# Rna_seq

*saleh fayyaz*

*January 30, 2018*

fist we get the data as

```
tbl <- read.table('PBMC_scRNA-seq.txt')
tbl <- as.matrix(tbl)
tbl_2 <- tbl
dim(tbl)
```

```
## [1] 6713 3694
```

so we have 6713 gens for 3694 cell. then counts for spike-in transcripts and endogenous genes are stored in a SingleCellExperiment.

```
## [1] 6713 3694
```

removing genes with existing in zero cell .

```
keep <- rowSums(counts(sce) > 0) > 0
sce <- sce[keep, ]
```

then we need to find spike-in transcript for using it for normalization part .

```
is.spike <- grepl("^ERC",rownames(sce))
isSpike(sce,"ERC") <- is.spike
summary(is.spike)
```

```
##    Mode   FALSE    TRUE
## logical    6708       5
```

so we have 4 ERCC spike-in and 5 ERC . totally 9 spike-in was seen in dataset.

for Quality Control part we need mitochonderial Gens .

```
##    Mode   FALSE    TRUE
## logical    6686      27
```

so we have 27 mitochonderial genes.

```
sce <- calculateQCMetrics(
    sce,
    feature_controls = list(
        ERCC = isSpike(sce, "ERC"),
        Mt = isSpike(sce, "MT")
  )
)

head(colnames(colData(sce)))
```

```
## [1] "total_features"              "log10_total_features"
## [3] "total_counts"                "log10_total_counts"
## [5] "pct_counts_top_50_features"  "pct_counts_top_100_features"
```

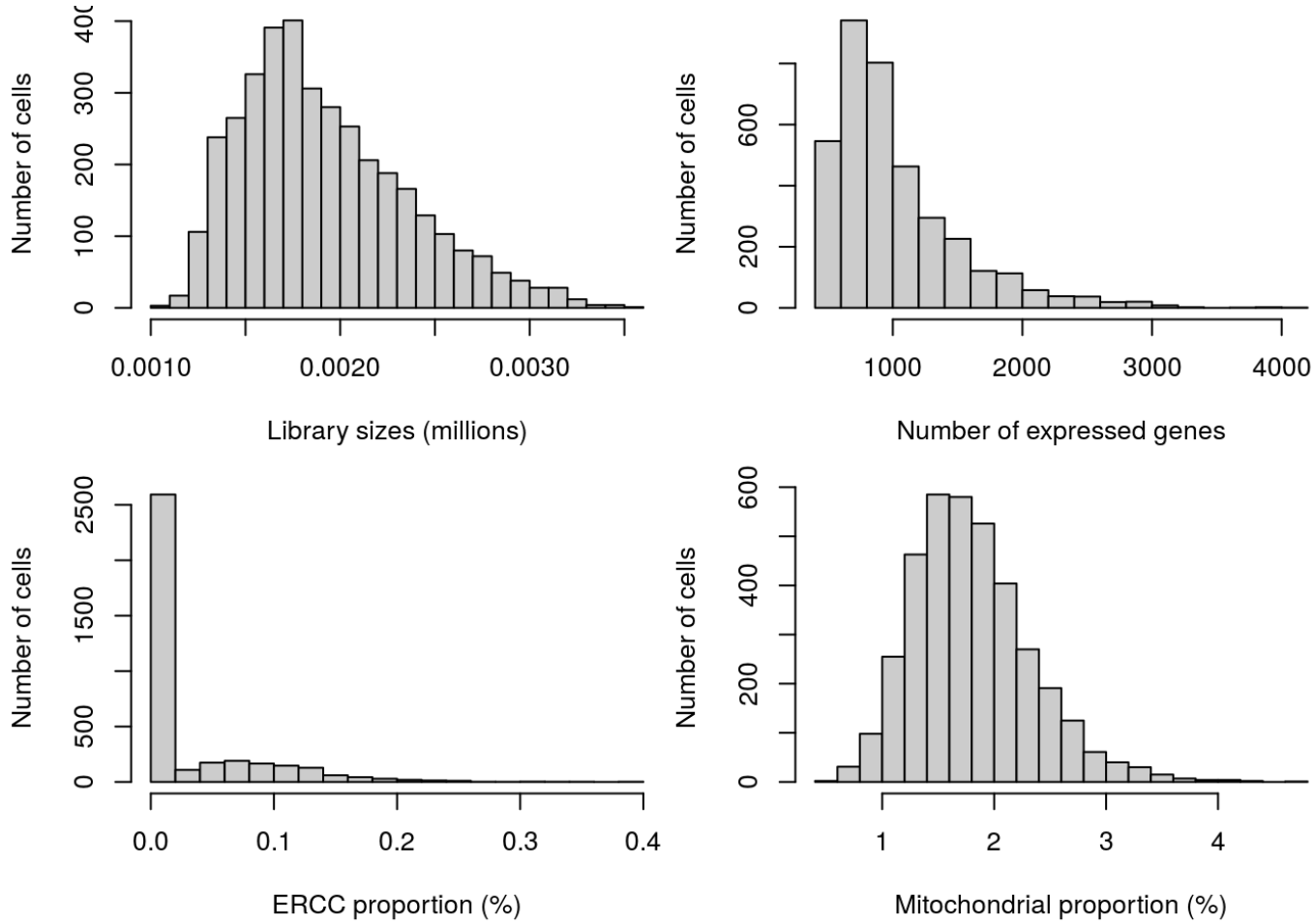
for Quality control we need creterion . #i find them in Biocunuctor site like this :

Low-quality cells need to be removed to ensure that technical effects do not distort downstream analysis results. We use several quality control (QC) metrics:

1- The library size is defined as the total sum of counts across all features, i.e., genes and spike-in transcripts. Cells with small library sizes are of low quality as the RNA has not been efficiently captured (i.e., converted into cDNA and amplified) during library preparation. 2- The number of expressed features in each cell is defined as the number of features with non-zero counts for that cell. Any cell with very few expressed genes is likely to be of poor quality as the diverse transcript population has not been successfully captured. 3- The proportion of reads mapped to spike-in transcripts is calculated relative to the library size for each cell. High proportions are indicative of poor-quality cells, where endogenous RNA has been lost during processing (e.g., due to cell lysis or RNA degradation). The same amount of spike-in RNA to each cell, so an enrichment in spike-in counts is symptomatic of loss of endogenous RNA. 4- In the absence of spike-in transcripts, the proportion of reads mapped to genes in the mitochondrial genome can also be used. High proportions are indicative of poor-quality cells (Islam et al. 2014; Ilicic et al. 2016), possibly because of loss of cytoplasmic RNA from perforated cells. The reasoning is that mitochondria are larger than individual transcript molecules and less likely to escape through tears in the cell membrane.

