Workshop 3

Correlation, dimension reduction, and clustering analysis using R

Lulu Shang

Department of Biostatistics
MD Anderson Cancer Center

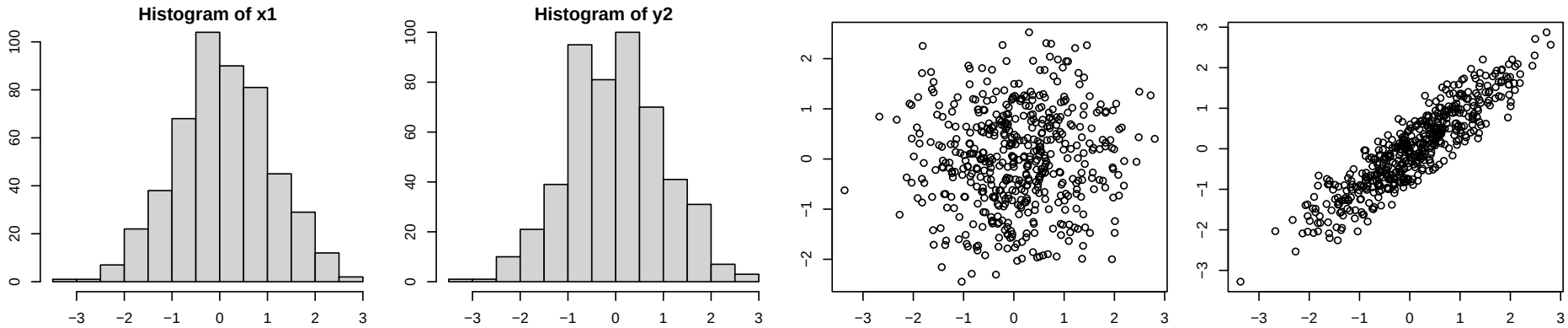
Introduction to Bioinformatics (GS011143)

THE UNIVERSITY OF TEXAS
MD Anderson Cancer Center
Making Cancer History®

Correlation

Scatterplot is the basic plot to show correlation between two random variables. We consider simulated bivariate data under two scenarios of uncorrelated and correlated cases.

```
set.seed(2707);x1 <- rnorm(500,0,1); y1 <- rnorm(500,0,1)
y2 <- 2*x1 + y1 # y2 is linearly related with x1 and y1
y2 <- y2-mean(y2) # center y2
y2 <- y2 / sd(y2) # scale y2
par(mfrow=c(1,4),mar=c(2.5,2.5,1,1))
hist(x1); hist(y2);plot(x1, y1); plot(x1, y2)
```



Correlation

```
cor(x1,y1)

## [1] 0.05340262

cor(x1,y2)

## [1] 0.9034111
```

Correlation

When there are more than two variables, we can check pairwise correlations. We revisit the TCGA BRCA data.

```
expdat = read.table("resExp_filtered.txt",header=TRUE)
dim(expdat)

## [1] 764 551

expdat[1:3,1:3]

##   dat...1.         A0D9         A0DB
## 1      BAD  0.07483513  0.100939142
## 2   MAD1L1  0.08422670 -0.004623776
## 3   CASP10 -0.01402855 -0.006260192
```

Correlation

```
mat = expdat[,-1] # remove the first column for gene names
is.data.frame(mat)

## [1] TRUE

mat = matrix(as.numeric(unlist(mat)),ncol=ncol(mat)) # transf
mat[1:3,1:3]

##              [,1]          [,2]       [,3]
## [1,]  0.07483513  0.100939142  0.2156010
## [2,]  0.08422670 -0.004623776 -0.0982547
## [3,] -0.01402855 -0.006260192 -0.1892582

genes = expdat[,1]
head(genes)

## [1] "BAD"    "MAD1L1" "CASP10" "CFLAR"  "CDC27"  "CREBBP"
```

Correlation

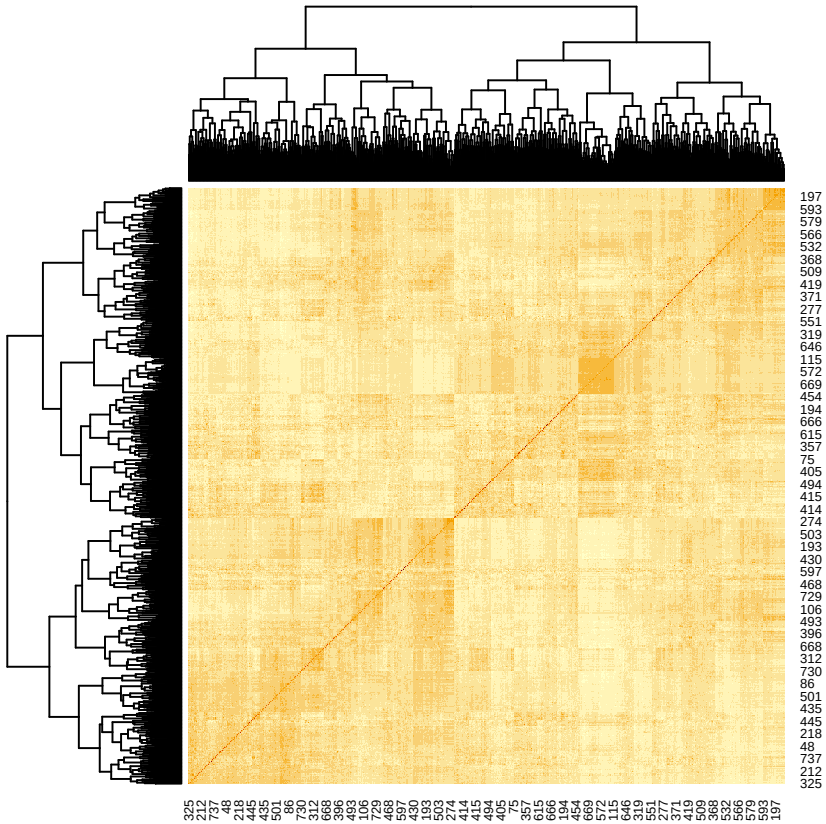
Using the 764×550 data, we are interested in identifying correlation patterns of genes. Recall that genes were stored row-wise in the data. The cor() function takes a matrix to compute all pairwise correlations among columns of the matrix. So we transpose the expression data and then apply the cor() function.

```
cormat = cor(t(mat))
dim(cormat)

## [1] 764 764

cormat[3,4] # Correlation between genes in the rows 3 and 4 o

## [1] 0.4754576

all(cormat == t(cormat)) ### Symmetric

## [1] TRUE
```

Correlation

We have 764×764 correlation matrix, which is symmetric and has unit diagonal values. Due to the large number of genes, it is hard to investigate the correlation pattern. Try heatmap!

```
heatmap(cormat)
```



Distance Metrics

There are various metrics to define distance between two vectors x and y with the same length.

```
x = rnorm(100,mean=3)
y = rnorm(100,mean=5)
# Euclidian Distance
euclidian_dist1 = sqrt(sum((x-y)^2))
euclidian_dist2 = dist(rbind(x,y))
euclidian_dist1
```

```
## [1] 25.18188
```

```
euclidian_dist2
```

```
##          x
## y 25.18188
```

Distance Metrics

```
# Manhattan Distance
manhattan_dist1 = sum(abs(x-y))
manhattan_dist2 = dist(rbind(x,y),method="manhattan")
manhattan_dist1
```

```
## [1] 225.9181
```

```
manhattan_dist2
```

```
##          x
## y 225.9181
```

```
# 1-abs(cor) #
1- abs(cor(x,y))
```

```
## [1] 0.7790254
```

Principal Component Analysis (PCA)

The purpose of principal component analysis is to find the best low-dimensional representation of the variance in a multivariate dataset. Before performing PCA, it would be better idea to first standardize all variables so that measurements for each variable have mean 0 and sd 1. Note that however, sometimes working with original data (before standardization) works better if the scales of variables are similar. We will investigate the crab data in the MASS library.
We draw scatter plots for the five morphological measures by species and sex.

