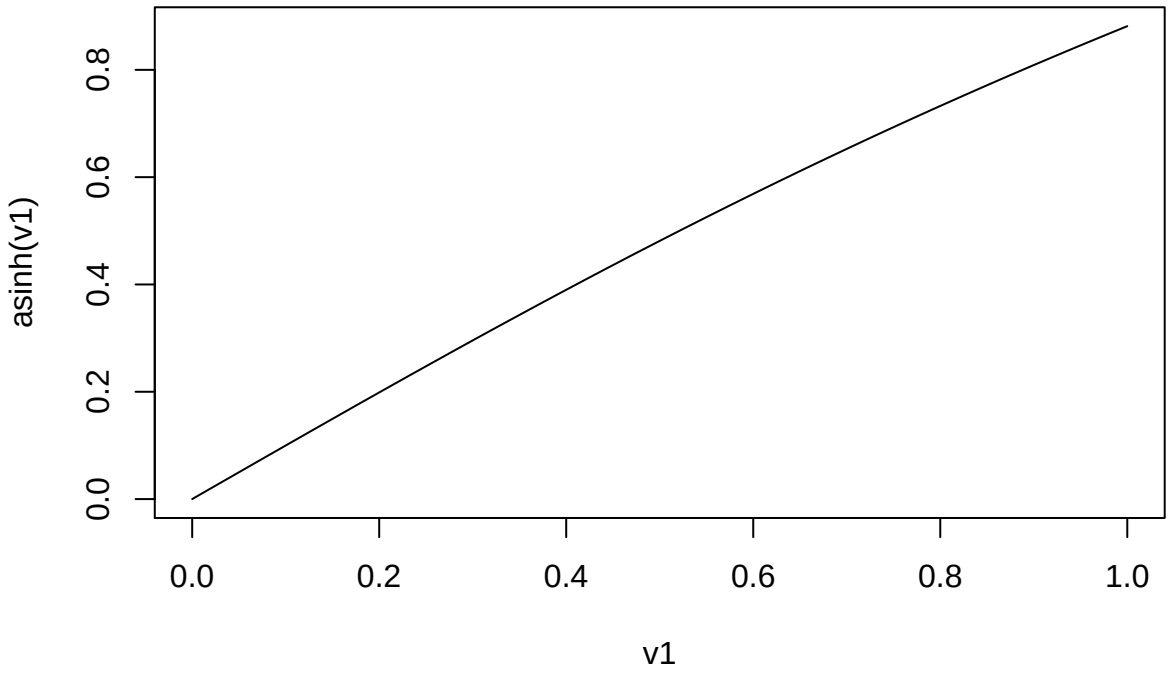


arcsin is a common transformation for flow and mass cytometry.

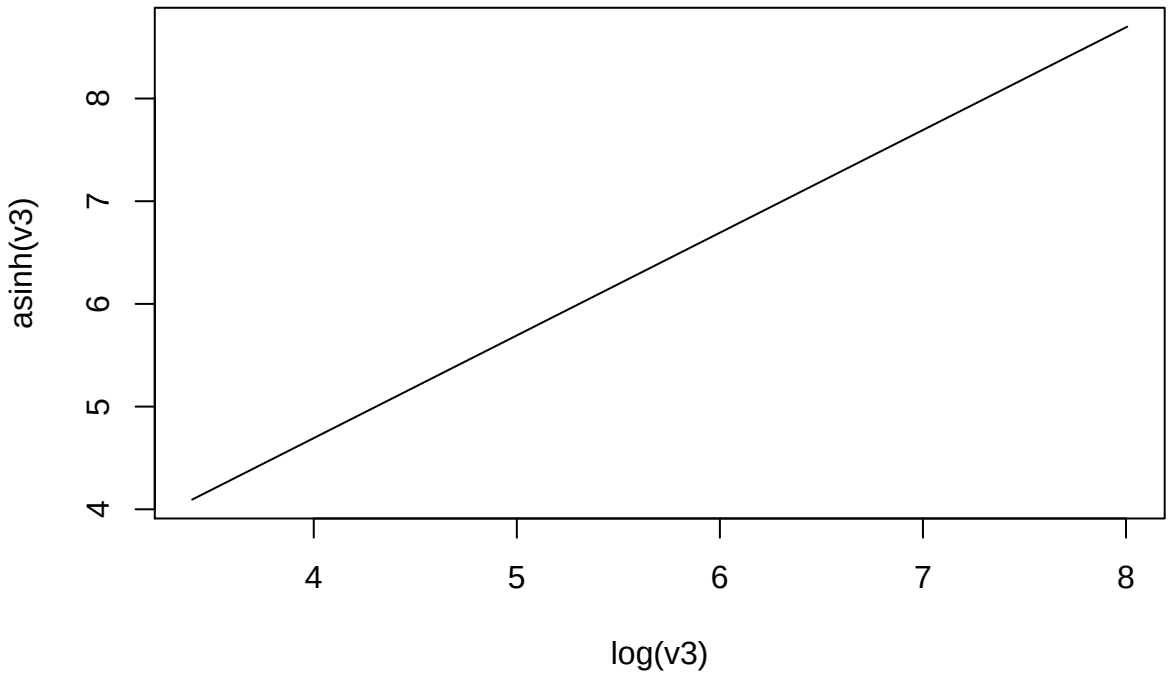
```
v1 = seq(0, 1, length.out = 100)
plot(log(v1), asinh(v1), type = 'l')
```



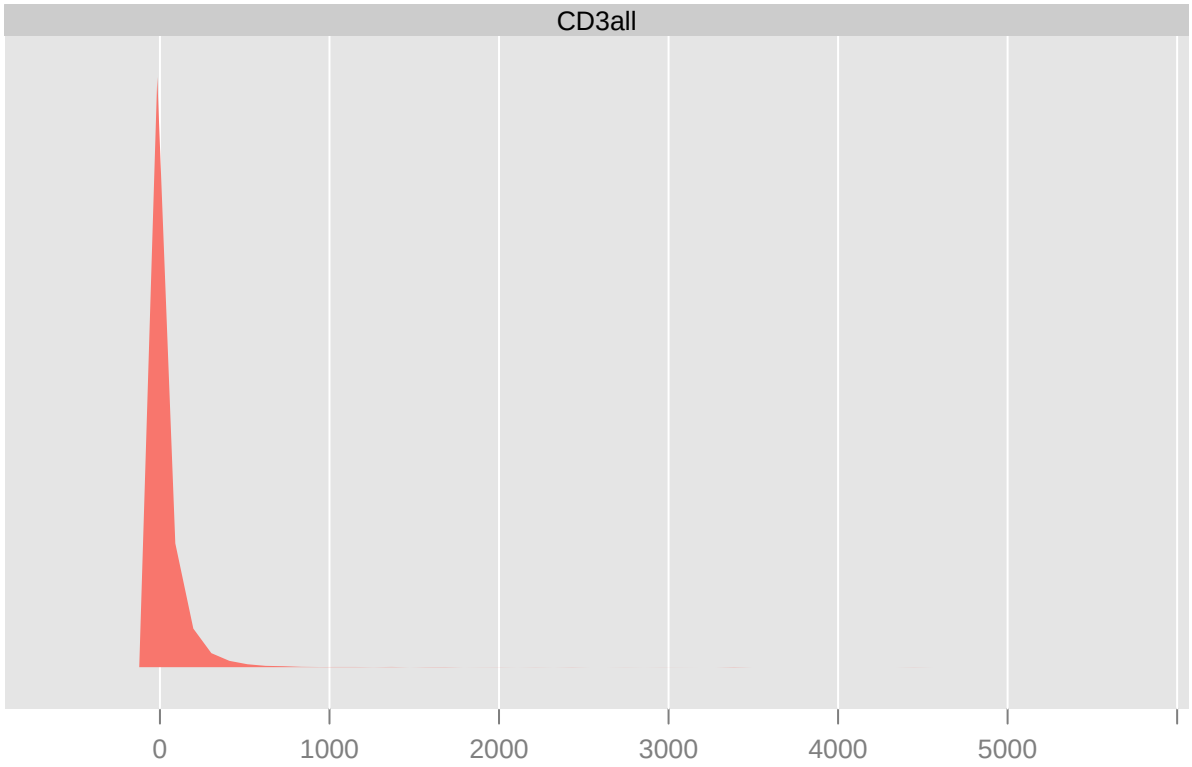

```
plot(v1, asinh(v1), type = 'l')
```



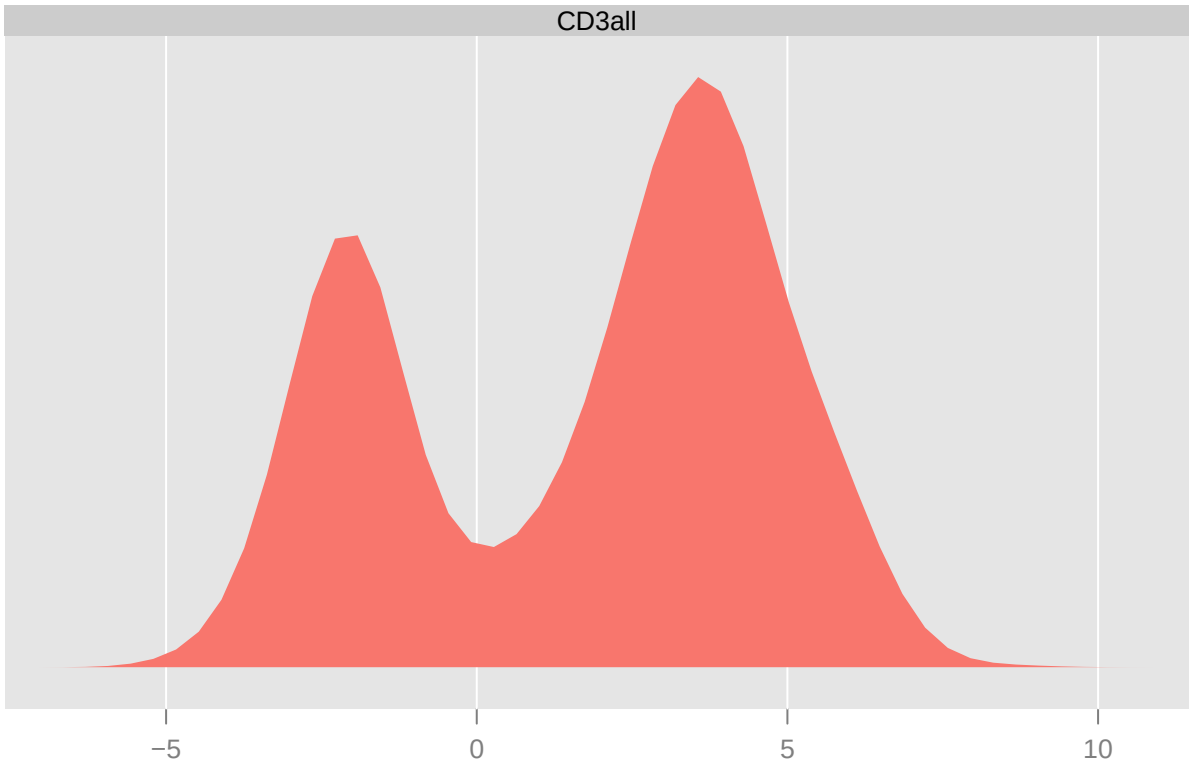

```
v3 = seq(30, 3000, length = 100)
plot(log(v3), asinh(v3), type= 'l')
```




```
#apply arcsinh transformation
asinhtrsf = arcsinhTransform(a = 0.1, b = 1)
fcsBT = transform(fcsB,
transformList(colnames(fcsB)[-c(1, 2, 41)], asinhtrsf))
densityplot( ~`CD3all`, fcsB)
```



```
densityplot( ~`CD3all`, fcsBT)
```



ALERT!!: Screw this package

```
kf = kmeansFilter("CD3all" = c("Pop1","Pop2"), filterId="myKmFilter")
fres = flowCore::filter(fcsBT, kf)
summary(fres)
```

```
## Pop1: 33429 of 91392 events (36.58%)
## Pop2: 57963 of 91392 events (63.42%)
```

5.5.3 Density-based clustering

Few markers and large number of cells is suited for density based clustering. Find high density separated by sparser regions.

```
library("dbscan")
fcsB = read.FCS(here("data","Bendall_2011.fcs"))
markersB = readr::read_csv(here("data","Bendall_2011_markers.csv"))
```