so for defining bad cell and remove it we first plot histogram for each part :

```
par(mfrow=c(2,2), mar=c(5.1, 4.1, 0.1, 0.1))
hist(sce$total_counts/1e6, xlab="Library sizes (millions)", main="",
    breaks=20, col="grey80", ylab="Number of cells")
hist(sce$total_features, xlab="Number of expressed genes", main="",
    breaks=20, col="grey80", ylab="Number of cells")
hist(sce$pct_counts_ERCC, xlab="ERCC proportion (%)",
    ylab="Number of cells", breaks=20, main="", col="grey80")


hist(sce$pct_counts_Mt, xlab="Mitochondrial proportion (%)",
    ylab="Number of cells", breaks=20, main="", col="grey80")
```




then based on 3 * MAD of data in each part I will remove bad cells . 1- I remove cells with log-library sizes that are more than 3 MADs below the median log-library size. A log-transformation improves resolution at small values, especially when the MAD of the raw values is comparable to or greater than the median. 2- I also remove cells where the log-transformed number of expressed genes is 3 MADs below the median value.(#BioConductor) then i will remove proportion of spike-in and mithochonderial criterion without log tranfrom and by type higher as we are identifying large outliers, for which the distinction should be fairly clear on the raw scale .

```
library(mvoutlier)
```

```
## Loading required package: sgeostat
```

```
## sROC 0.1-2 loaded
```

```
libsize.drop <- isOutlier(sce$total_counts, nmads=3, type="lower", log=TRUE)
feature.drop <- isOutlier(sce$total_features, nmads=3, type="lower", log=TRUE)

spike.drop <- isOutlier(sce$pct_counts_ERCC, nmads=3, type="higher")
mito.drop <- isOutlier(sce$pct_counts_Mt  , nmads=3, type="higher")
```

now i remove bad cells based on criterions . and then i will how many

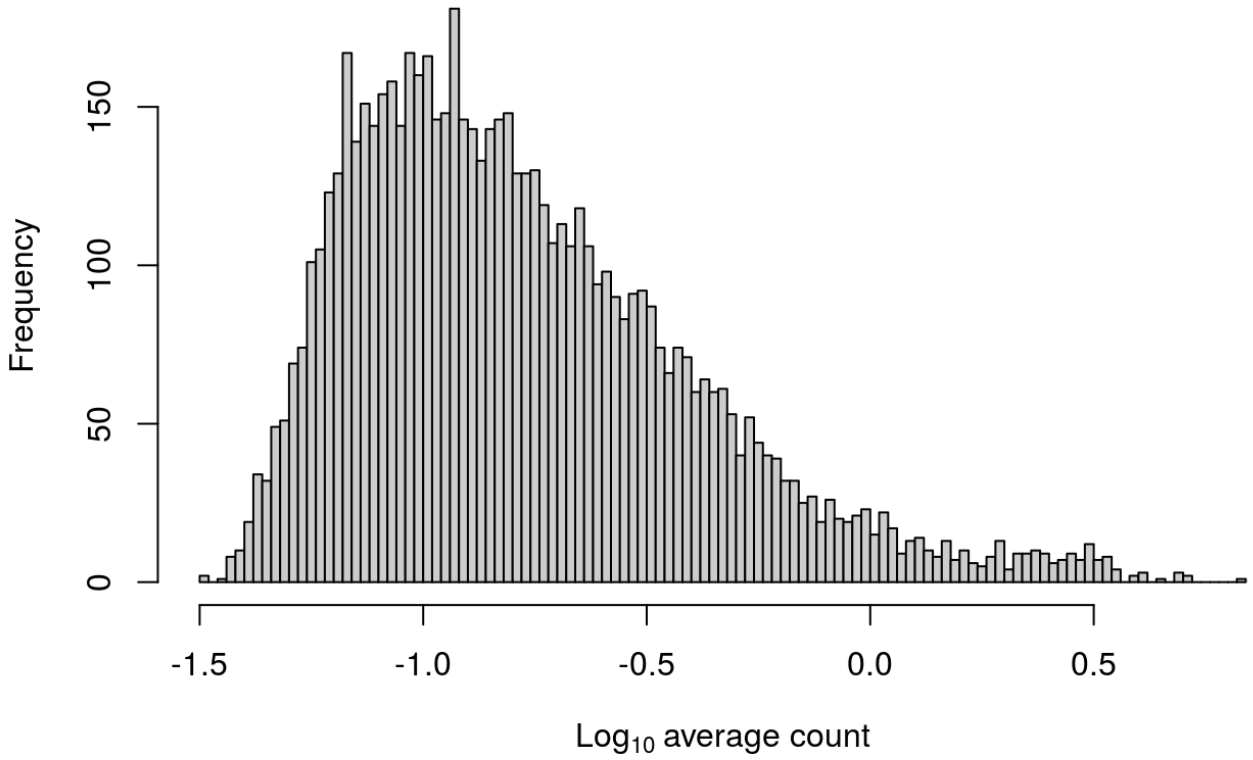
```
sce <- sce[,!(libsize.drop | feature.drop | spike.drop | mito.drop )]
data.frame(ByLibSize=sum(libsize.drop), ByFeature=sum(feature.drop),
    BySpike=sum(spike.drop),ByMito=sum(mito.drop), Remaining=ncol(sce))
```

```
##   ByLibSize ByFeature BySpike ByMito Remaining
## 1         0         0    1107     61      2540
```

# gene filtering

first i will use average count for each gene beacause as we know some cells may have low gene expression for some gene by default . then i will use second approach for rare populations .
I calculate this using the calcAverage function, which also performs some adjustment for library size differences between cells We typically observe a peak of moderately expressed genes following a plateau of lowly expressed genes .

```
ave.counts <- calcAverage(sce)
hist(log10(ave.counts), breaks=100, main="", col="grey80",
    xlab=expression(Log[10]~"average count"))
```




then i will remove genes by avg_count less than 1 .

```
keep_genes <- ave.counts >= 1
filtered.sce <- sce[keep_genes,]
summary(keep_genes)
```

```
##    Mode   FALSE    TRUE
## logical    6430     283
```

i use second appraoch for rare populations . i will use 'nxprs'from scater library that is counting the number of expressed genes per cell.

```
cell_exp_count <- nexprs(sce, byrow=TRUE)
### i will use zero threshold for removing gens that are not expressed in any cell
.
to.keep <- cell_exp_count > 0
sce <- sce[to.keep,]
summary(to.keep)
```

```
##    Mode   FALSE    TRUE
## logical       5    6708
```