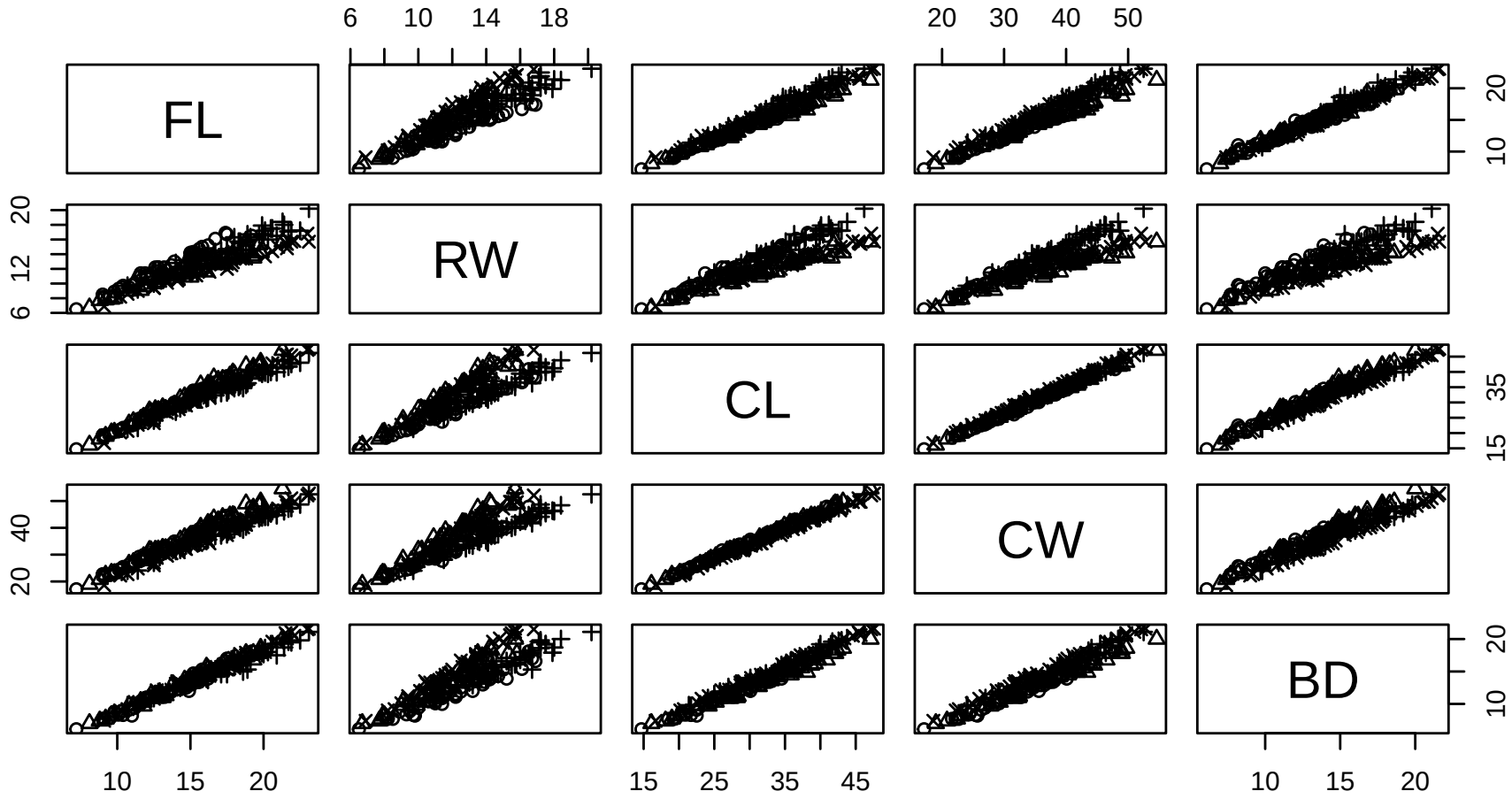
```
library(MASS)
data(crabs)
is.data.frame(crabs)

## [1] TRUE
```

Principal Component Analysis (PCA)

```
dim(crabs)

## [1] 200   8

head(crabs,2) # species - "B" or "O" for blue or orange.

##   sp sex index  FL  RW   CL   CW  BD
## 1  B   M     1 8.1 6.7 16.1 19.0 7.0
## 2  B   M     2 8.8 7.7 18.1 20.8 7.4

fac <- as.factor(paste(crabs[,1],crabs[,2],sep="_"))
table(fac)

## fac
## B_F B_M O_F O_M
##  50  50  50  50
```

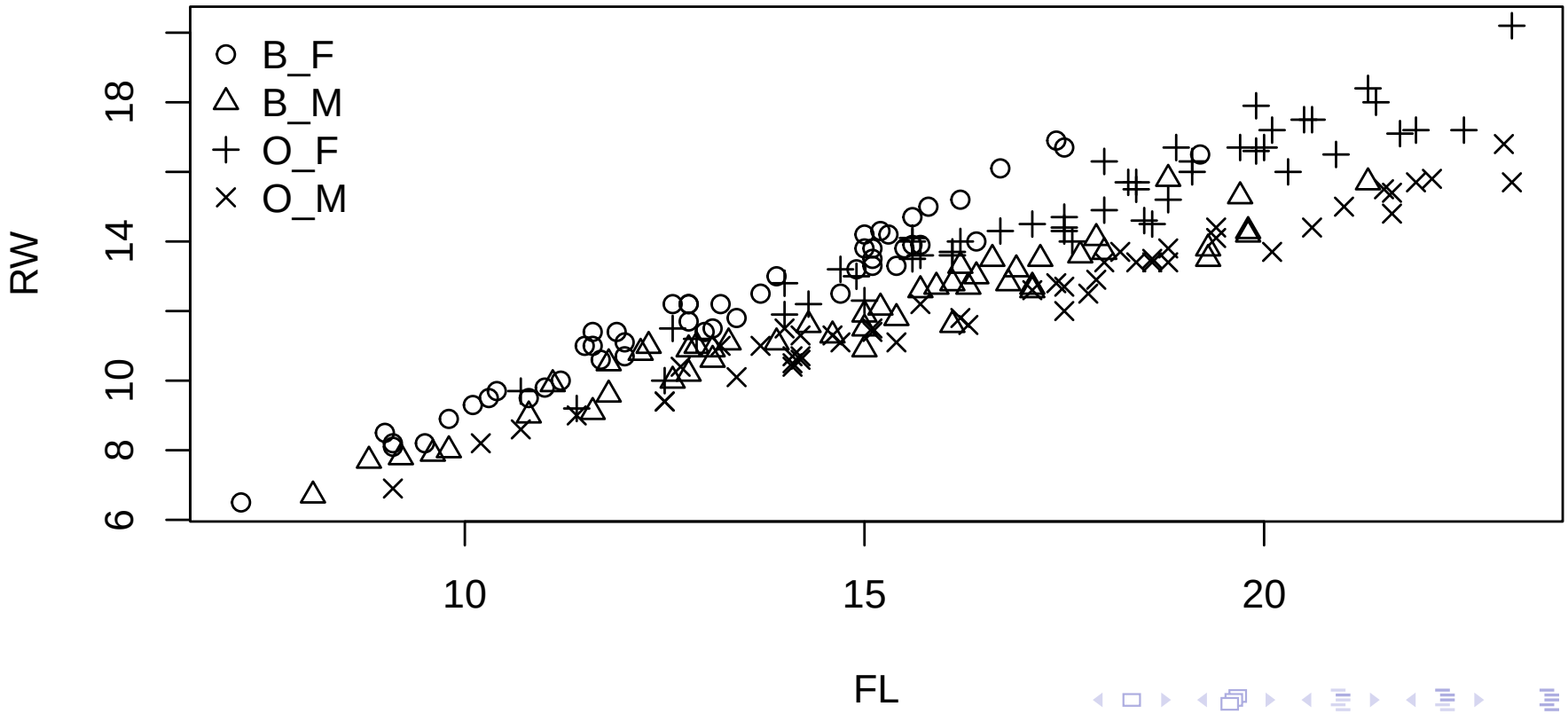
Crabs data

```
plot(crabs[,4:8], pch=as.numeric(fac))
```