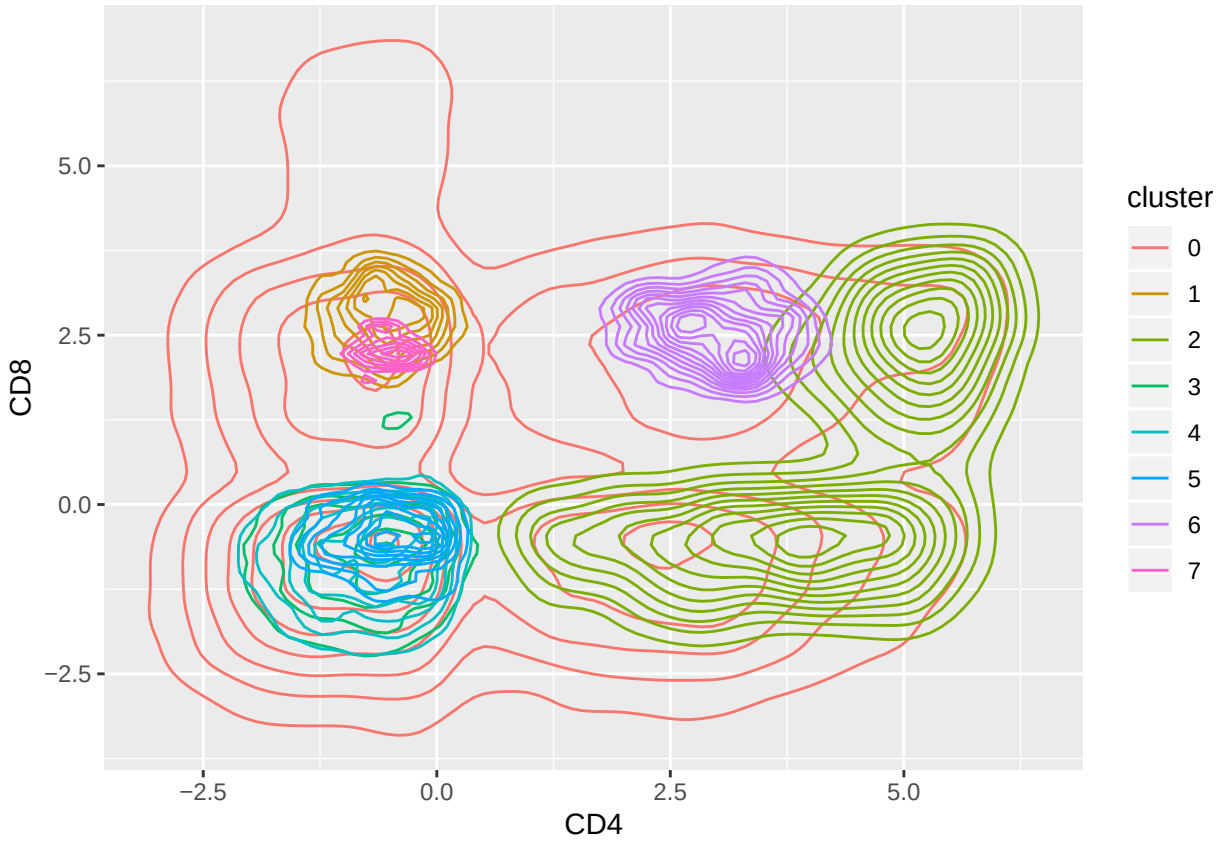
```
## Parsed with column specification:
## cols(
##   isotope = col_character(),
##   marker = col_character()
## )
```

```
#Replace columns names with marker names
mt = match(markersB$isotope, colnames(fcsB))
stopifnot(!any(is.na(mt)))
```

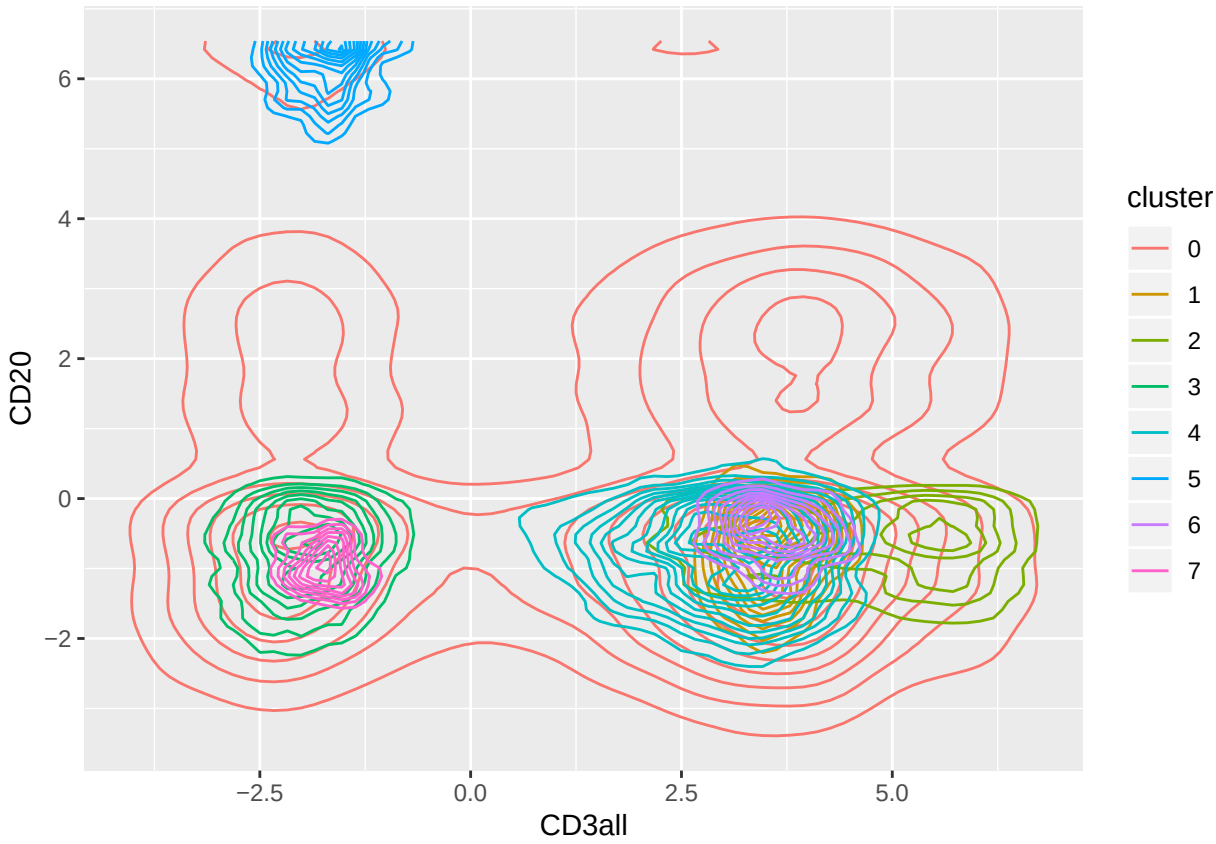
```
colnames(fcsB)[mt] = markersB$marker
mc5 = Biobase::exprs(fcsBT)[, c(15,16,19,40,33)]
res5 = dbscan::dbscan(mc5, eps = 0.65, minPts = 30)
mc5df = data.frame(mc5, cluster = as.factor(res5$cluster))
table(mc5df$cluster)
```

```
##
##     0     1     2     3     4     5     6     7
## 77655   230  5114  4616  3310   207   231    29
```

```
#Visualized grouped by cluster
ggplot(mc5df, aes(x=CD4,    y=CD8,  col=cluster))+geom_density2d()
```




```
ggplot(mc5df, aes(x=CD3all, y=CD20, col=cluster))+geom_density2d()
```