# Normalization

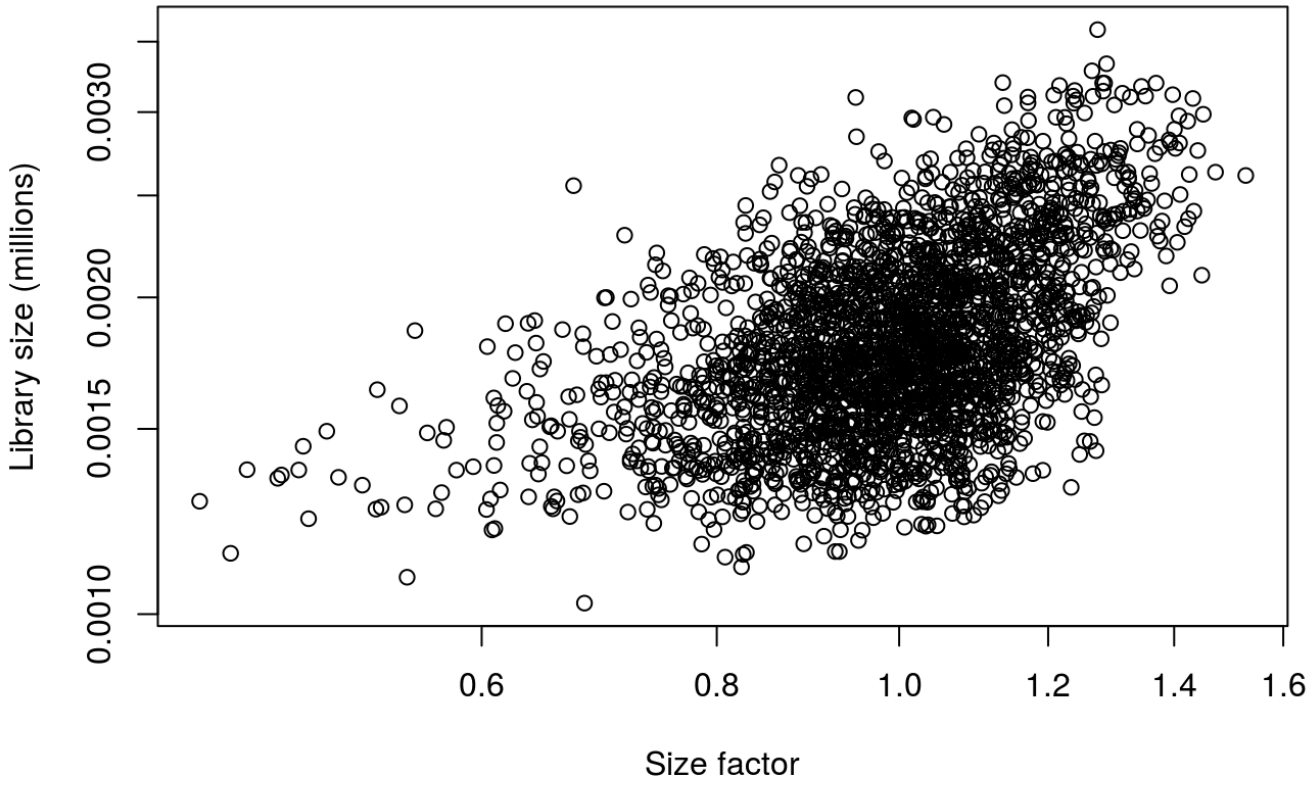
1- #deconvlotion Cell-specific biases are normalized using the computeSumFactors method, which implements the deconvolution strategy for scRNA-seq normalization. This computes size factors that are used to scale the counts in each cell.

as said there is an assumption that for most genes are not diffrentialy expressed between cells . so what we can do for difrentially expressed genes that we use them later for clustring ? in R For larger data sets, clustering should be performed with the quickCluster function before normalization. Briefly, cells are grouped into clusters of similar expression; normalization is applied within each cluster to compute size factors for each cell; and the factors are rescaled by normalization between clusters. This reduces the risk of violating the above assumption when many genes are DE between clusters in a heterogeneous population.

```
sce2 <- computeSumFactors(sce)
summary(sizeFactors(sce2))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4249  0.9029  1.0009  1.0000  1.0996  1.5283
```

```
plot(sizeFactors(sce2), sce2$total_counts/1e6, log="xy",
    ylab="Library size (millions)", xlab="Size factor")
```




if our hypothesis about a lot of none-DE genes was good we have to see a high correlation between libery size and size factor (linear) . as any DE between cells would yield a non-linear trend between the total count and size factor, and/or increased scatter around the trend. as you see in plot this is not very good assumption . so we use quickcluster . before that i will use data in spike-in RNA . as we know the same amount of spike-in was added to each cell prior to library prepration . (biocundoctor)it is strongly recommended to compute a separate set of size factors for the spike-ins. This is because the spike-ins are not affected by total mRNA content. Using the deconvolution size factors will over-normalize the spike-in counts, whereas the spike-in size factors are more appropriate. for not overwriting the former i will set not genral use in below function . This means that the spike-in-based size factors will be computed and stored in the SingleCellExperiment object, but will only be used by the spike-in transcripts.

```
clusters <- quickCluster(sce, method = "igraph")
sce <- computeSumFactors(sce, cluster=clusters)
sce <- computeSpikeFactors(sce, general.use=FALSE)
#Applying the size factors to normalize gene expression
sce <- normalize(sce)
```