Crabs data

```
plot(crabs[,4:5], pch=as.numeric(fac))
legend("topleft",legend=levels(fac),pch=1:4,bty="n")
```



Crabs data

There are four distinct groups of crabs by species and sex, but it is difficult to classify crabs. The 5 measures are highly correlated:

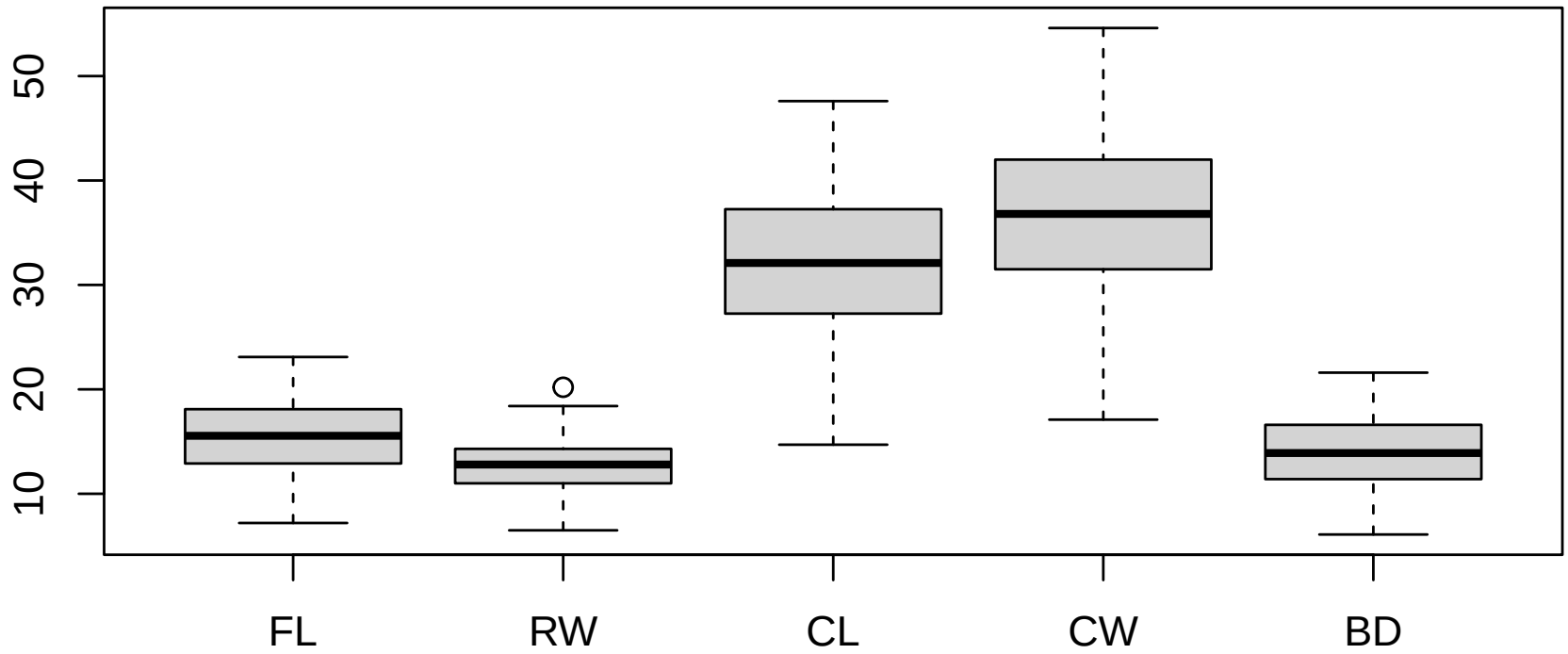
```
cor(crabs[,4:8])
```

```
##           FL        RW        CL        CW        BD
## FL 1.0000000 0.9069876 0.9788418 0.9649558 0.9876272
## RW 0.9069876 1.0000000 0.8927430 0.9004021 0.8892054
## CL 0.9788418 0.8927430 1.0000000 0.9950225 0.9832038
## CW 0.9649558 0.9004021 0.9950225 1.0000000 0.9678117
## BD 0.9876272 0.8892054 0.9832038 0.9678117 1.0000000
```

The information in the crab data is redundant. We want to make more parsimonious data by PCA.

Crabs data

```
?princomp # always check the help file
mat = crabs[,4:8] # try with scale(crabs[,4:8])
boxplot(mat) # check distribution for each variable
```



Crabs data

```
apply(mat,2,mean) # check if mean is all zero
```

```
##      FL      RW      CL      CW      BD
## 15.5830 12.7385 32.1055 36.4145 14.0305
```

```
apply(mat,2,sd) # check if sd is all 1
```

```
##       FL       RW       CL       CW       BD
## 3.495325 2.573340 7.118983 7.871955 3.424772
```

```
fit = princomp(mat) # fit PCA to the the standardized
summary(fit)
```

```
## Importance of components:
##                             Comp.1      Comp.2      Comp.3
## Standard deviation     11.8322521 1.135936870 0.997631086
## Proportion of Variance  0.9824718 0.009055108 0.006984337
## Cumulative Proportion   0.9824718 0.991526908 0.998511245
```

Crabs data

When we fit PCA for $n \times p$ data, we obtain p principal components (transformed data), that are in the $n \times p$ score matrix object

```
dim(fit$scores)

## [1] 200   5

head(fit$scores,3)

##       Comp.1     Comp.2     Comp.3      Comp.4       Comp.5
## 1 -26.46457 -0.5765335  0.6115677 -0.02868117  0.49658452
## 2 -23.56174 -0.3364196  0.2373877  0.02220942 -0.01652072
## 3 -21.74319 -0.7118646 -0.0654972  0.18255551  0.23740504

cor(fit$scores) ## cor for the transformed data

##                Comp.1        Comp.2        Comp.3        Com
## Comp.1  1.000000e+00 -2.474785e-15 -2.173229e-15 -6.293437e
## Comp.2 -2.474785e-15  1.000000e+00 -7.354758e-16 -2.451698e
```