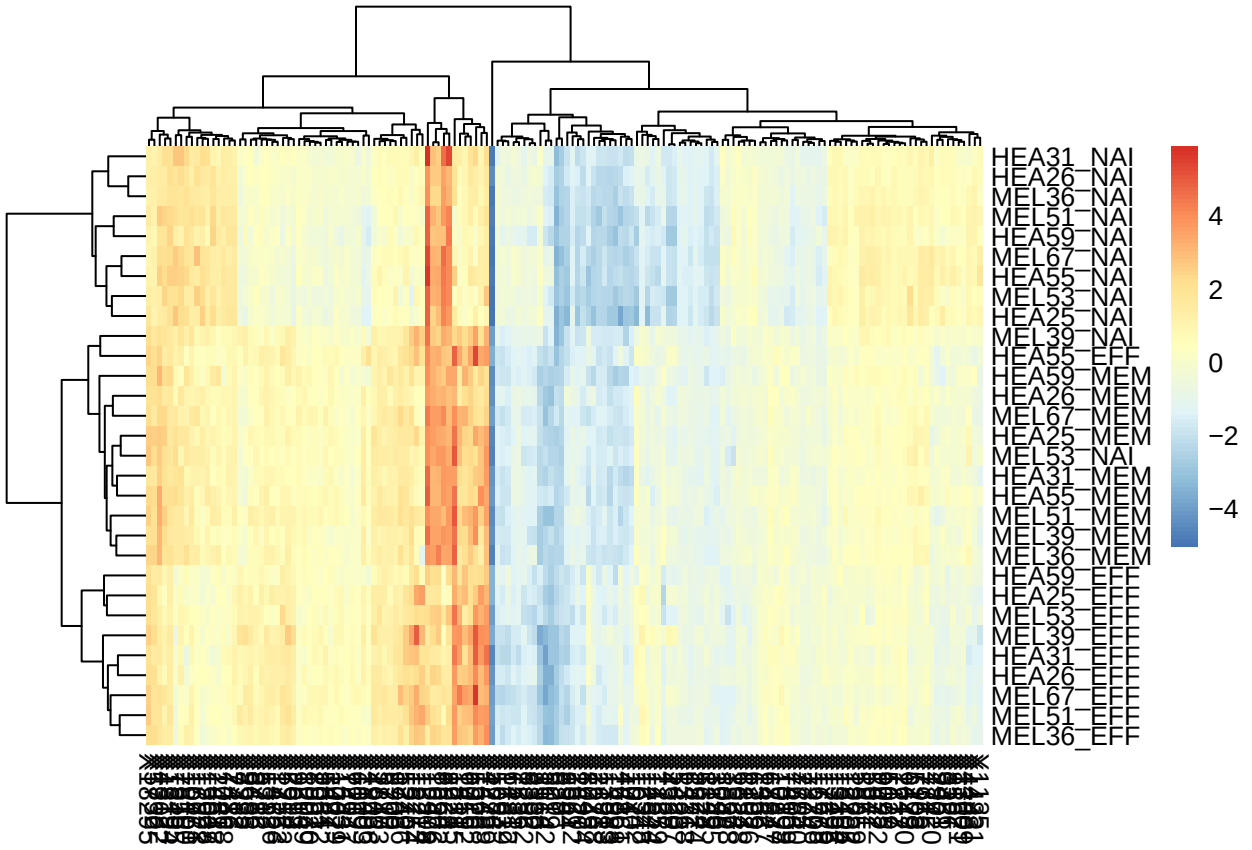
5.6 Hierarchical clustering

5.6.1 How to compute (dis)similarities between aggregated clusters?

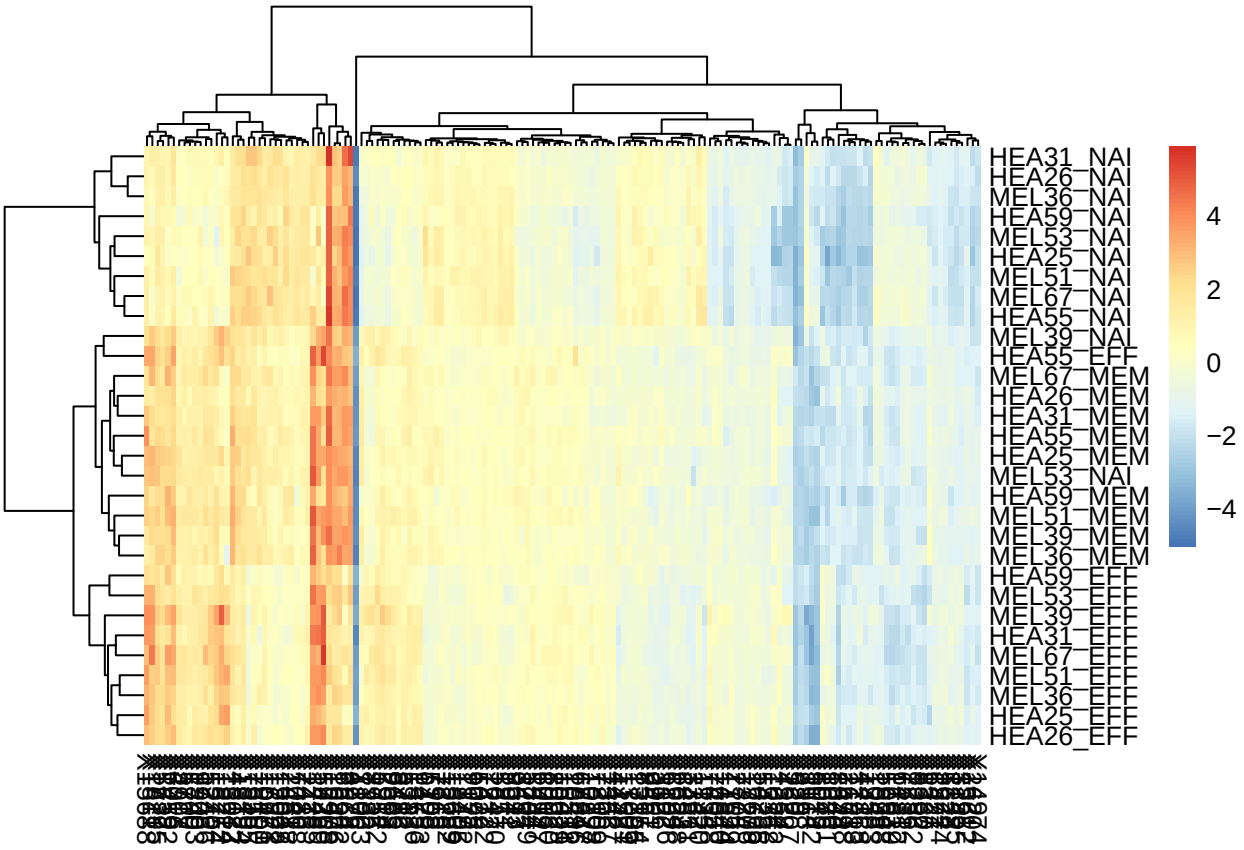
We can compute this by: minimal jump, maximal jump, average linkage, ward's method using ANOVA approach.

Q5.8

```
library(pheatmap)
load(here("data", "Morder.RData"))
pheatmap(Morder, clustering_distance_cols = "euclidean", clustering_distance_rows = "euclidean")
```




```
pheatmap(Morder, clustering_distance_cols = "manhattan", clustering_distance_rows = "manhattan")
```



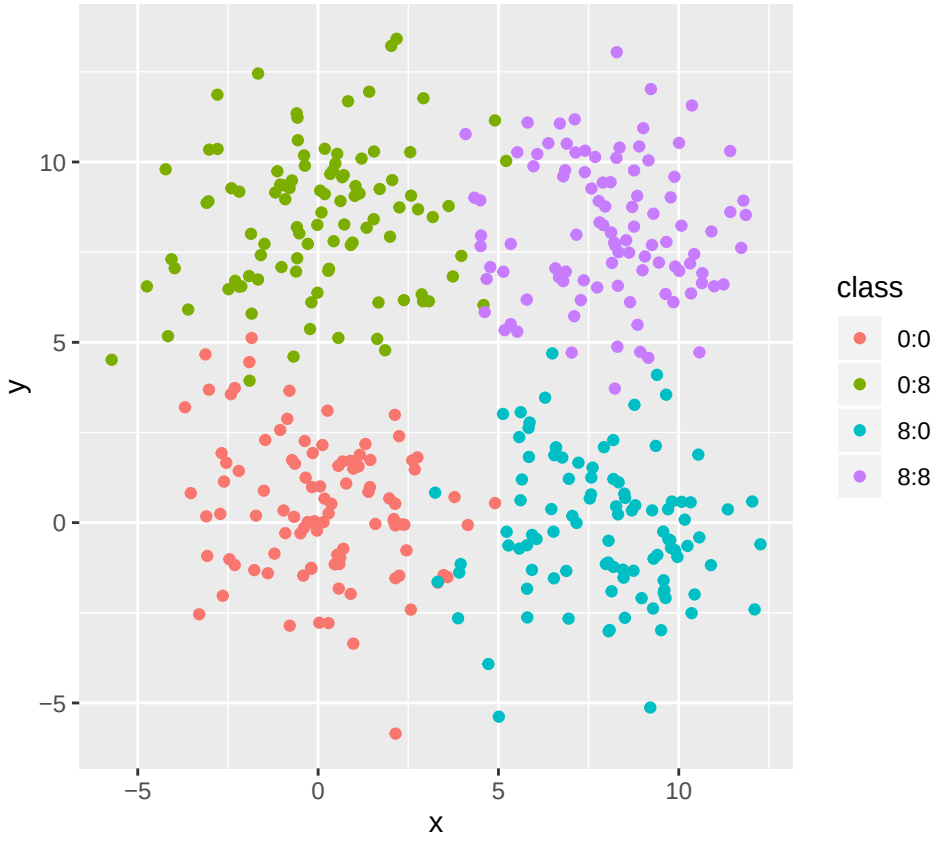
5.7 Validating and finding members of clusters

```
library("dplyr")

#Simulate 100 each data from groups where the mu_X and mu_Y Combos are: (0,0), (0,8), (8,0), (8,8)
simdat = lapply(c(0, 8), function(mx) {
  lapply(c(0,8), function(my) {
    tibble(x = rnorm(100, mean = mx, sd = 2),
           y = rnorm(100, mean = my, sd = 2),
           #column that says which group data is from
           class = paste(mx, my, sep = ":"))
   }) %>% bind_rows
}) %>% bind_rows
simdat
```

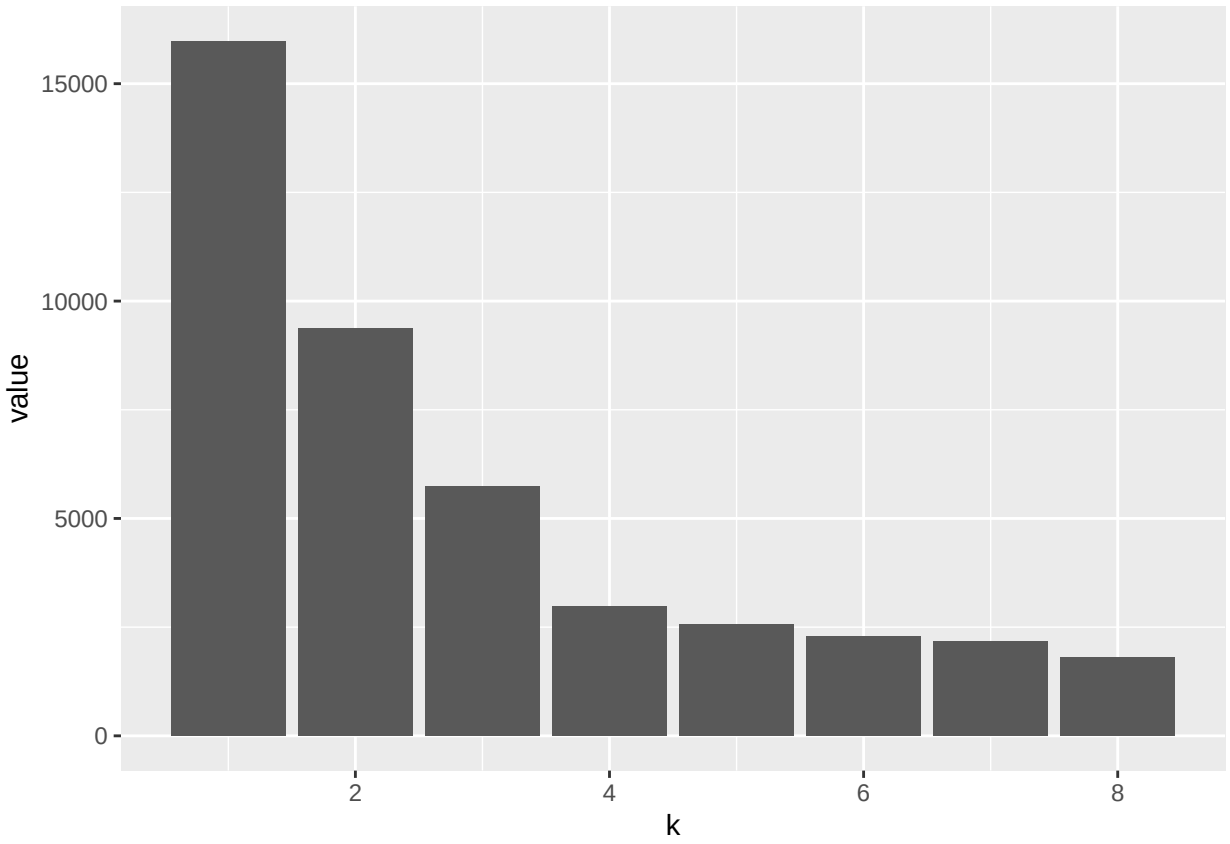
```
## # A tibble: 400 x 3
##         x      y class
##     <dbl>  <dbl> <chr>
##  1  0.573 -1.83  0:0
##  2 -2.46  -1.02  0:0
##  3  0.588 -1.15  0:0
##  4  2.13   0.526 0:0
##  5 -2.42   3.56  0:0
##  6 -0.666  0.159 0:0
##  7  0.979  1.50  0:0
```

```
##  8 -2.54   1.66  0:0
##  9 -0.721  1.74  0:0
## 10 -1.05   2.57  0:0
## # ... with 390 more rows
```

```
#Remoce column with (0,0), (0,8), (8,0), (8,8) label
simdatxy = simdat[, c("x", "y")]

#Create a scatterplot colored by class
ggplot(simdat, aes(x = x, y = y, col = class)) + geom_point() +
  coord_fixed()
```




```
#Create a tiblle with number of groups 1:8, and blank for wss (Within group sum of square value)
wss = tibble(k = 1:8, value = NA_real_)
#set for 1 group, the value of just the data set lumped together
wss$value[1] = sum(scale(simdatxy, scale = FALSE)^2)
#calculate it for 2:8
for (i in 2:nrow(wss)) {
  #use kmeans of simdatxy with the number of centers equal to k
  km  = kmeans(simdatxy, centers = wss$k[i])
  #replace wss value in tibble with withinss parameter from kmeans object
  wss$value[i] = sum(km$withinss)
}
#Make a boxchart of wss, see where it makes an elbow: AT 4!
ggplot(wss, aes(x = k, y = value)) + geom_col()
```



Q5.12

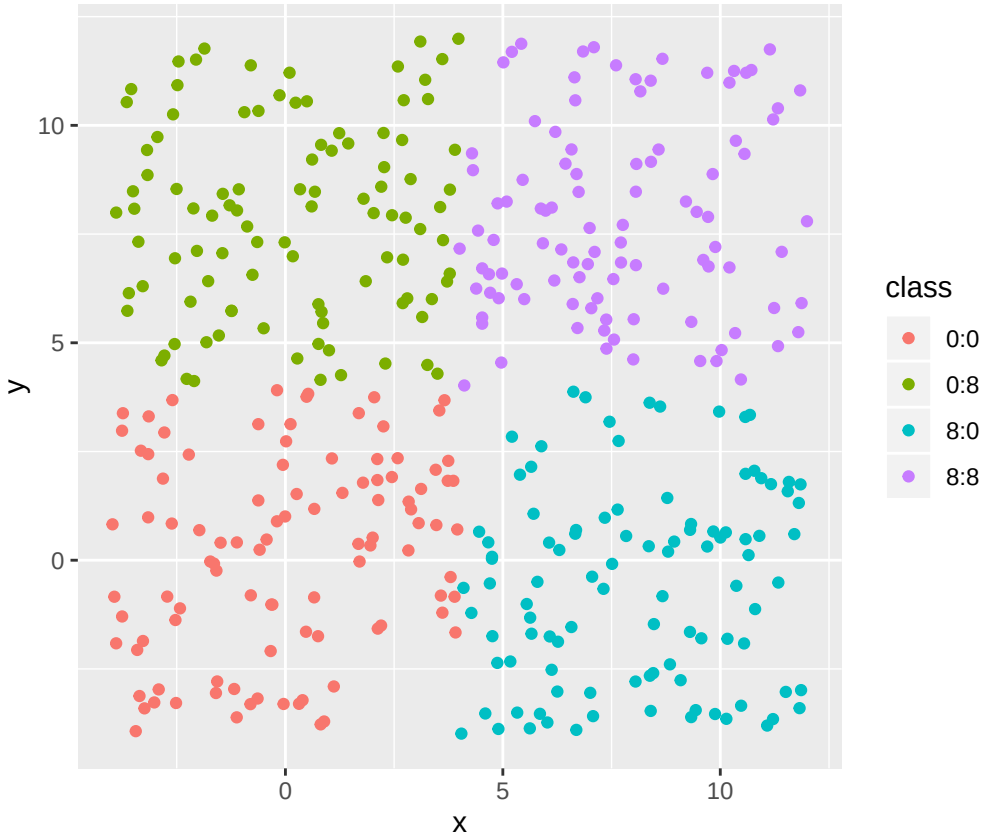
Redo with uniform

```
library("dplyr")

#Simulate 100 each data from groups where the mu_X and mu_Y Combos are: (0,0), (0,8), (8,0), (8,8)
simdat2 = lapply(c(0, 8), function(mx) {
  lapply(c(0,8), function(my) {
    tibble(x = runif(100, min = mx-4, max = mx+4),
           y = runif(100, min = my-4, max = my+4),
           #column that says which group data is from
           class = paste(mx, my, sep = ":"))
   }) %>% bind_rows
}) %>% bind_rows
simdat2
```