# another approach for normalization can be :

```
library(DESeq2)
```

```
##
## Attaching package: 'DESeq2'
```

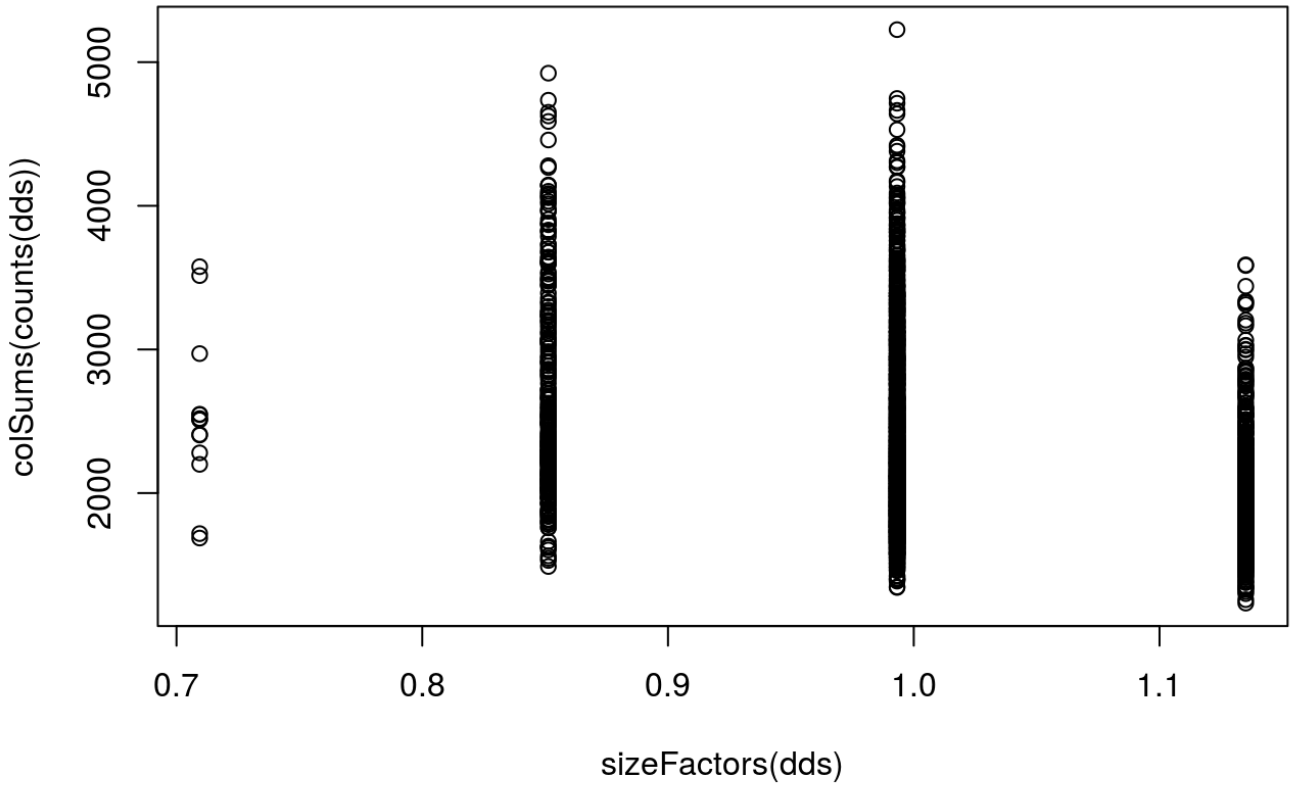
```
## The following object is masked from 'package:scater':
##
##     fpkm
```

```
library(GenomicRanges)
se <- SummarizedExperiment(ceiling(counts(sce)))
colData(se) <- DataFrame(colData(sce))
dds <- DESeqDataSet( se, design = ~ 1 )
```

```
## renaming the first element in assays to 'counts'
```

```
## converting counts to integer mode
```

```
#Estimate size factors
dds <- estimateSizeFactors( dds )

#Plot column sums according to size factor
plot(sizeFactors(dds), colSums(counts(dds)))
```




```
#The argument normalized equals true, divides each column by its size factor.
logcounts <- log2( counts(dds, normalized=TRUE) + 1 )
```

# why I used decovloution method for this data ?

Briefly this method deals with the problem of vary large numbers of zero values per cell by pooling cells together calculating a normalization factor (similar to CPM) for the sum of each pool. Since each cell is found in many different pools, cell-specific factors can be deconvoluted from the collection of pool-specific factors using linear algebra. as you can see in the data we have a lorge number of zeros in dataset so this method is realiable for this data . #

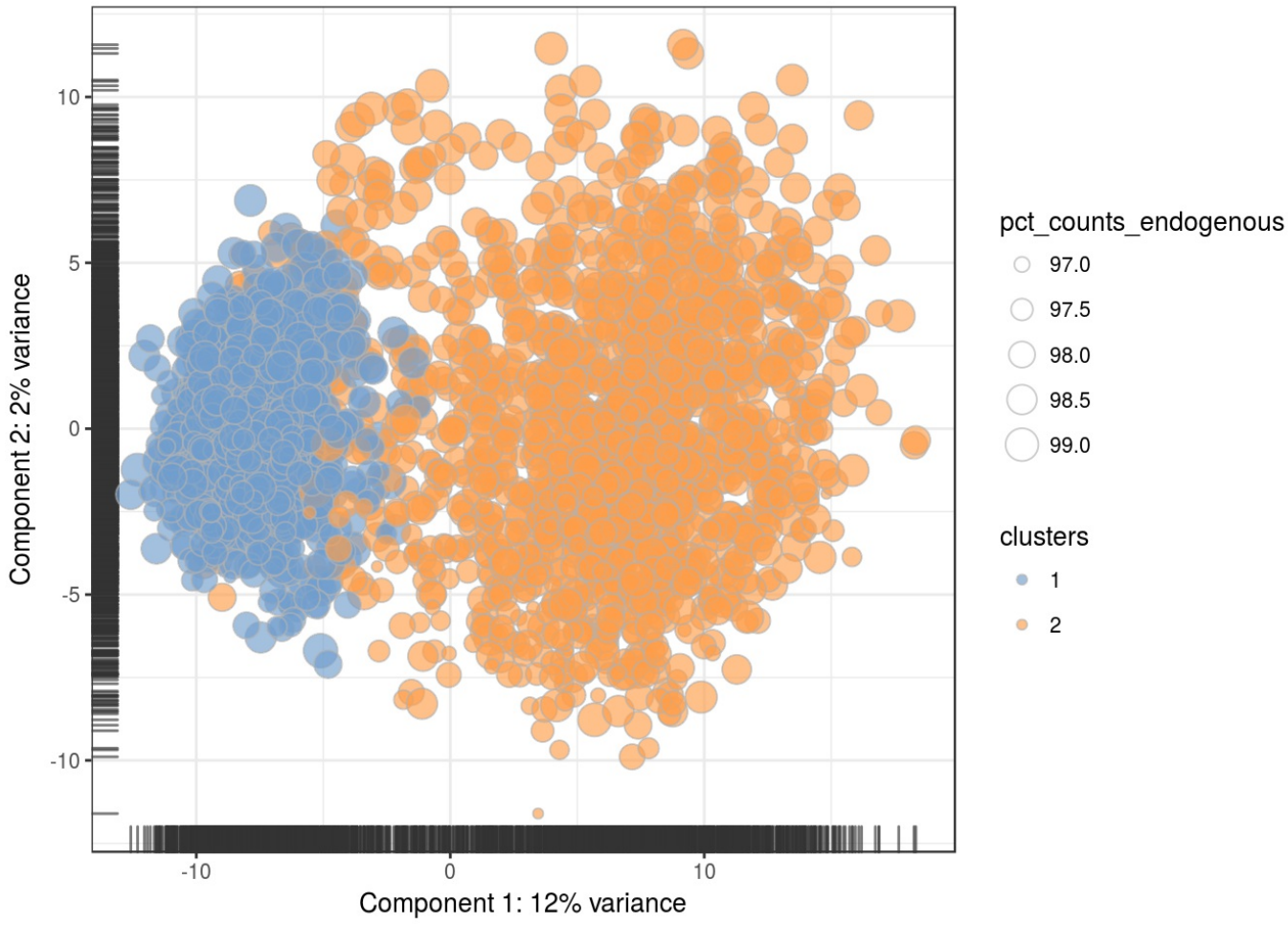
```
# create a list to simplify the plotting step


# visualise the data as a set of boxplots
#pdf(file="./normalising_ex2.pdf", width=10)
#boxplot(counts(sce))
#dev.off()
```