Crabs data

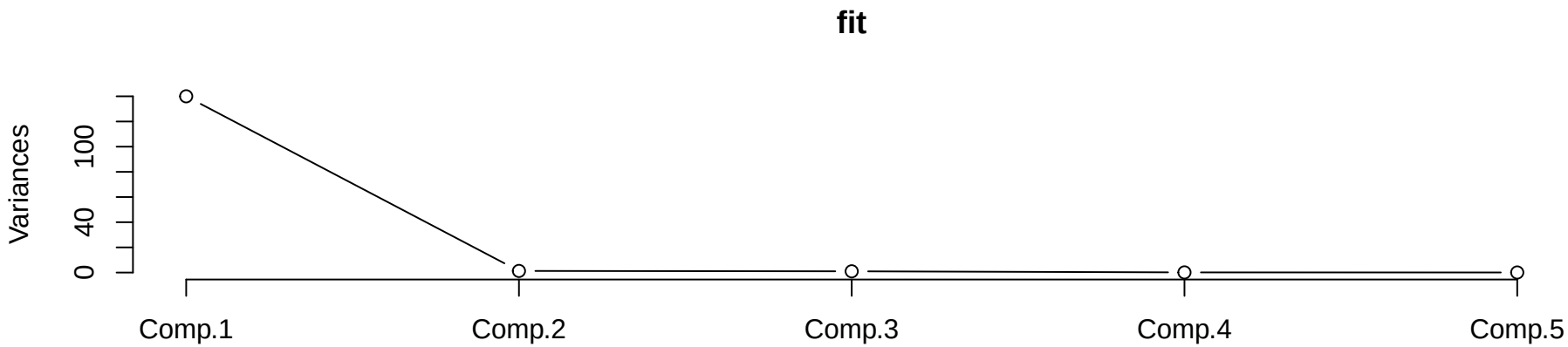
Now, you can check that the p principle components are not correlated. For dimension reduction, we need to reduce the number of variables p to k such as $k < p$ by selecting the most informative principal components. The principal component that have high variance is informative to explain the original data.

```
sd.scores = apply(fit$scores,2,sd) # sd for principal componen
sum(sd.scores^2) # should be the same as the below

## [1] 143.216

sum(apply(mat,2,var))

## [1] 143.216
```

Crabs data

The total variance of PCs should be the same as total variance of the orginal data. The total variances from original data and the transformed data should be the same. Now, we select the principal components by screeplot.

```
screeplot(fit,type="lines")
```



Crabs data

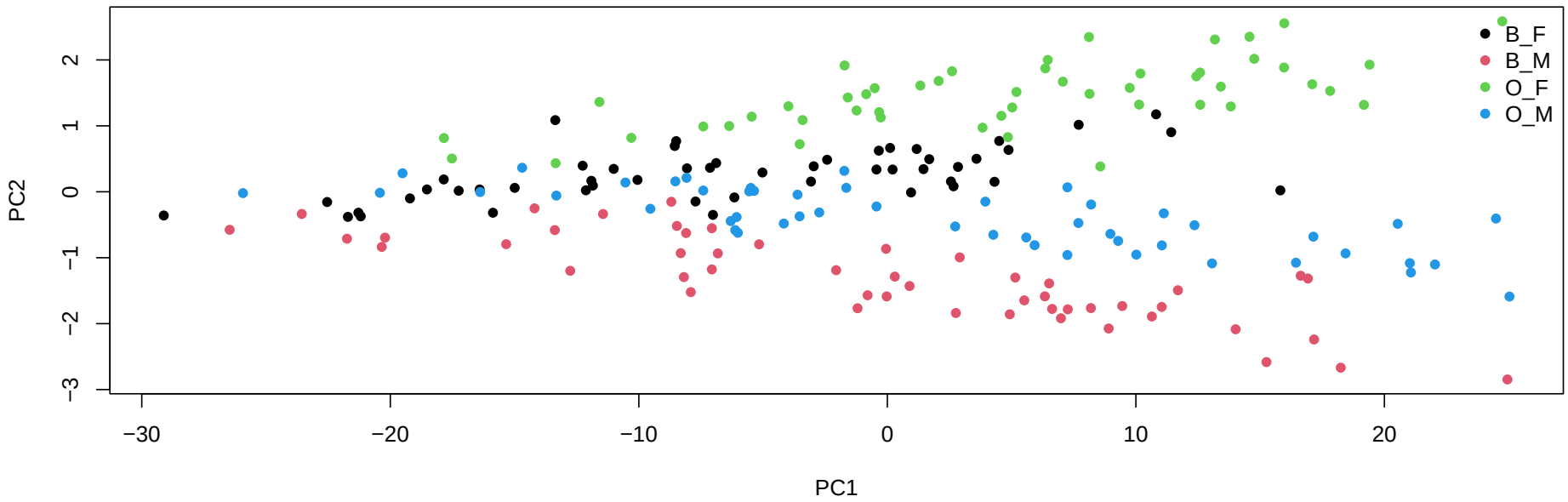
PC1 explains 98% of the total variance. If we add PC2, PC1 and PC2 explain 99%.

```
sd.scores[1]^2/sum(sd.scores^2)

##    Comp.1
## 0.9824718

# Proportion of variance explained by PC1
```

Crabs data

It was expected when we looked at the pairwise correlations of the original data: all the correlations were so high, which means that all the five variables are very similar acting like one variable.

```
par(mar=c(4.5,4.5,1,1))
plot(fit$score[,1],fit$score[,2],pch=16,col=as.numeric(fac),
xlab="PC1", ylab="PC2")
legend("topright",legend=levels(fac),pch=16, col=1:4,bty="n")
```



PCA example

We will go back to the TCGA Breast Cancer data. We first standardize the data.

```
mat = expdat[,-1] # remove the first column for gene names
mat = matrix(as.numeric(unlist(mat)),ncol=ncol(mat)) # transf
genes = expdat[,1]
stdmat = t(scale(t(mat)))
```

PCA example

For high-dimensional case, where the number of genes are larger than the sample size ($p > n$), the princomp function in R does not work. Instead, we need use prcomp function that works regardless of the data dimension. Perform PCA.

```
fit <- prcomp(t(stdmat))
names(fit)

## [1] "sdev"     "rotation" "center"   "scale"    "x"

dim(fit$x) # transformed data (scores)

## [1] 550 550

fit$x[1:2,1:2]

##             PC1        PC2
## [1,]  5.487394 -6.9755328
## [2,] 10.727838  0.4773371
```

PCA example

```
cor(fit$x)[1:5,1:3] # Uncorrelated
```

```
##                PC1           PC2           PC3
## PC1  1.000000e+00  3.807276e-16 -4.097914e-18
## PC2  3.807276e-16  1.000000e+00  1.593344e-16
## PC3 -4.097914e-18  1.593344e-16  1.000000e+00
## PC4  9.941794e-17 -7.672926e-17 -2.112759e-16
## PC5 -1.698408e-16  3.079954e-16  5.304797e-16
```