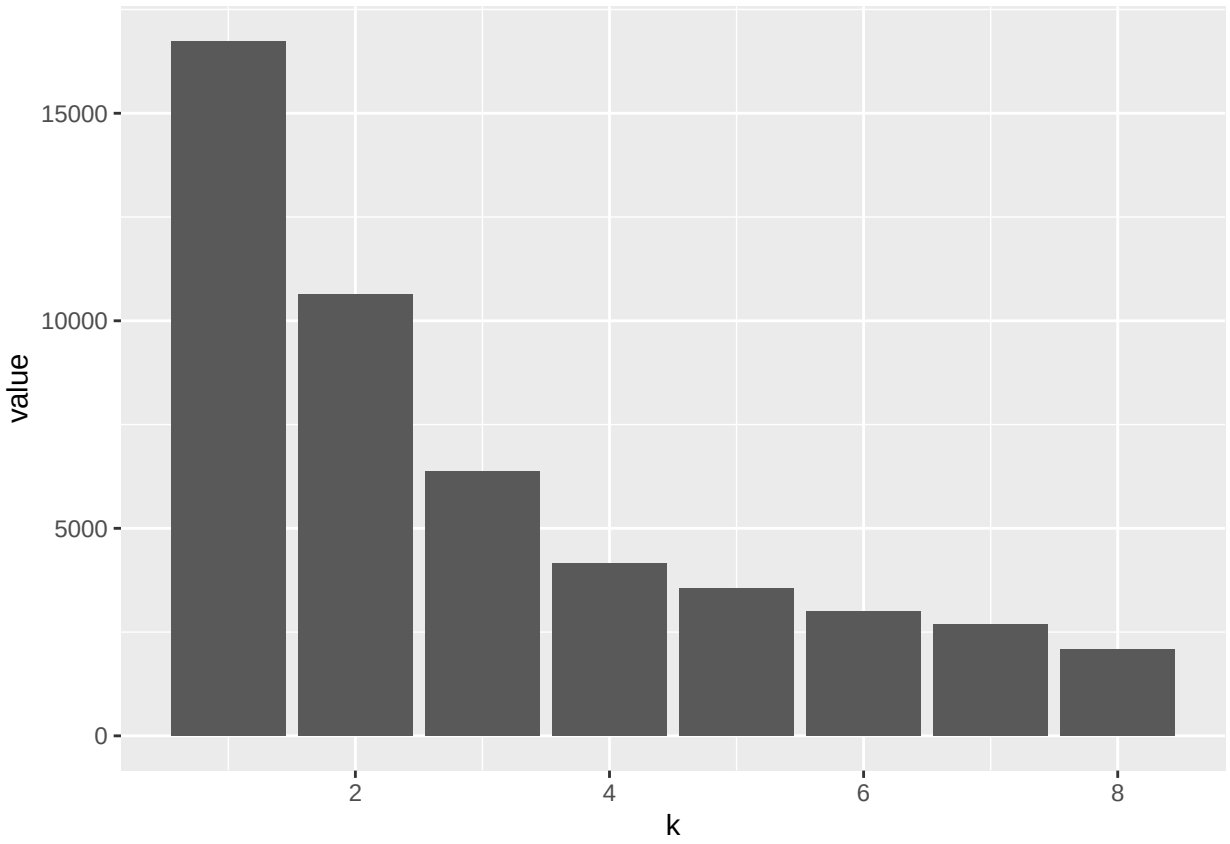
```
## # A tibble: 400 x 3
##         x      y class
##     <dbl>  <dbl> <chr>
##  1  1.69   3.38  0:0
##  2  3.47   0.809 0:0
##  3  3.73   1.83  0:0
##  4 -3.14   3.31  0:0
##  5 -3.15   2.44  0:0
##  6 -1.12  -3.61  0:0
##  7  0.661 -0.854 0:0
```

```
##  8 -3.44  -3.93  0:0
##  9  3.66   3.68  0:0
## 10  3.58  -0.813 0:0
## # ... with 390 more rows
```

```
#Remoce column with (0,0), (0,8), (8,0), (8,8) label
simdat2xy = simdat2[, c("x", "y")]

#Create a scatterplot colored by class
ggplot(simdat2, aes(x = x, y = y, col = class)) + geom_point() +
  coord_fixed()
```




```
#Create a tiblle with number of groups 1:8, and blank for wss (Within group sum of square value)
wss = tibble(k = 1:8, value = NA_real_)
#set for 1 group, the value of just the data set lumped together
wss$value[1] = sum(scale(simdat2xy, scale = FALSE)^2)
#calculate it for 2:8
for (i in 2:nrow(wss)) {
  #use kmeans of simdatxy with the number of centers equal to k
  km  = kmeans(simdat2xy, centers = wss$k[i])
  #replace wss value in tibble with withinss parameter from kmeans object
  wss$value[i] = sum(km$withinss)
}
#Make a boxchart of wss, see where it makes an elbow: AT 4!
ggplot(wss, aes(x = k, y = value)) + geom_col()
```



5.7.1 Using the gap statistic

We can compare $\log(WSS_k)$ for various k values and compare to that obtained for data with similar dimensions that is uniform and non-clustered.

This gap statistic is:

$$gap(k) = \frac{1}{B}\sum_{b=1}^{B} \log W^*_{kb} - \log WSS_k$$

This will be positive if the clustering is good because clustering should minimize the WSS (the within sum of squares). SO we want the highest gap value.

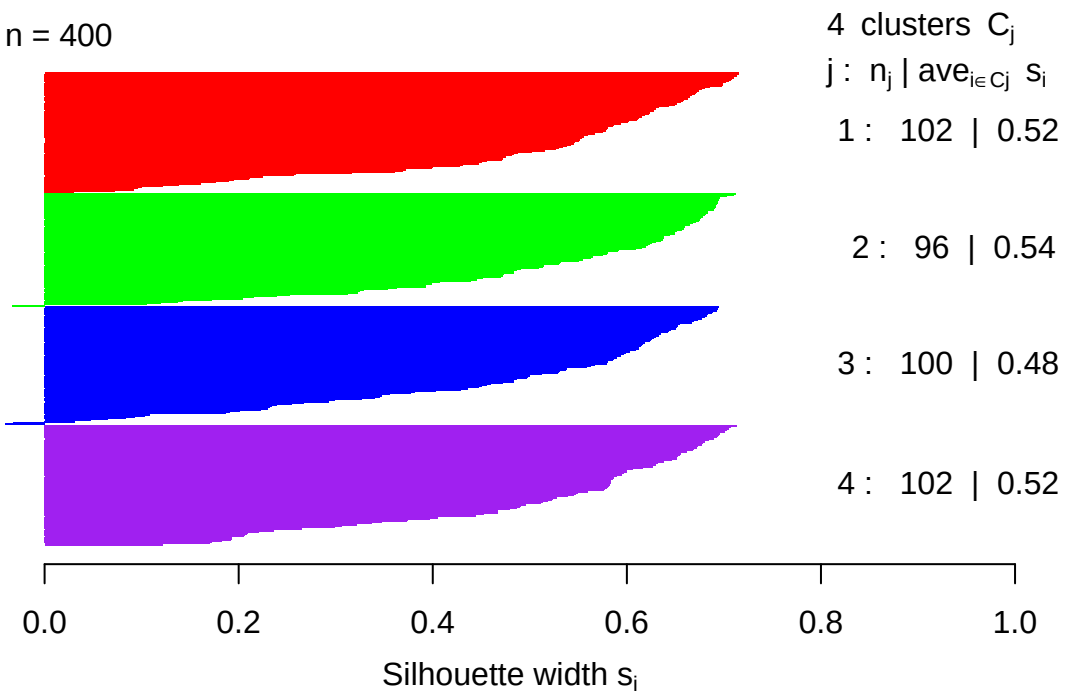
5.14

```
{r} # library("Hiiragi2013") # data("x") #  # #Find more variable genes? # selFeats = order(rowVars(Biobase::exprs(x)), decreasing = TRUE)[1:50] # embmat = t(Biobase::exprs(x)[selFeats, ]) # embgap = clusGap(embmat, FUN = pamfun, K.max = 24, verbose = FALSE) # #what k is not larger than first local maxima minus sampling error # k1 = maxSE(embgap$Tab[, "gap"], embgap$Tab[, "SE.sim"]) # #what k under tibshirani, walther, hastie # k2 = maxSE(embgap$Tab[, "gap"], embgap$Tab[, "SE.sim"], #            method = "Tibs2001SEmax") # c(k1, k2) #  # #Plot gap statistic # plot(embgap, main = "") # cl = pamfun(embmat, k = k1)$cluster # table(pData(x)[names("sampleGroup"], cl) #
```

5.11 Exercises

5.1a

```
library("cluster")
pam4 = pam(simdatxy, 4)
sil = silhouette(pam4, 4)
plot(sil, col=c("red", "green", "blue", "purple"), main="Silhouette")
```