# clustring

# PCA for Dimention Reduction

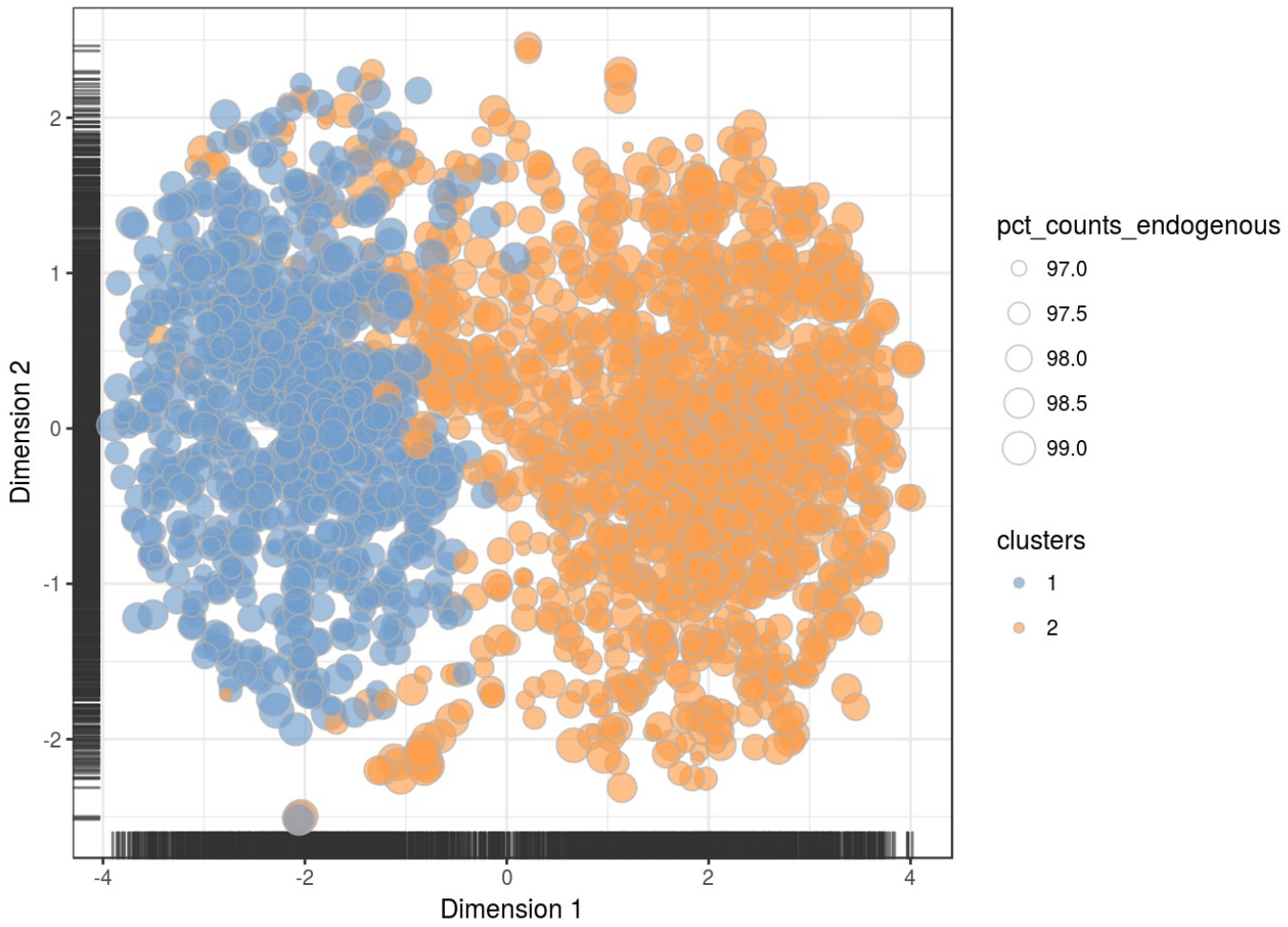
for better visulaising i use quick clustring in the latter part to color each cell . for size too i will use “pct_counts_endogenous” as argument.

```
sce$clusters = clusters
plotPCA(sce, colour_by = 'clusters', size_by = "pct_counts_endogenous")
```




as you can see a good clustring is happening in PCA aproach . #t-SNE

```
plotTSNE(sce, colour_by = 'clusters', size_by = "pct_counts_endogenous")
```




its hard ro justify which one one is better on accuracy but intrestingly PCA get much less time than T-SNE that makes it more intresting . ##Validation Measures(clValid package) 1 - #Internal validation: measures take only the dataset and the clustering partition as input and use intrinsic information in the data to assess the quality of the clustering. 2 - #The stability measures : are a special version of internal measures. They evaluate the consistency of a clustering result by comparing it with the clusters obtained after each column is removed, one at a time 3- # Biological validation : evaluates the ability of a clustering algorithm to produce biologically meaningful clusters

1-A) we selected measures that reflect the #compactness, connectedness, and #separation of the cluster partitions. Connectedness relates to what extent observations are placed in the same cluster as their nearest neighbors in the data space, and is here measured by the connectivity . Compactness assesses cluster homogeneity, usually by looking at the intra-cluster variance, while separation quantifies the degree of separation between clusters (usually by measuring the distance between cluster centroids)

The Dunn Index (Dunn, 1974) and Silhouette Width (Rousseeuw, 1987) are both examples of non-linear combinations of the compactness and separation.