PCA example

Now, using prcomp function, the x object includes n PCs. Note that when $p > n$, n number of PCs are fitted. Now choose number of PCs.

```
vars = as.numeric(apply(fit$x,2,var)) # variances
sum(vars) # should be the same as # of genes by using std. da
```

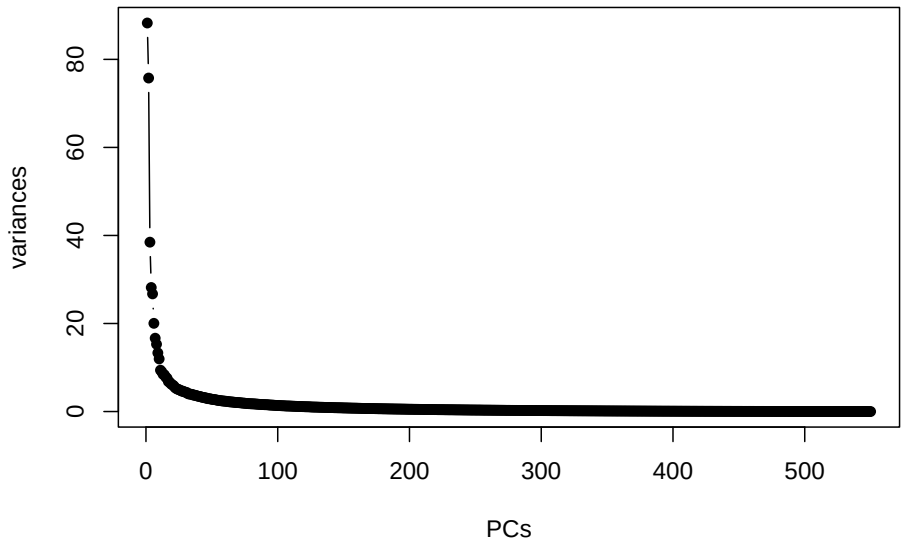
```
## [1] 764
```

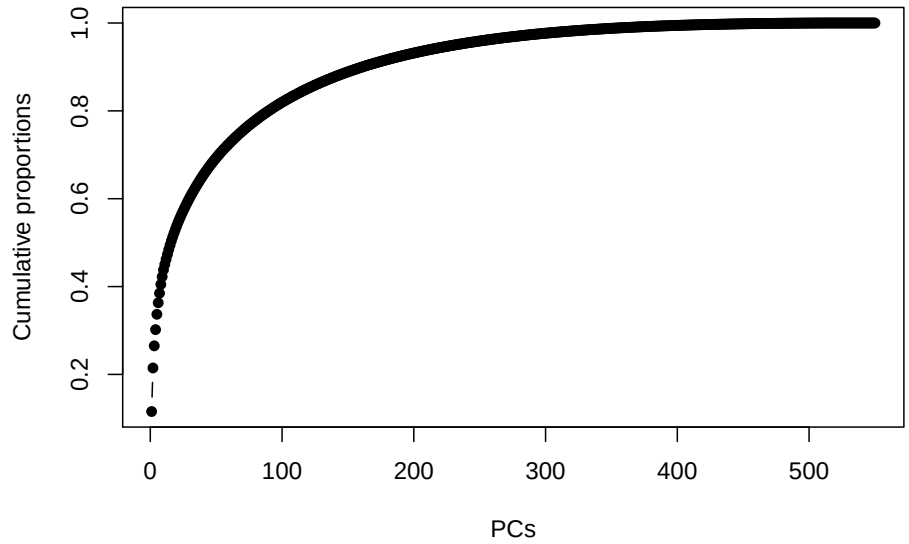
```
cumprops = cumsum(vars)/sum(vars)
# cumulative proportions of variances
head(cumprops,3)
```

```
## [1] 0.1155288 0.2146924 0.2650518
```

PCA example

```
par(mfrow=c(1,2),mar=c(4.5,4.5,1,1))
plot(vars,type="b",pch=16,ylab="variances",xlab="PCs")
plot(cumprops,type="b",pch=16,
ylab="Cumulative proportions",xlab="PCs")
```

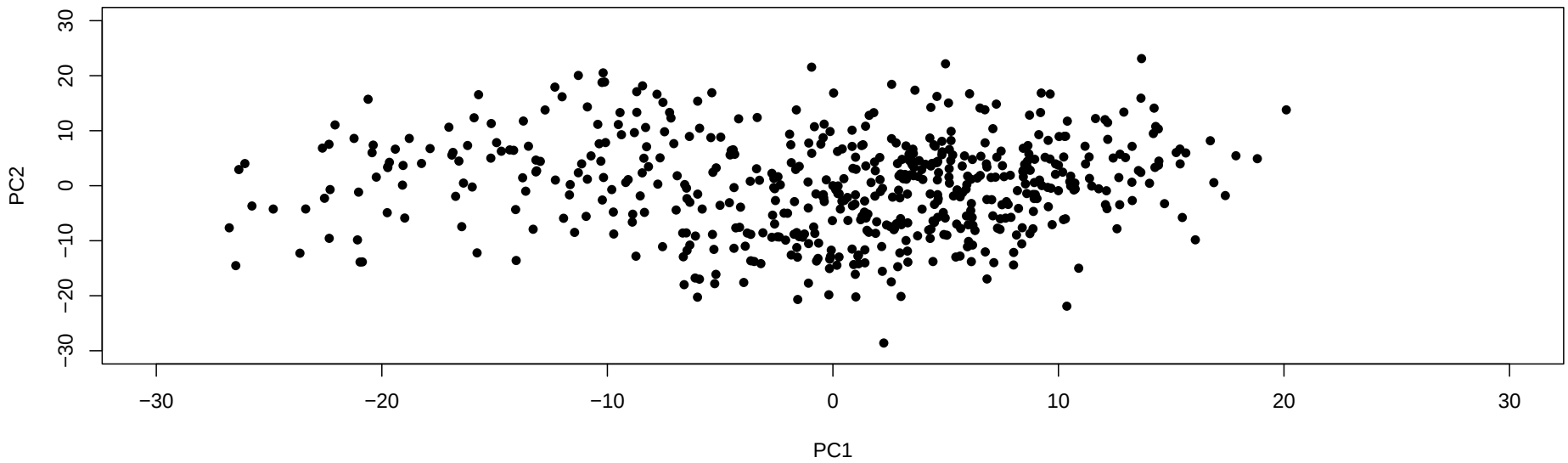




PCA example

We will need 90 PCs to explain around 80% of the total variance. In this ordination method, the data points (i.e., the samples) can be projected onto the 2D plane by the top 2 PCs such that they spread out optimally. Let's check sample distances on the PC1 and PC2 plane.

```
par(mfrow=c(1,1),mar=c(4.5,4.5,1,1))
plot(fit$x[,1],fit$x[,2],pch=16,xlab="PC1",
ylab="PC2",xlim=c(-30,30),ylim=c(-30,30))
```



PCA example

```
vars[1] / sum(vars) # % variance explained by PC1
```

```
## [1] 0.1155288
```

```
vars[2] / sum(vars) # % varinace explained by PC2
```

```
## [1] 0.09916357
```

TCGA data example

- A useful first step in a multivariate (high-dimensional) data analysis is to assess overall similarity between samples, i.e. your experimental design. For example, which samples are similar to each other, which are different? Does this fit to the expectation from the experiment design? If you have control and treated groups of patients, the patients within group should be clustered together.
- Using TCGA breast cancer example, we can check transcriptomic similarities by tumor receptor subtypes, ER/PR positive, HER2 positive and Triple negative.
- We load clinical data for the 550 patients. The columns of expdat and rows of the clinical data are matched.