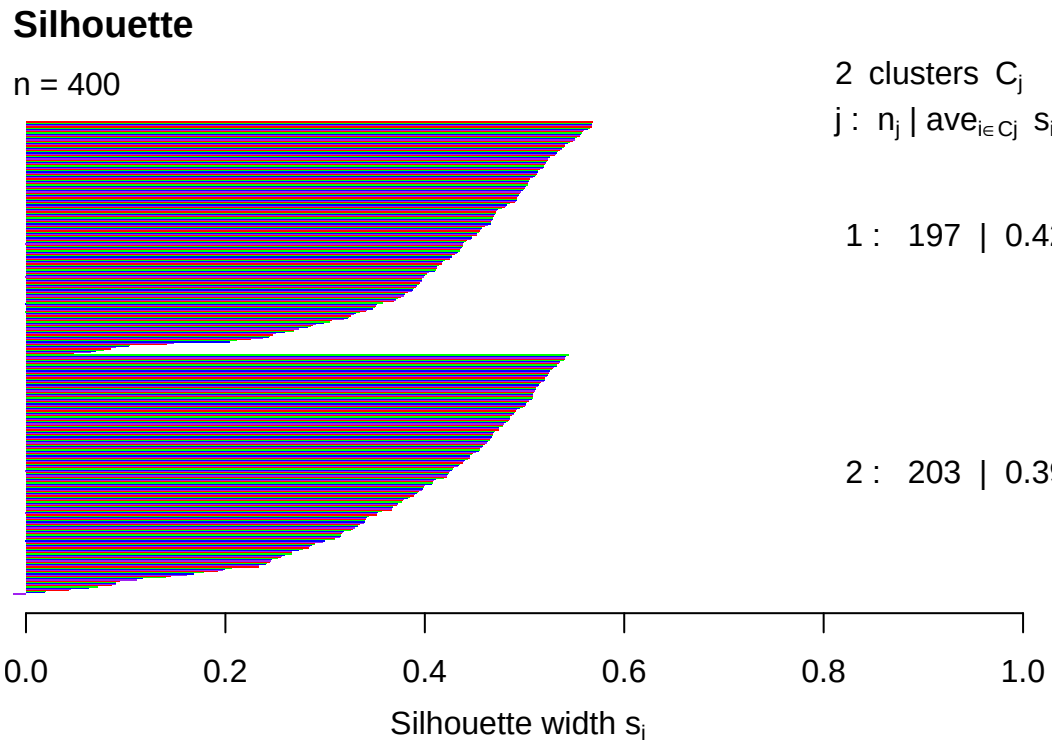


Average silhouette width : 0.51

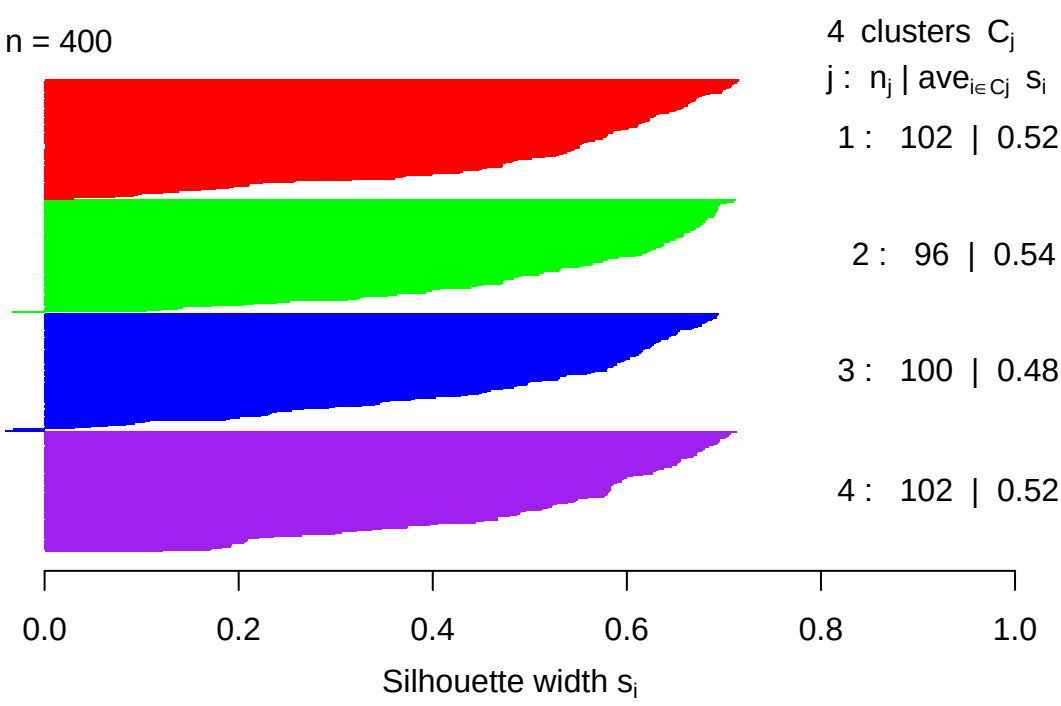
5.1.b

Four has the highest sil values

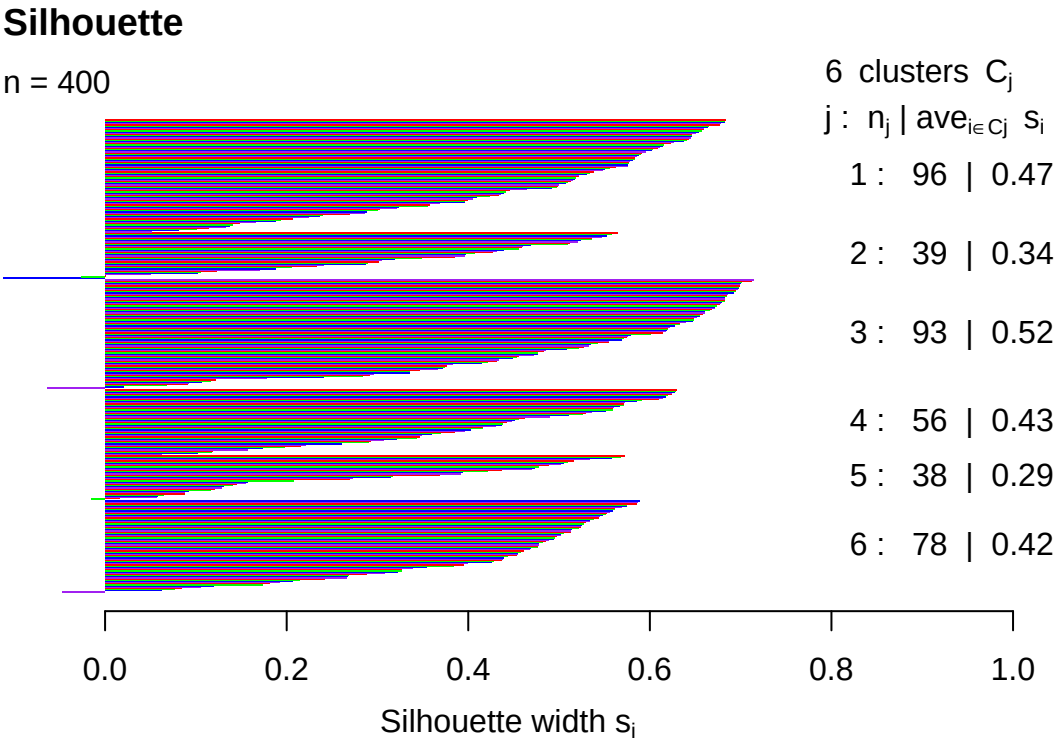
```
for(k in 2:8){
sil = silhouette(pam(simdatxy,k), k)
plot(sil, col=c("red", "green", "blue", "purple"), main="Silhouette")
}
```


Silhouette
n = 400
2 clusters C_j
j : n_j | $ave_{i \in Cj}$ s_i
1 : 197 | 0.42
2 : 203 | 0.39
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width s_i
Average silhouette width : 0.4

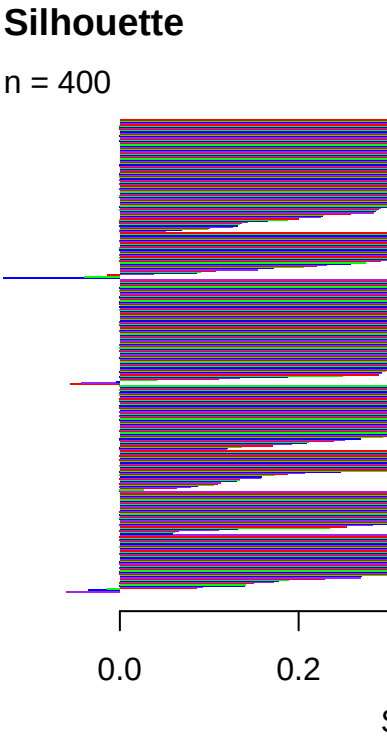

Silhouette
n = 400
0.0
0.2
Average silhouette width :


Silhouette
n = 400
4 clusters C_j
j : n_j | $ave_{i \in Cj}$ s_i
1 : 102 | 0.52
2 : 96 | 0.54
3 : 100 | 0.48
4 : 102 | 0.52
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width s_i
Average silhouette width : 0.51

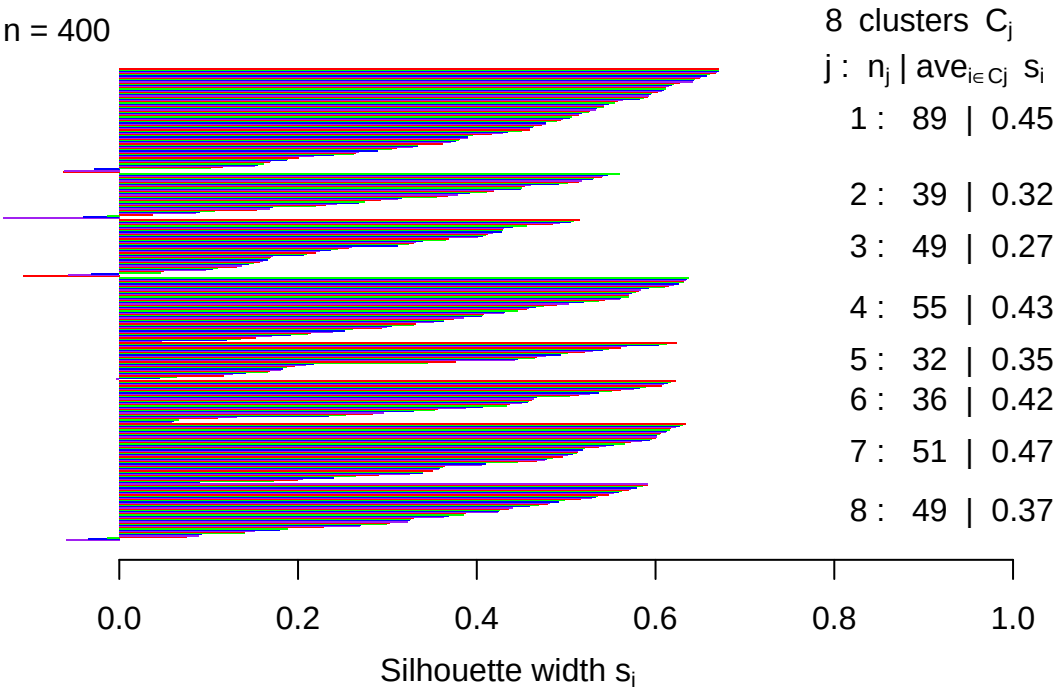

Silhouette
n = 400
0.0
0.2
Average silhouette width :



Average silhouette width : 0.44



Average silhouette width :

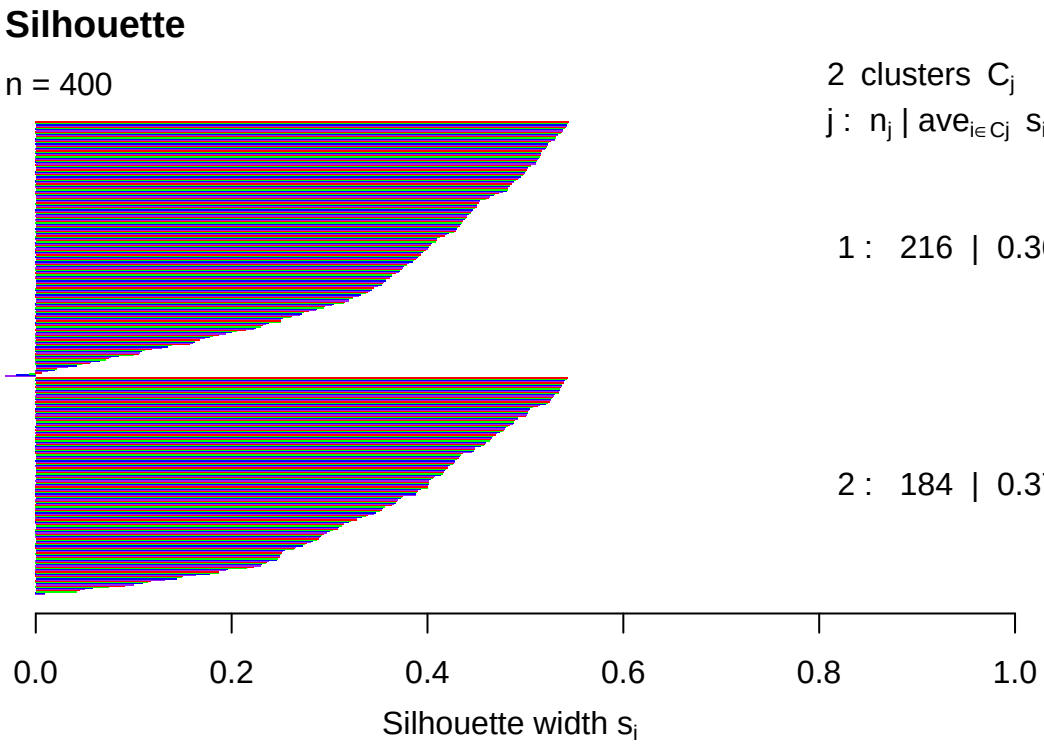


Average silhouette width : 0.39

5.1.c

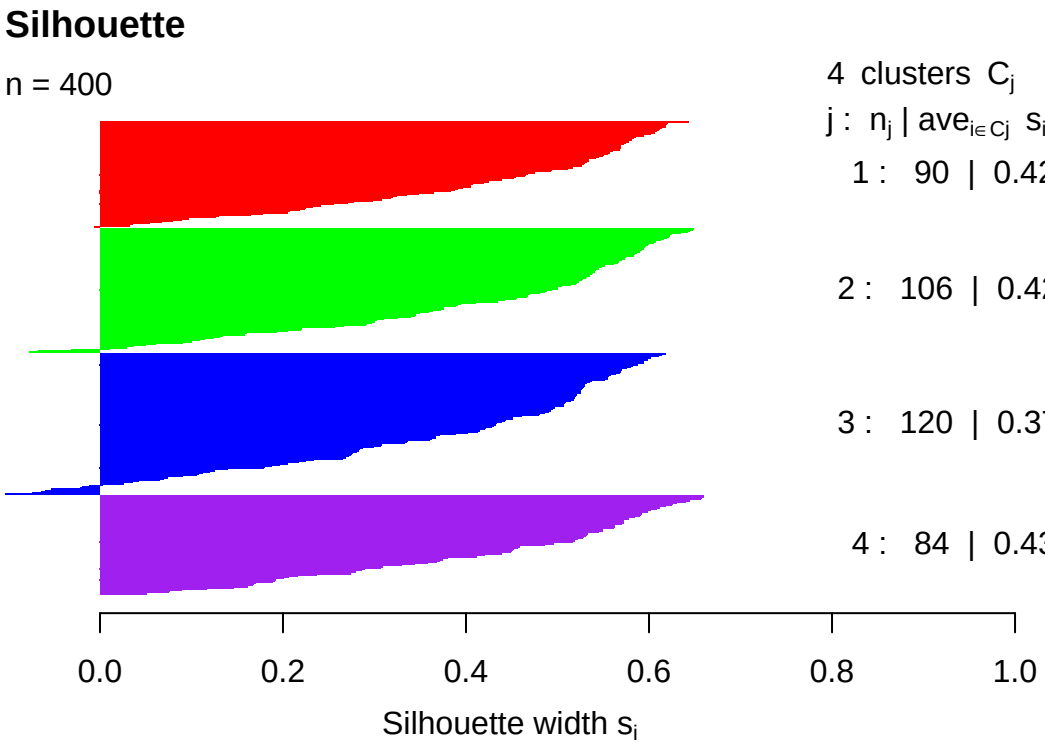
Repeat with uniform for comparison

```
for(k in 2:8){
sil = silhouette(pam(simdat2xy,k), k)
plot(sil, col=c("red", "green", "blue", "purple"), main="Silhouette")
}
```