# Silhouette Width

The Silhouette Width is the average of each observation's Silhouette value. The Silhouette value measures the degree of confidence in the clustering assignment of a particular observation, with well-clustered observations having values near 1 and poorly clustered observations having values near −1. For observation i, it is defined as:

$$S(i) = \frac{b_i - a_i}{max(b_i, a_i)},$$

where $a_i$ is the average distance between $i$ and all other observations in the same cluster, and $b_i$ is the average distance between $i$ and the observations in the "nearest neighboring cluster", i.e.

$$b_i = \min_{C_k \in \mathcal{C} \setminus C(i)} \sum_{j \in C_k} \frac{dist(i,j)}{n(C_k)},$$

where C(i) is the cluster containing observation i, dist(i, j) is the distance (e.g. Euclidean, Manhattan) between observations i and j, and n(C) is the cardinality of cluster C. The Silhouette Width thus lies in the interval [−1, 1], and should be maximized. For more information, see the help page for the silhouette() function in package cluster (Rousseeuw et al., 2006).

# Dunn Index

The Dunn Index is the ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance. It is computed

$$D(\mathcal{C}) = \frac{\min_{C_k, C_l \in \mathcal{C}, C_k \neq C_l} \left( \min_{i \in C_k, j \in C_l} dist(i,j) \right)}{\max_{C_m \in \mathcal{C}} diam(C_m)},$$

where diam(Cm) is the maximum distance between observations in cluster Cm. The Dunn Index has a value between zero and ∞, and should be maximized.

# Stability measures

The stability measures compare the results from clustering based on the full data to clustering based on removing each column, one at a time. These measures work especially well if the data are highly correlated, which is often the case in high-throughput genomic data. The included measures are the average proportion of non-overlap (APN), the average distance (AD), the average distance between means (ADM), and the figure of merit (FOM) (Datta and Datta, 2003; Yeung et al., 2001). In all cases the average is taken over all the deleted columns, and all measures should be minimized.

# Biological

1- Biological Homogeneity Index (BHI)

As its name implies, the BHI measures how homogeneous the clusters are biologically. Let $\mathcal{B} = \{B_1, \ldots, B_F\}$ be a set of $F$ functional classes, not necessarily disjoint, and let $B(i)$ be the functional class containing gene $i$ (with possibly more than one functional class containing $i$). Similarly, we define $B(j)$ as the function class containing gene $j$, and assign the indicator function $I(B(i) = B(j))$ the value 1 if $B(i)$ and $B(j)$ match (any one match is sufficient in the case of membership to multiple functional classes), and 0 otherwise. Intuitively, we hope that genes placed in the same statistical cluster also belong to the same functional classes. Then, for a given statistical clustering partition $\mathcal{C} = \{C_1, \ldots, C_K\}$ and set of biological classes $\mathcal{B}$,

the BHI is defined as

$$BHI(\mathcal{C}, \mathcal{B}) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{n_k(n_k - 1)} \sum_{i \neq j \in C_k} I\left(B(i) = B(j)\right) .$$

Here $n_k = n(C_k \cap B)$ is the number of annotated genes in statistical cluster $C_k$. Note that if $n_k = 1$ or 0, i.e. there is only one or no annotated genes in statistical cluster $C_k$, then a value of zero is added for that cluster. The BHI is in the range $[0, 1]$, with larger values corresponding to more biologically homogeneous clusters.

The BSI is similar to the other stability measures, and inspects the consistency of clustering for genes with similar biological functionality. Each sample is removed one at a time, and the cluster membership for genes with similar functional annotation is compared with the cluster membership using all available samples. The BSI is defined as

$$BSI(\mathcal{C}, \mathcal{B}) = \frac{1}{F} \sum_{k=1}^{F} \frac{1}{n(B_k)(n(B_k) - 1)M} \sum_{\ell=1}^{M} \sum_{i \neq j \in B_k} \frac{n(C^{i,0} \cap C^{j,\ell})}{n(C^{i,0})} ,$$

where $F$ is the total number of functional classes, $C^{i,0}$ is the statistical cluster containing observation $i$ based on all the data, and $C^{j,\ell}$ is the statistical cluster containing observation $j$ when column $\ell$ is removed. Note that if $n(B_k) = 0$ or 1, i.e. there are less than two genes belonging to functional class $B_k$, then a value of zero is taken for that class. The BSI is in the range $[0, 1]$, with larger values corresponding to more stable clusters of the functionally annotated genes.

bsi

# Clustring methods :

in here i will find best k by internal measure in “clValid” package that discussed before. before that i will find PCA for dimention reduction and work better for clustring algorithms.

```
sce.pca <- prcomp(t(counts(sce)))
sce.pca$sdev[1:100]
```

```
##   [1] 12.553700  6.315560  4.922936  4.594124  4.064708  3.954331  3.477288
##   [8]  3.140602  2.882191  2.809438  2.615697  2.428879  2.404814  2.330443
##  [15]  2.235442  2.209674  2.106285  2.069190  2.046430  2.034864  1.969372
##  [22]  1.954479  1.943955  1.915451  1.885360  1.877593  1.858263  1.841860
##  [29]  1.836018  1.828852  1.810826  1.806248  1.798184  1.779003  1.776676
##  [36]  1.763920  1.762136  1.752338  1.743901  1.741819  1.740947  1.738189
##  [43]  1.728098  1.724382  1.721278  1.718921  1.713400  1.708525  1.706513
##  [50]  1.705101  1.700974  1.697445  1.692385  1.689857  1.684019  1.682152
##  [57]  1.681070  1.679258  1.670830  1.668725  1.664972  1.662229  1.659115
##  [64]  1.657795  1.657321  1.651876  1.650348  1.647397  1.644624  1.643480
##  [71]  1.641746  1.638767  1.637174  1.631308  1.630315  1.628941  1.625970
##  [78]  1.624932  1.622046  1.621277  1.619215  1.617028  1.614158  1.610864
##  [85]  1.610578  1.607793  1.605832  1.603381  1.601041  1.599422  1.597936
##  [92]  1.596808  1.594820  1.592951  1.589802  1.589204  1.585849  1.585189
##  [99]  1.583887  1.582302
```

```
### if you see sce.pca$sdev you will find out a lot of varience in whole data is in
first 5 Principle component.
n_pc<-5
ex_pc <- sce.pca$x[,1:n_pc]
```

# Best K:

then for finding best k for clustring i will use ‘clvalid’ package that by using cluster validation measure in top defines which k is better . in here i will give clvalid 3 diffrent approach in clustering and for internal criterion to find best k .