TCGA data example

```
cldat = read.table("TCGABRCA_cldat.txt",header=TRUE)
dim(cldat)

## [1] 550  30

head(cldat[,1]) # patient id for clinical data

## [1] "TCGA-A7-A0D9" "TCGA-A7-A0DB" "TCGA-A7-A13G" "TCGA-BH-A
## [6] "TCGA-BH-A0AZ"

head(colnames(expdat)[-1]) # patient id for gene expression da

## [1] "A0D9" "A0DB" "A13G" "A0AU" "A0AY" "A0AZ"
```

TCGA data example

The patient ids from the two datasets, colnames(expdat)[-1] and cldat[,1] should be matched. For example, A0D9 in expdat and TCGA-A7-A0D9 represent the same patient. Then we construct the subtype based on ER, PR and HER2 statuses:

```
ER = cldat[,"ER.Status"]
PR = cldat[,"PR.Status"]
HER2 = cldat[,"HER2.Final.Status"]
Subtype = rep(NA,nrow(cldat))# Tumor Receptor subtype
Subtype[ER=="Positive"|PR=="Positive"] = "ER/PR positive"
Subtype[HER2=="Positive"] = "HER2 positive"
Subtype[ER=="Negative"&PR=="Negative"&HER2=="Negative"] =
"Triple negative"
```

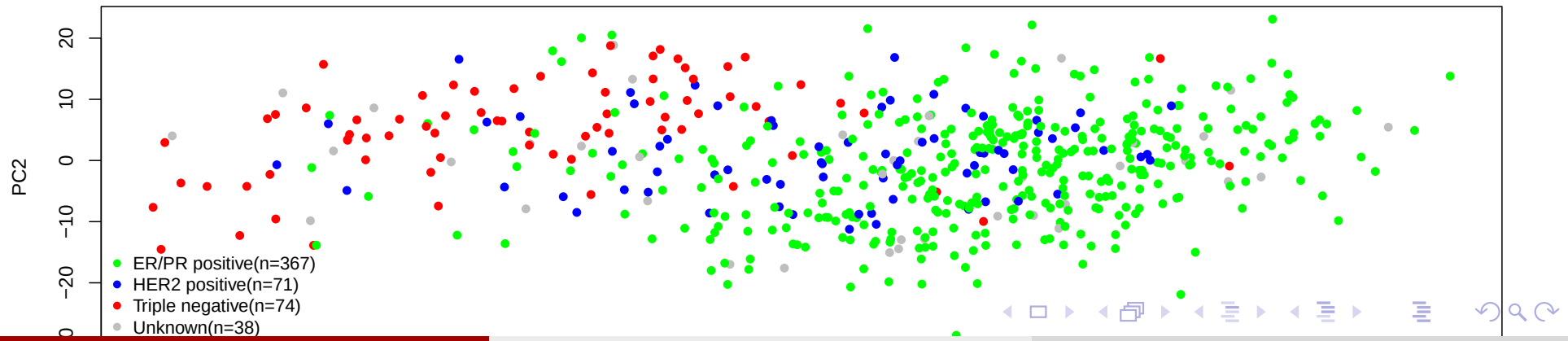
TCGA data example

```
table(Subtype,useNA="ifany")

## Subtype
##  ER/PR positive   HER2 positive Triple negative
##             367              71              74

Subtype[is.na(Subtype)] = "Unknown"
Subtype = as.factor(Subtype)
```

TCGA data example

Now, we have subtype information for the patients. We can label each points of the scatter plot for PC1 and PC2 by the subtype info.This plot looks more informative: Triple negative patients are genomically different from ER/PR positive patients.

```
cols= c("green","blue","red","grey")
par(mfrow=c(1,1),mar=c(4.5,4.5,1,1))
plot(fit$x[,1],fit$x[,2],pch=16,
col=cols[as.numeric(Subtype)],xlab="PC1",ylab="PC2")
legend("bottomleft",legend=paste(levels(Subtype),"(","n=",
table(Subtype),")",sep=""),pch=16,col=cols,bty="n",cex=0.9)
```



TCGA data example

Now, let's try performing the dimension reduction using t-SNE

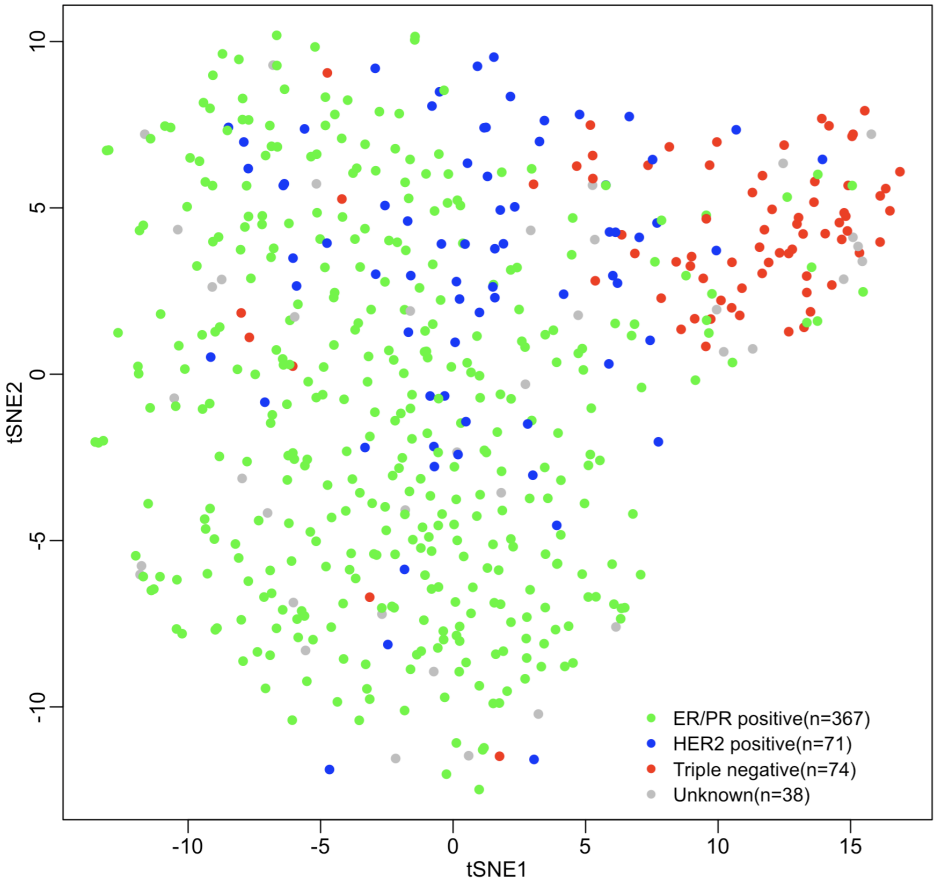
```
# install.packages("Rtsne")
library(Rtsne)
res = Rtsne(t(stdmat))
cols= c("green","blue","red","grey")
par(mfrow=c(1,1),mar=c(4.5,4.5,1,1))
plot(res$Y[,1:2],pch=16,
     col=cols[as.numeric(Subtype)],xlab="tSNE1",ylab="tSNE2")
legend("bottomright",legend=paste(levels(Subtype),"(","n=",
                    table(Subtype),")",sep=""),
                    pch=16,col=cols,bty="n",cex=0.9)
```