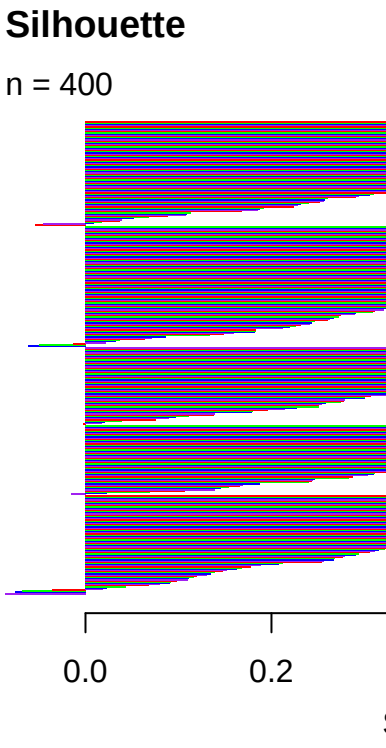


Average silhouette width : 0.36

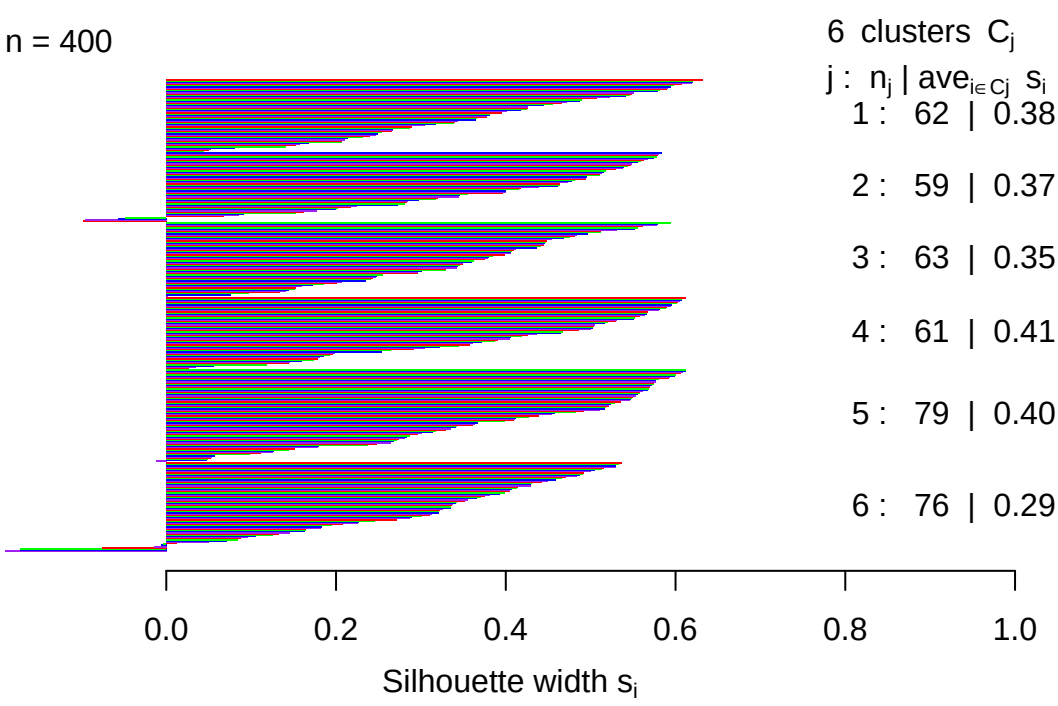
Average silhouette width :



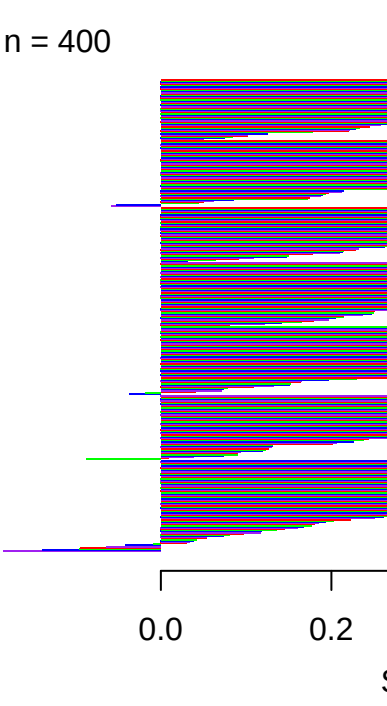
Average silhouette width : 0.41



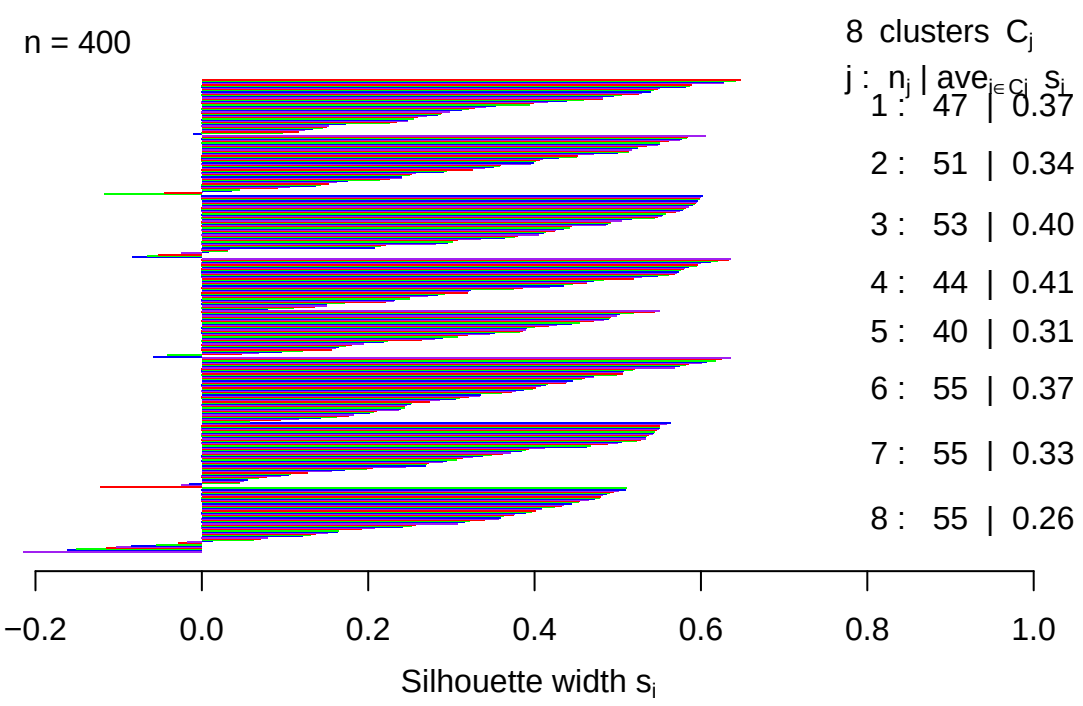
Average silhouette width :



Average silhouette width : 0.37



Average silhouette width :



5.2

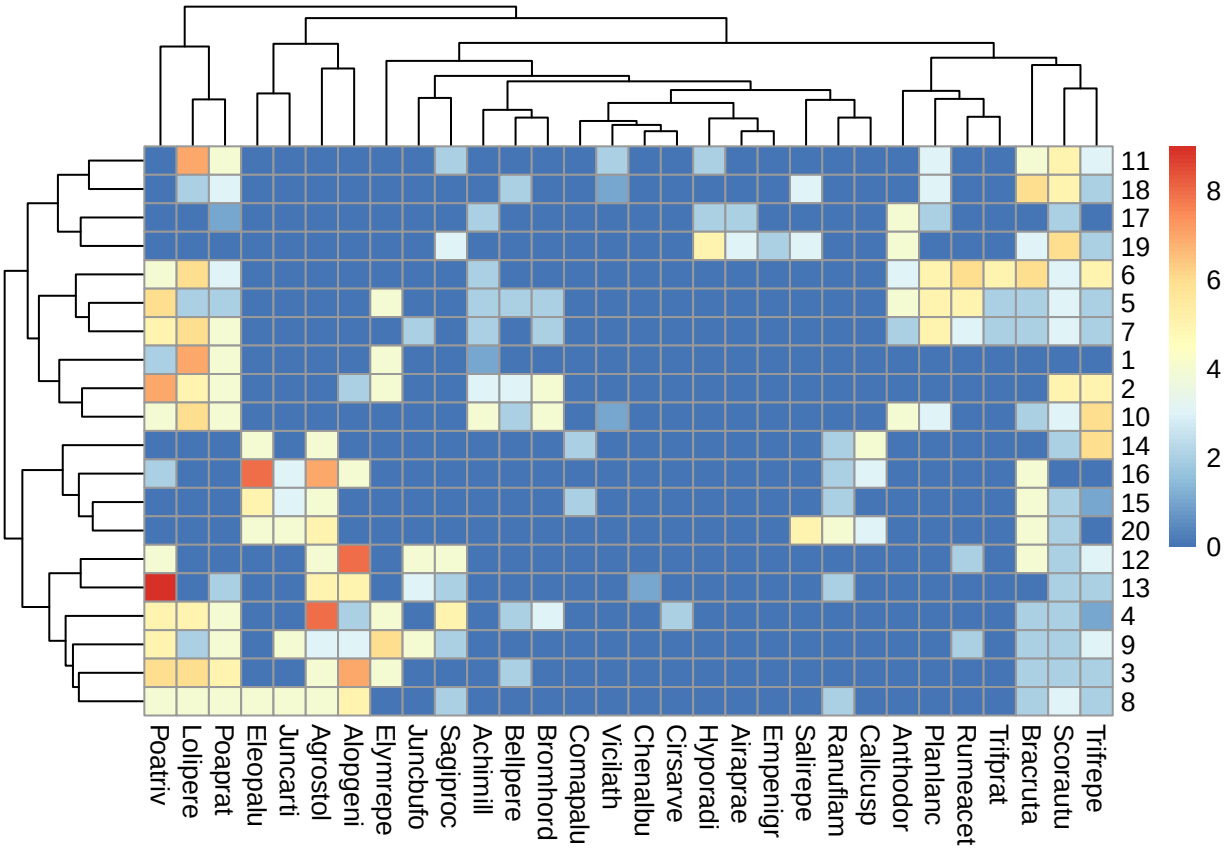
(a) & (b)

```
library(vegan)
data(dune)
symnum(cor(dune))
```

```
##          Ac Ag Ar Al An Bl Brm Ch Cr Cm Elp Ely Em H Jncr Jncb L Pl Pp Pt
## Achimill 1
## Agrostol ,  1
## Airaprae       1
## Alopgeni .  .     1
## Anthodor ,  .  .  .  1
## Bellpere .              1
## Bromhord ,              ,  1
## Chenalbu                       1
## Cirsarve    .           .  .      1
## Comapalu                              1
## Eleopalu .  .        .  .             .  1
## Elymrepe                .          .     .   1
## Empenigr       +     .                           1
## Hyporadi    .  *     .                           +  1
## Juncarti .  .        .  .  .            ,             1
## Juncbufo          .     .      .                           1
## Lolipere .  .  .        .  .          .  .   .         .         1
## Planlanc .  ,     .  .                 .              .         . 1
## Poaprat  .     .        .  .          .  .   .   .              +    1
```

```
## Poatriv        .  .     .  .   .     .  .   .      .      .         .  1
## Ranuflam .  .        .  .  .         .  ,   .        ,         . .  .
## Rumeacet .  .        .                                          ,     .
## Sagiproc .  .     .              ,            .        .      .
## Salirepe                                      .  .           .    .  .
## Scorautu    .  .       .              .       .  . .           .
## Trifprat .  .        .                                       . ,
## Trifrepe .  .              .                          .
## Vicilath    .                                                . .     .
## Bracruta                        .            .                .     .
## Callcusp    .                         .  ,           .        .    .  .
##          Rn Rm Sg Sl Sc Trfp Trfr V Brc Cl
## Achimill
## Agrostol
## Airaprae
## Alopgeni
## Anthodor
## Bellpere
## Bromhord
## Chenalbu
## Cirsarve
## Comapalu
## Eleopalu
## Elymrepe
## Empenigr
## Hyporadi
## Juncarti
## Juncbufo
## Lolipere
## Planlanc
## Poaprat
## Poatriv
## Ranuflam 1
## Rumeacet .  1
## Sagiproc       1
## Salirepe .        1
## Scorautu .        .  1
## Trifprat    +           1
## Trifrepe             .       1
## Vicilath             .            1
## Bracruta    .     .     .         . 1
## Callcusp ,           .                  1
## attr(,"legend")
## [1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
```

```
pheatmap(dune)
```