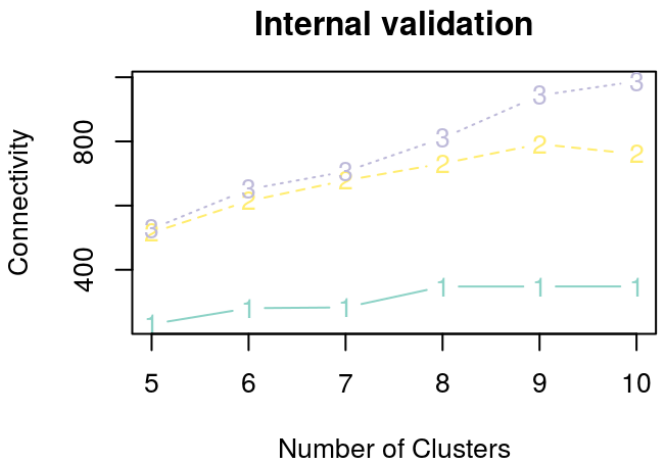
```
library(clValid)
```

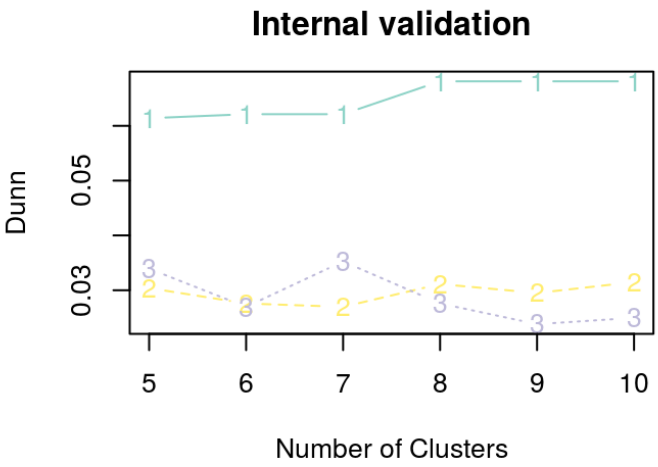
```
## Loading required package: cluster
```

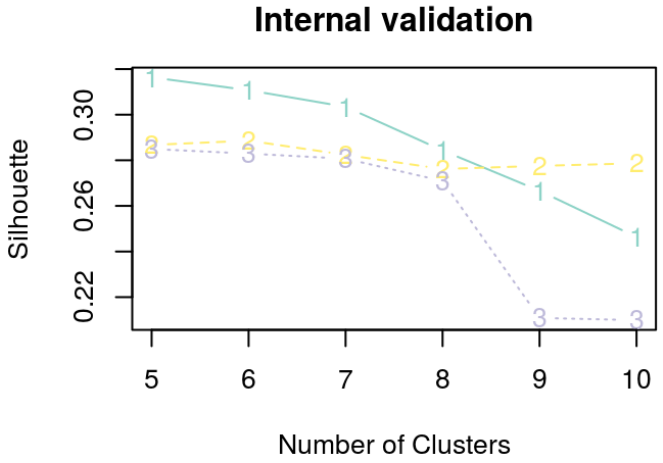
```
intern <- clValid(ex_pc, 5:10, clMethods=c("hierarchical","kmeans","pam"), validati
on="internal",maxitems=7000)
summary(intern)
```

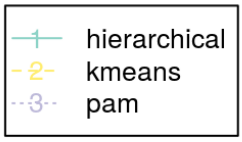
```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  5 6 7 8 9 10
##
## Validation Measures:
##                                  5        6        7        8        9
10
##

## hierarchical Connectivity  230.6508 280.4143 282.7468 347.8000 348.0111 348.2790
##              Dunn            0.0614   0.0621   0.0621   0.0681   0.0681   0.068
1
##              Silhouette      0.3164   0.3107   0.3035   0.2843   0.2666   0.246
3
## kmeans       Connectivity  515.3770 614.4147 678.5492 731.0687 791.7794 760.3250
##              Dunn            0.0304   0.0276   0.0270   0.0311   0.0295   0.031
5
##              Silhouette      0.2867   0.2886   0.2824   0.2762   0.2776   0.278
7
## pam          Connectivity  527.5425 650.8202 706.2794 810.1667 943.3456 987.3147
##              Dunn            0.0339   0.0270   0.0353   0.0276   0.0238   0.025
0
##              Silhouette      0.2850   0.2829   0.2807   0.2709   0.2109   0.209
9
##
## Optimal Scores:
##
##              Score    Method       Clusters
## Connectivity 230.6508 hierarchical 5
## Dunn           0.0681 hierarchical 8
## Silhouette     0.3164 hierarchical 5
```

```r
## as you can see results for k=5 and k==8 are optimal
# here i will ploting them too .
op <- par(no.readonly=TRUE)
par(mfrow=c(2,2),mar=c(4,4,3,1))
plot(intern, legend=FALSE)
plot(nClusters(intern),measures(intern,"Dunn")[,,1],type="n",axes=F, xlab="",ylab="
")
legend("center", clusterMethods(intern), col=1:9, lty=1:9, pch=paste(1:9))
```




```
par(op)
```

# Clustring

at last i will plot each clustring then i will use kmeans as my clustring algorithm . you can findout here that i had used 5 PC for clustring but for visulization only 2 of them is need.

```
ex_2pc <- as.data.frame(sce.pca$x[,1:2])
ex_2pc$class5 <-  (kmeans(ex_pc[,1:2] , centers= 5, iter.max = 10000 , nstart =  1
00))$cluster
ex_2pc$class6 <-  (kmeans(ex_pc[,1:2] , centers= 6, iter.max = 10000 , nstart =  1
00))$cluster

ex_2pc$class7 <-  (kmeans(ex_pc[,1:2] , centers= 7, iter.max = 10000 , nstart =  1
00))$cluster

ex_2pc$class8 <-  (kmeans(ex_pc[,1:2] , centers= 8, iter.max = 10000 , nstart =  1
00))$cluster

ex_2pc$class9 <-  (kmeans(ex_pc[,1:2] , centers= 9, iter.max = 10000 , nstart =  1
00))$cluster

ex_2pc$class10 <-  (kmeans(ex_pc[,1:2] , centers= 10,iter.max = 10000 , nstart =
100))$cluster
```