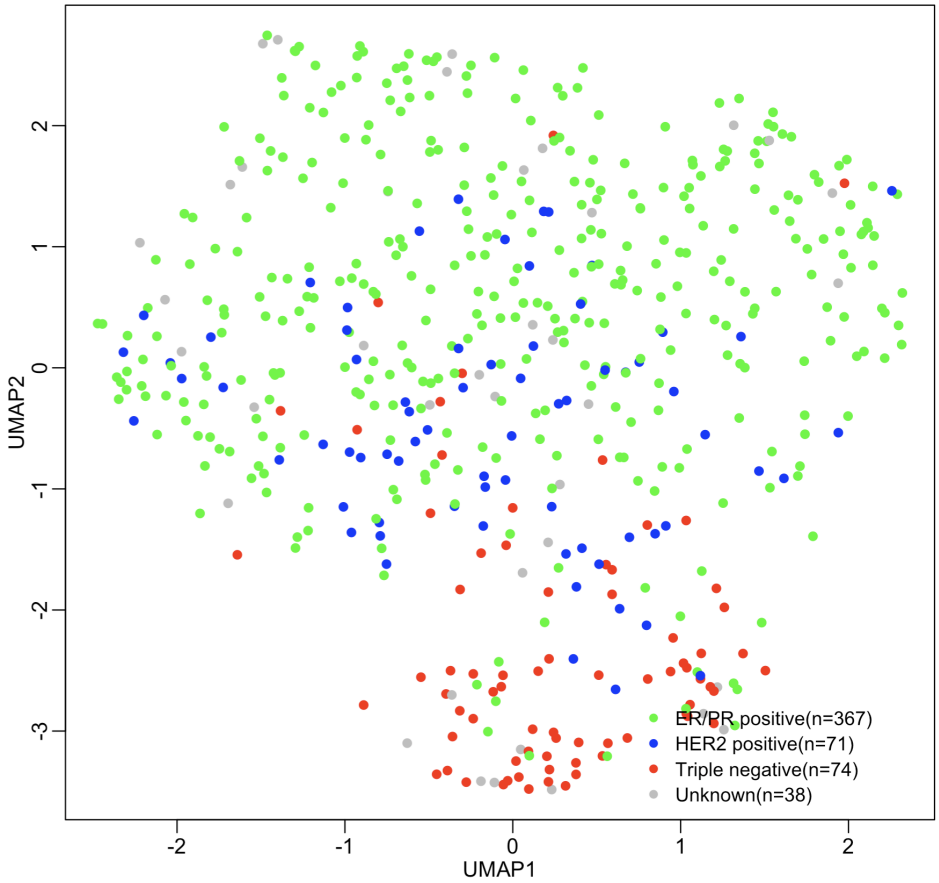
TCGA data example

Now, let's try performing the dimension reduction using UMAP

```
# install.packages("umap")
library(umap)
res = umap(t(stdmat))
cols= c("green","blue","red","grey")
par(mfrow=c(1,1),mar=c(4.5,4.5,1,1))
plot(res$layout[,1:2],pch=16,
     col=cols[as.numeric(Subtype)],xlab="UMAP1",ylab="UMAP2")
legend("bottomright",legend=paste(levels(Subtype),"(","n=",
                    table(Subtype),")",sep=""),
                    pch=16,col=cols,bty="n",cex=0.9)
```

tSNE and UMAP figures in TCGA example





K-means

Try an animation for K-means.

```
library(animation)
dev.new()
dat = cbind(X1 = rnorm(100),X2=rnorm(100))
kmeans.ani(dat,centers=3,pch=1:3,col=1:3)
```

K-means

You can interpret the animation: (1) Randomly choose three points; (2) compute Euclidian distance and draw the assigned clusters; (3) Compute the centroids, the mean of the clusters, and repeat until no data changes cluster.

```
?iris
dim(iris);head(iris,2)

## [1] 150   5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa

km <- kmeans(iris[,1:4], 3)
plot(iris[,1], iris[,2], col=km$cluster)
points(km$centers[,c(1,2)], col=1:3, pch=8, cex=2)
```

K-means

```
table(km$cluster, iris$Species)
```

```
##
##      setosa versicolor virginica
##   1      50          0         0
##   2       0         48        14
##   3       0          2        36
```

K-means

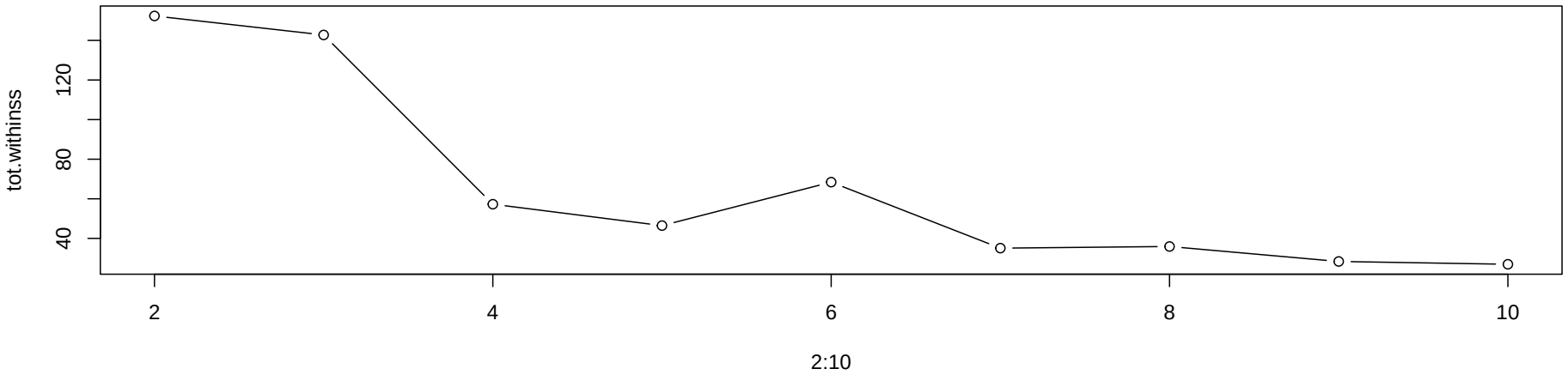
One possible way to choose the K (the number of clusters) is to use within-group heterogeneity. Consider to create the function that runs $K-$means with fixed K and store the total within clusters sum of squares.

```
kmean_withinss <- function(K,dat) {
    cl <- kmeans(dat,K)
    return(cl$tot.withinss)
}
```

K-means

Then, you can fit K-means for $K = 2, \cdots, 10$ and draw the plot for the total within sum of squares.