5.3

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

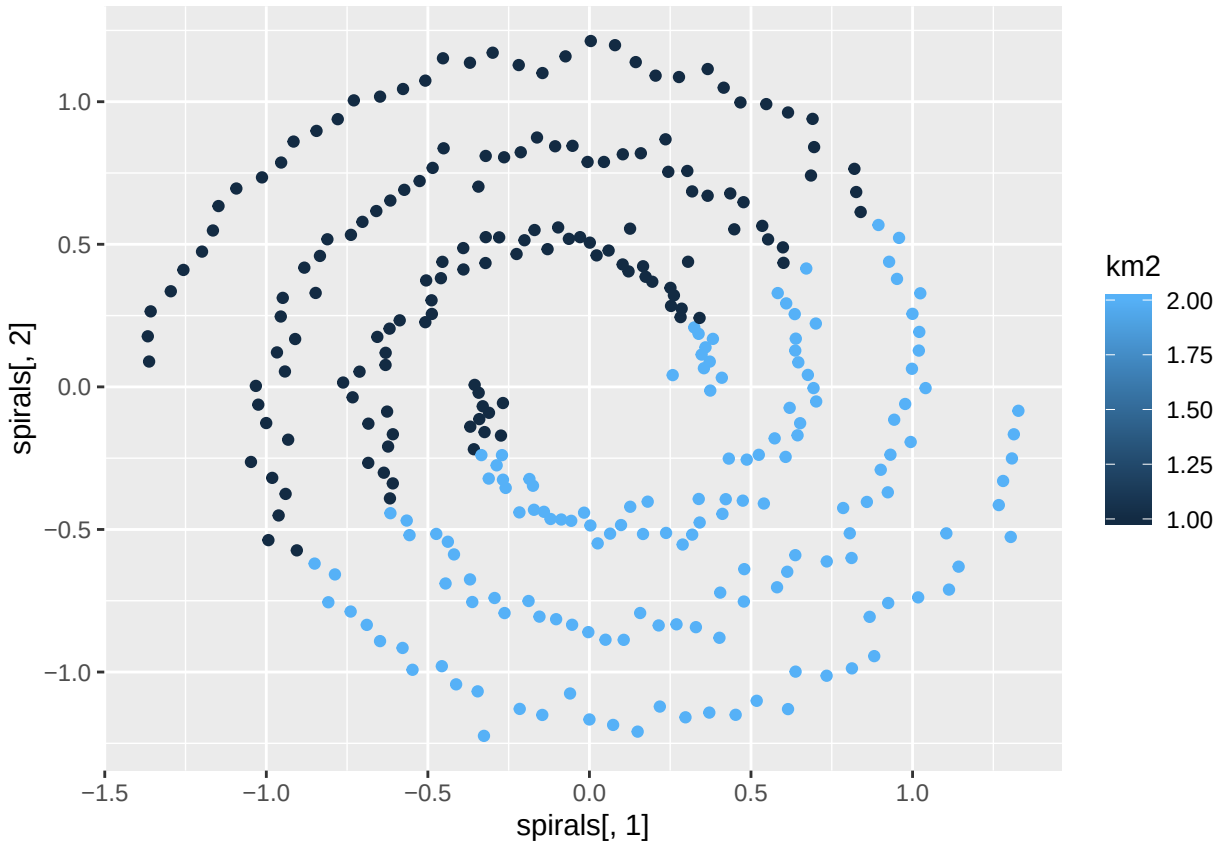
```
## The following object is masked from 'package:DelayedArray':
##
##     type
```

```
## The following object is masked from 'package:BiocGenerics':
##
##     type
```

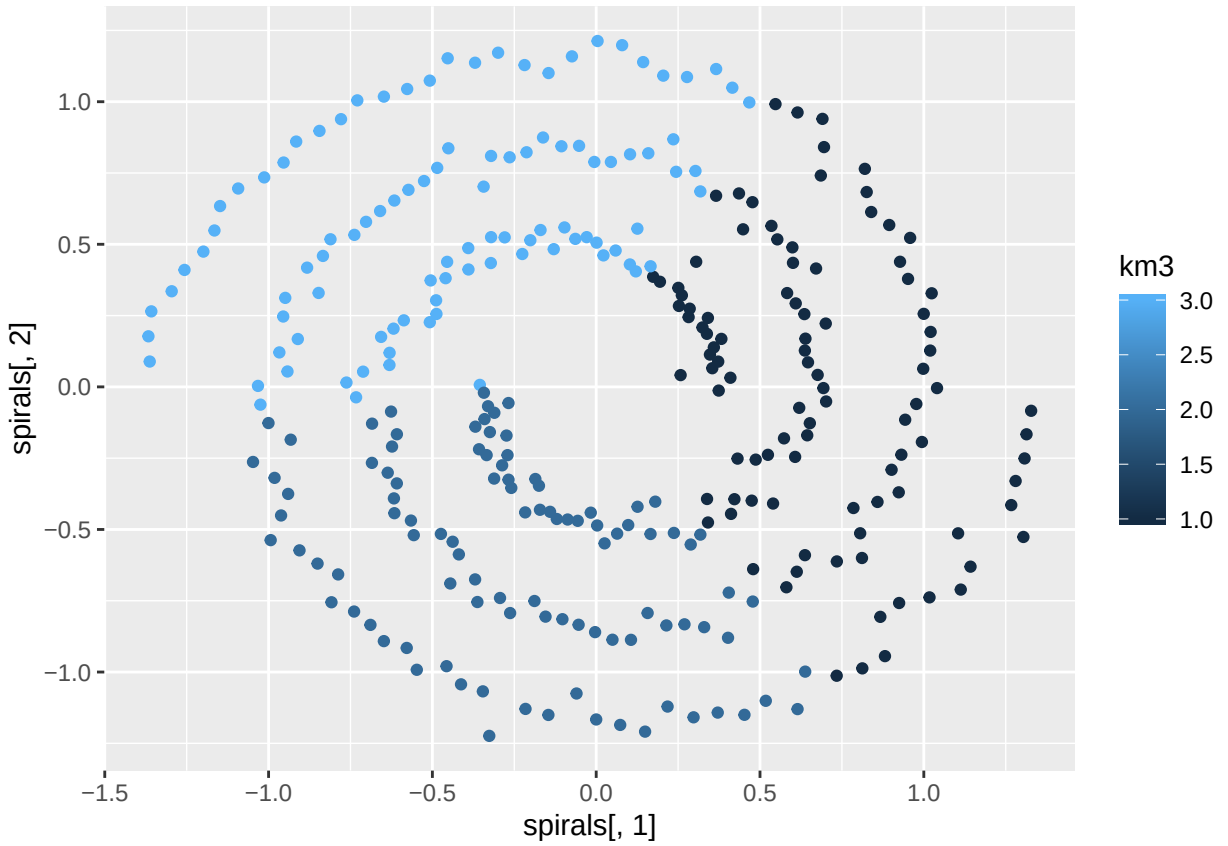
```
## The following object is masked from 'package:permute':
##
##     how
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
data(spirals)
km2 = kmeans(spirals, centers = 2)$cluster
ggplot(data = as.data.frame(spirals), aes(x=spirals[,1], y=spirals[,2], col=km2)) +
  geom_point()
```



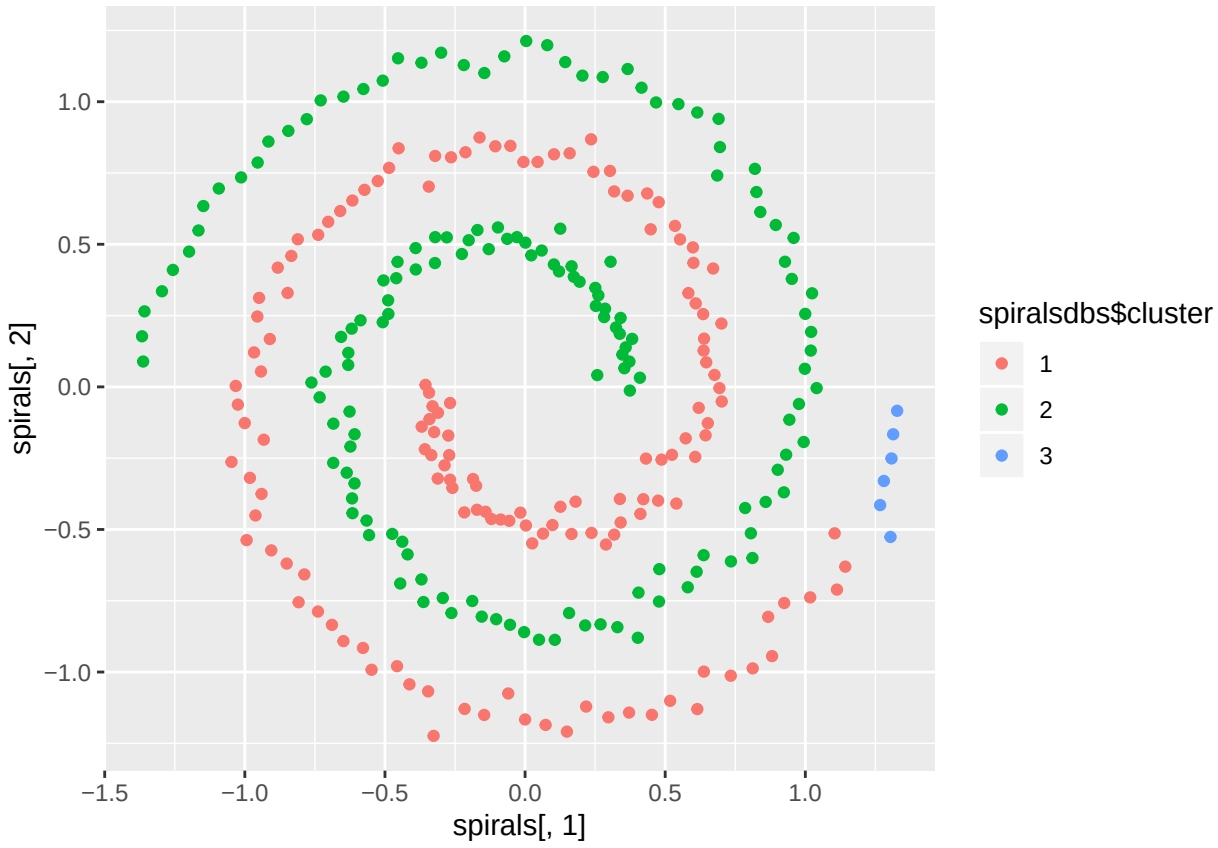

```
km3 = kmeans(spirals, centers = 3)$cluster
ggplot(data = as.data.frame(spirals), aes(x=spirals[,1], y=spirals[,2], col=km3)) +
  geom_point()
```




```
#eps 0.15 minPts=2 to recreate text example
dbs<-dbscan(spirals, eps=0.15, minPts=2)
spiralsdbs = data.frame(spirals, cluster = as.factor(dbs$cluster))
table(dbs$cluster)
```

```
##
##   1   2   3
## 144 150   6
```