# Plot :

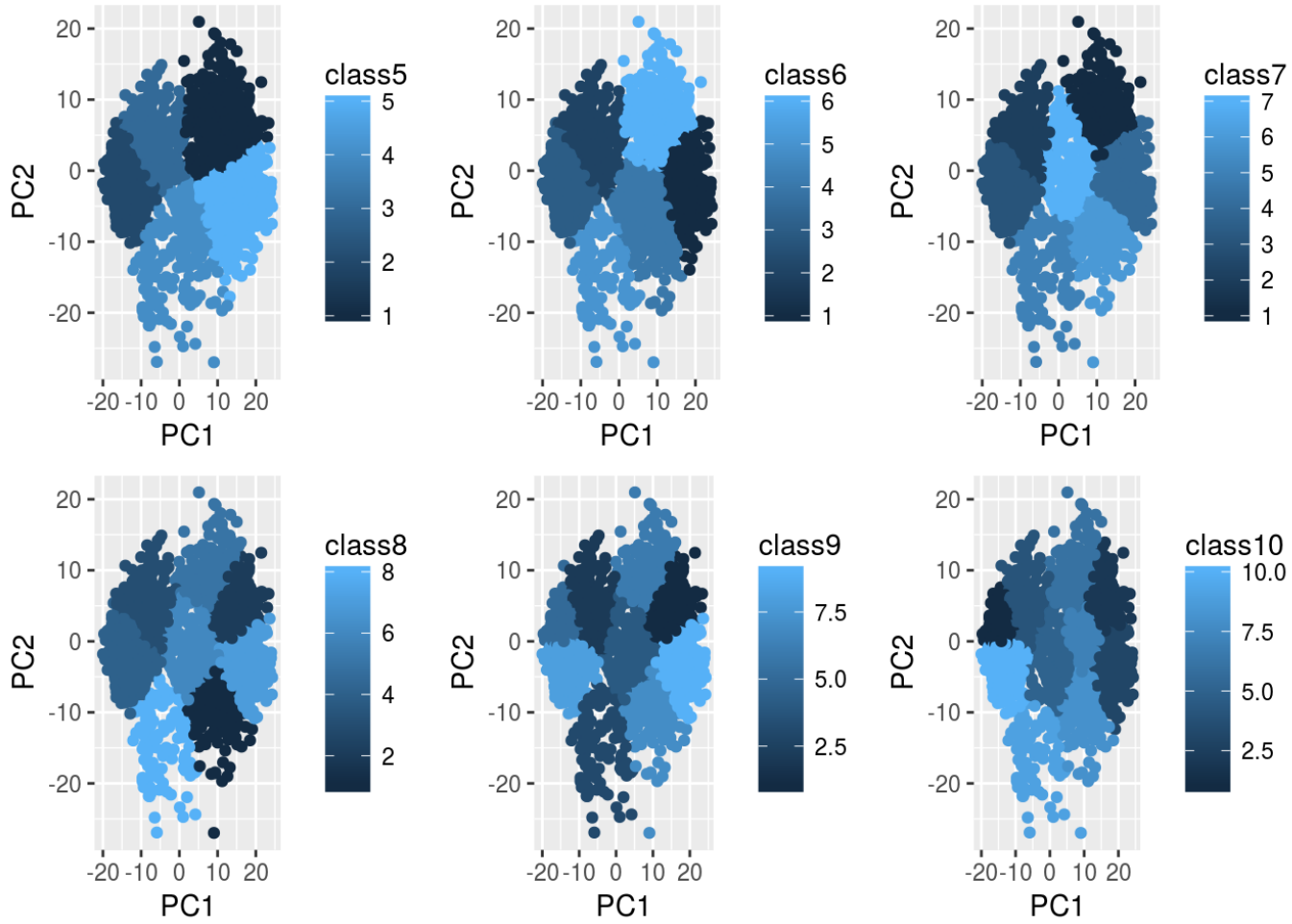
```
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:Biobase':
##
##     combine
```

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

```
### for visualization part i will choose only 2 PC
par(mfrow=c(2,3))

plot_r_1 <- ggplot(ex_2pc, aes(ex_2pc$PC1, ex_2pc$PC2 , color = class5)) + geom_poi
nt() +  xlab("PC1") + ylab("PC2")

plot_r_2 <- ggplot(ex_2pc, aes(ex_2pc$PC1, ex_2pc$PC2 , color = class6)) + geom_poi
nt() +  xlab("PC1") + ylab("PC2")

plot_r_3 <- ggplot(ex_2pc, aes(ex_2pc$PC1, ex_2pc$PC2 , color = class7)) + geom_poi
nt() +  xlab("PC1") + ylab("PC2")

plot_r_4 <- ggplot(ex_2pc, aes(ex_2pc$PC1, ex_2pc$PC2 , color = class8)) + geom_poi
nt() +  xlab("PC1") + ylab("PC2")

plot_r_5 <- ggplot(ex_2pc, aes(ex_2pc$PC1, ex_2pc$PC2 , color = class9)) + geom_poi
nt() +  xlab("PC1") + ylab("PC2")

plot_r_6 <- ggplot(ex_2pc, aes(ex_2pc$PC1, ex_2pc$PC2 , color = class10)) + geom_po
int() +  xlab("PC1") + ylab("PC2")

grid.arrange( plot_r_1, plot_r_2,plot_r_3 , plot_r_4 , plot_r_5 , plot_r_6 , ncol =
3)
```




as you can see none of clusters remains fixed completly . but you can say that almost in every picture you can find 6 cluster in class8 plot that remains constant in class 9 and 10 too .

# Naming clusters based on cells on clusters

for naming each cluster we need to find important genes in each cluster that by them we can name the cluster . we name these genes markers . but how to define importance of genes . i say a gene is diffrentially expresed in each cluster . for finding these we need to have some hypothesis testing prosecure . Find candidate marker genes for clusters of cells, by testing for differential expression between clusters. as i searched The first step in our marker gene identification process is to identify previously reported cell type markers . Fortunately, the literature is rich in papers measuring gene expression in isolated immune cell populations.