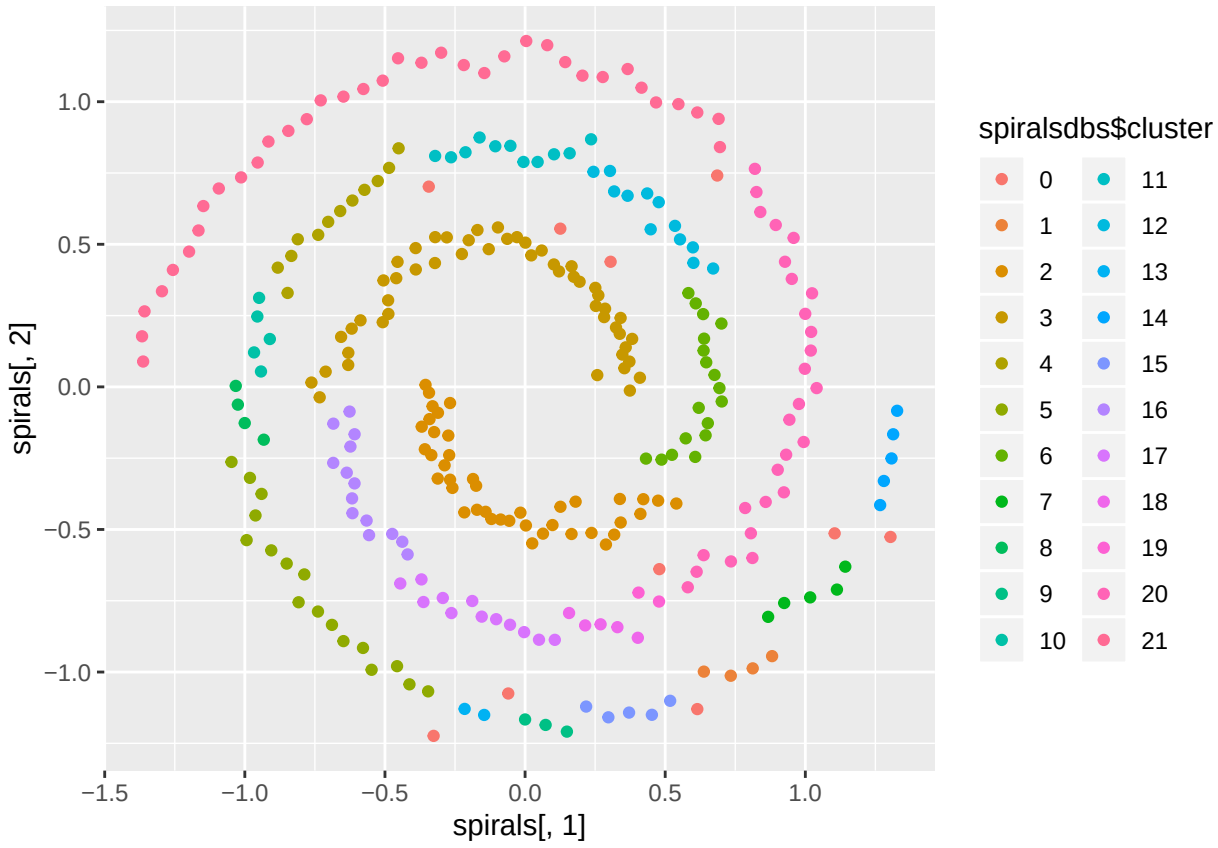
```
ggplot(as.data.frame(spirals), aes(x=spirals[,1], y=spirals[,2], col=spiralsdbs$cluster)) +
  geom_point()
```




```
#eps 0.1 minPts=2
dbs<-dbscan(spirals, eps=0.1, minPts=2)
spiralsdbs = data.frame(spirals, cluster = as.factor(dbs$cluster))
table(dbs$cluster)
```

```
##
##  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
## 10  4 41 50 12 17 18  5  4  3  5 11 12  2  5  5 14 12  5  2 27 36
```

```
ggplot(as.data.frame(spirals), aes(x=spirals[,1], y=spirals[,2], col=spiralsdbs$cluster)) +
  geom_point()
```




```
#eps 0.01 minPts=2 to recreate text example
dbs<-dbscan(spirals, eps=0.01, minPts=2)
spiralsdbs = data.frame(spirals, cluster = as.factor(dbs$cluster))
table(dbs$cluster)
```

```
##
##   0
## 300
```

```
ggplot(as.data.frame(spirals), aes(x=spirals[,1], y=spirals[,2], col=spiralsdbs$cluster)) +
  geom_point()
```

ALERT!!: Number of groups inflates quickly. There is a definite sweet spot between all different groups and all in the same group. What is this?

Q5.4

Water lines and population density?

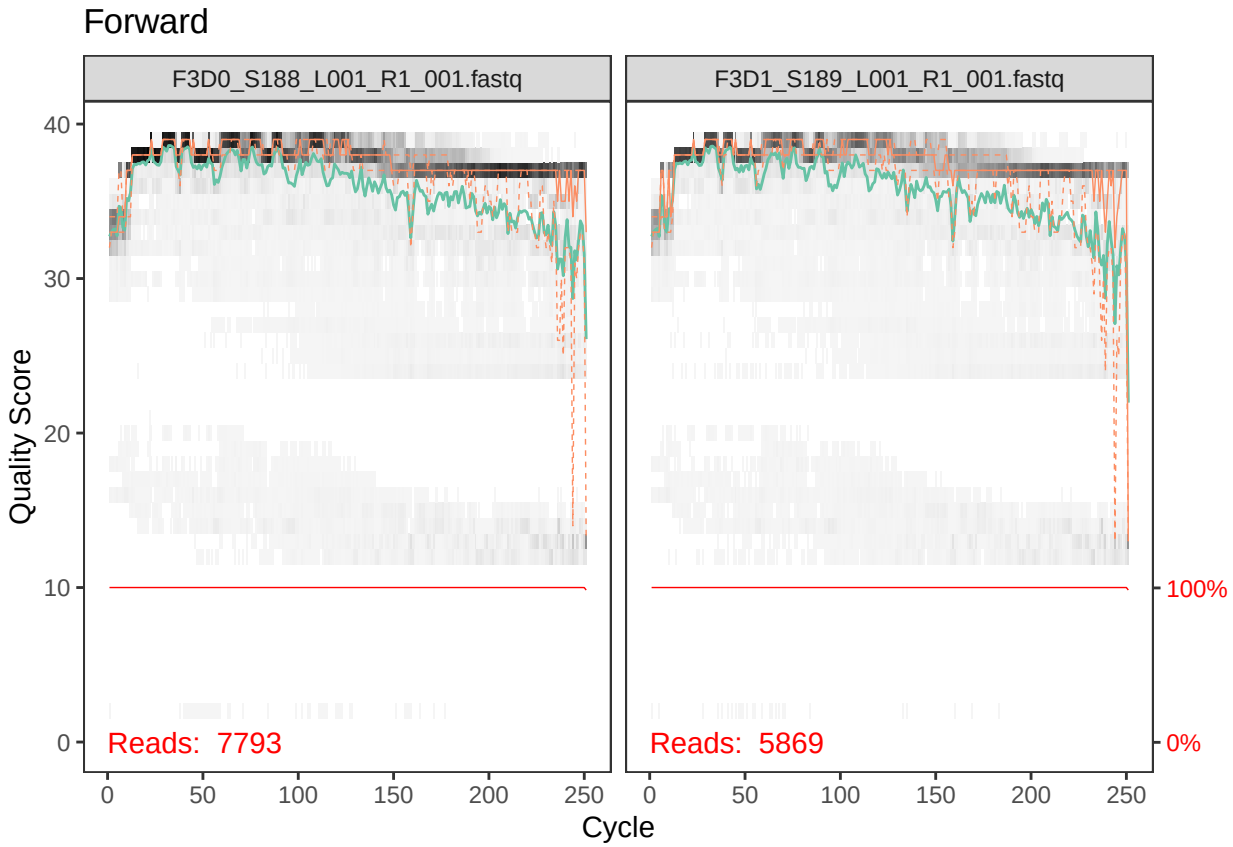
Q 5.5

```
library(dada2)
base_dir = here("data")
miseq_path = file.path(base_dir, "MiSeq_SOP")
filt_path = file.path(miseq_path, "filtered")
fnFs = sort(list.files(miseq_path, pattern="_R1_001.fastq"))
fnRs = sort(list.files(miseq_path, pattern="_R2_001.fastq"))
sampleNames = sapply(strsplit(fnFs, "_"), `[`, 1)
if (!file_test("-d", filt_path)) dir.create(filt_path)
filtFs = file.path(filt_path, paste0(sampleNames, "_F_filt.fastq.gz"))
filtRs = file.path(filt_path, paste0(sampleNames, "_R_filt.fastq.gz"))
fnFs = file.path(miseq_path, fnFs)
fnRs = file.path(miseq_path, fnRs)
print(length(fnFs))
```

```
## [1] 20
```

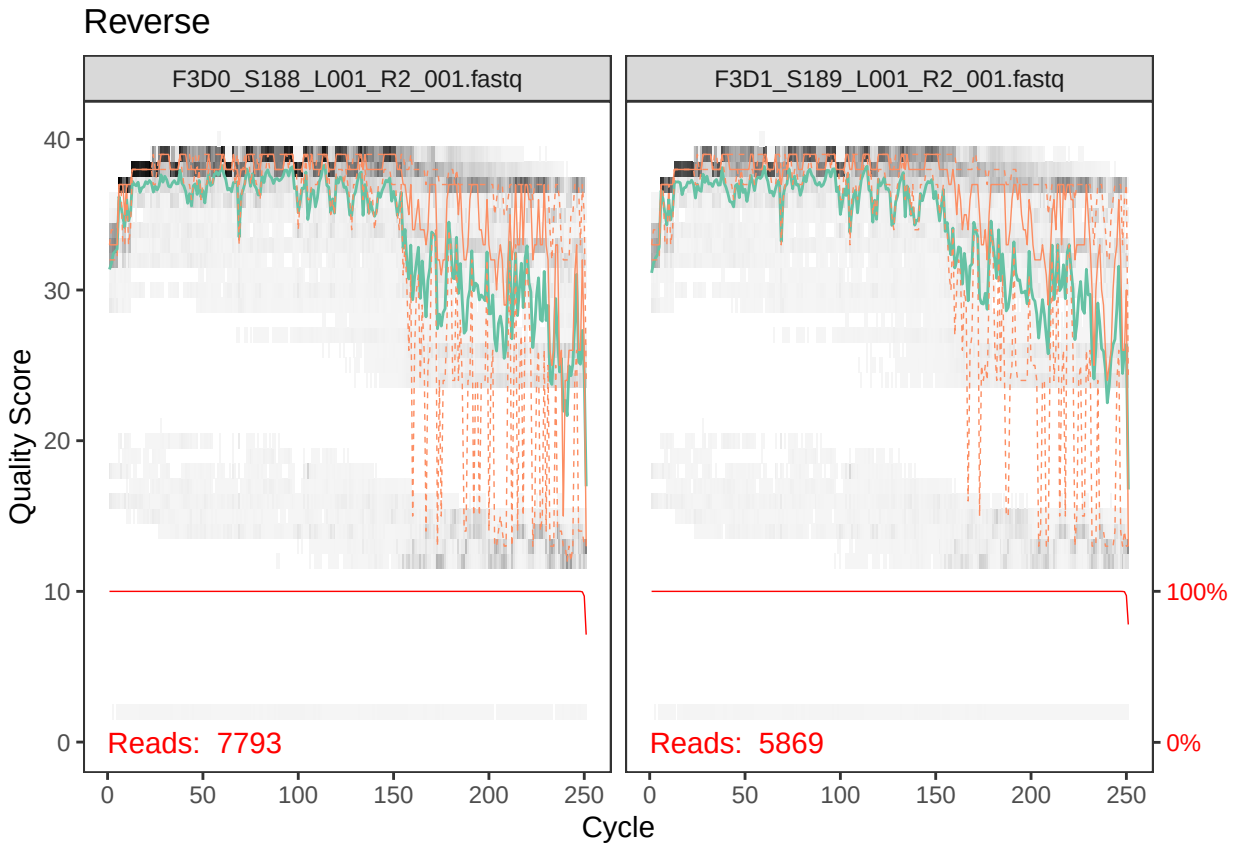
```
plotQualityProfile(fnFs[1:2]) + ggtitle("Forward")
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
```




```
plotQualityProfile(fnRs[1:2]) + ggtitle("Reverse")
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
```



Q 5.6

ALERT!!: What am I randomizing?

Q 5.7

ALERT!!: Also declaring this a que?

Q. 5.8