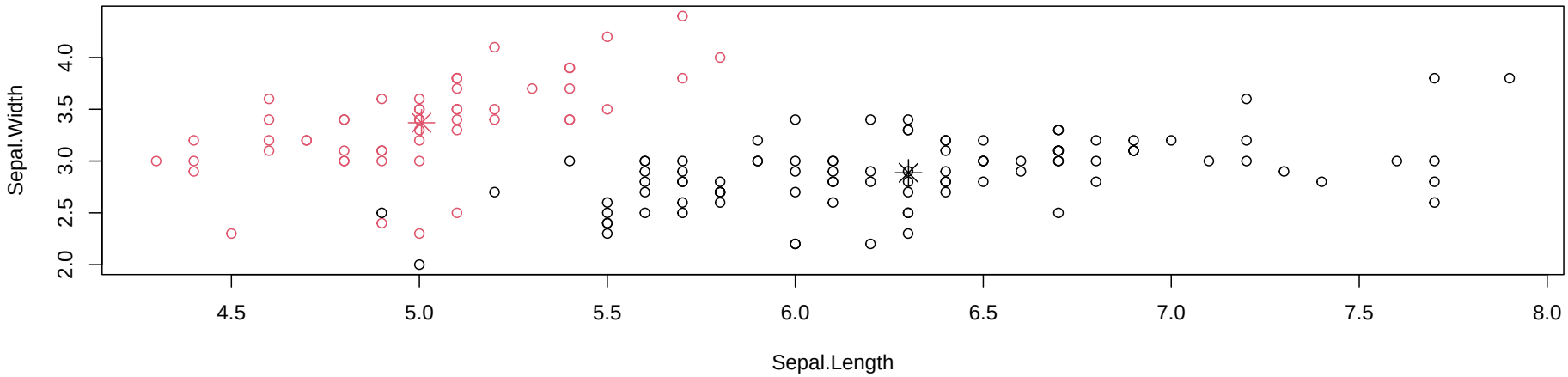
```
tot.withinss = sapply(2:10,
function(k) kmean_withinss(K=k,dat=iris[,1:4]))
plot(2:10,tot.withinss,type="b")
```



K-means

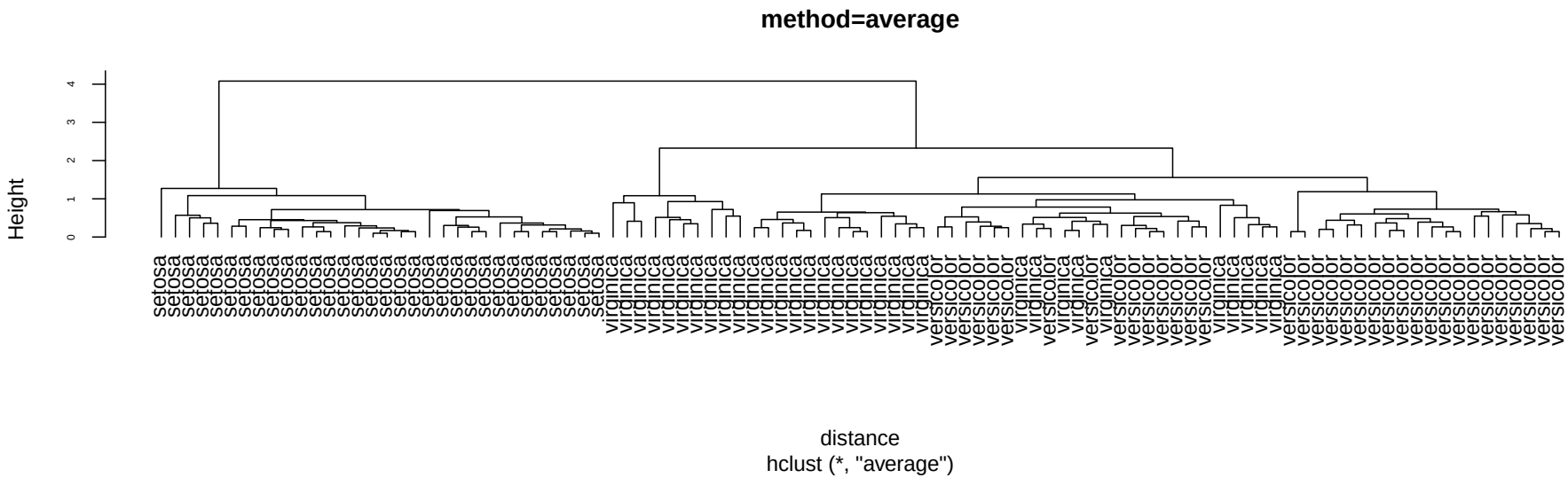
If we select K=2, how does it look like?

```
km <- kmeans(iris[,1:4], 2)
plot(iris[,1:2], col=km$cluster)
points(km$centers[,c(1,2)], col=1:2, pch=8, cex=2)
```



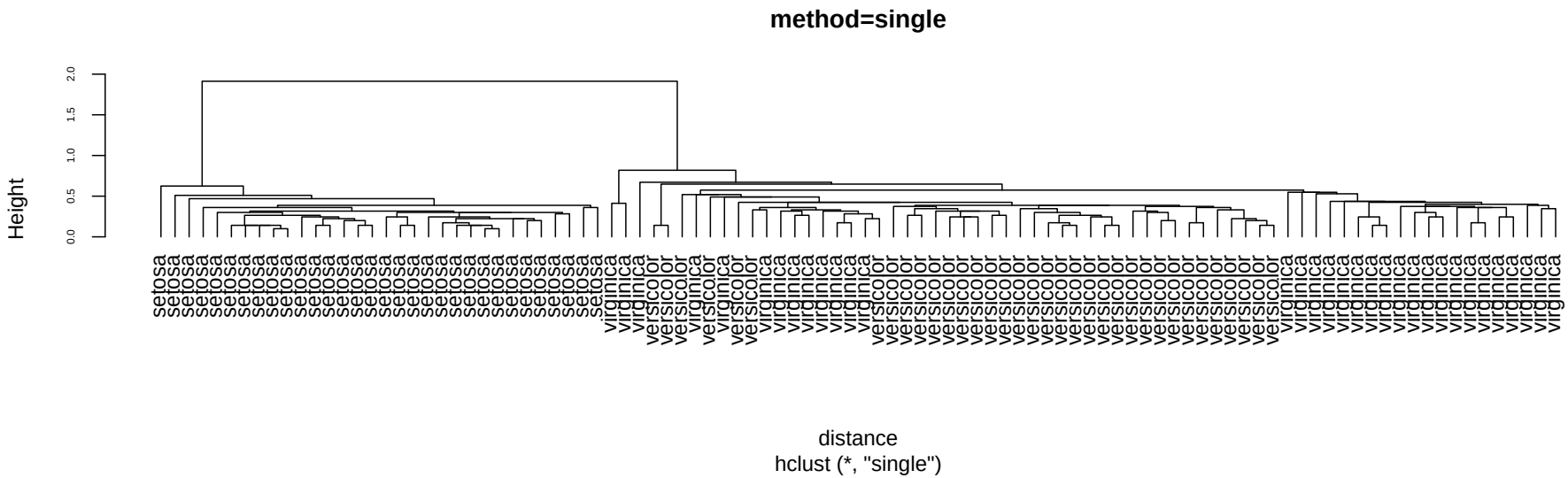
Hierarchical clustering

```
s.iris = iris[sample(1:150,100),]
distance <- dist(s.iris[,-5], method="euclidean")
##vary euclidean distances
cluster <- hclust(distance, method="average")
plot(cluster, hang=-1, label=s.iris$Species,
main="method=average",cex.axis=0.5)
```



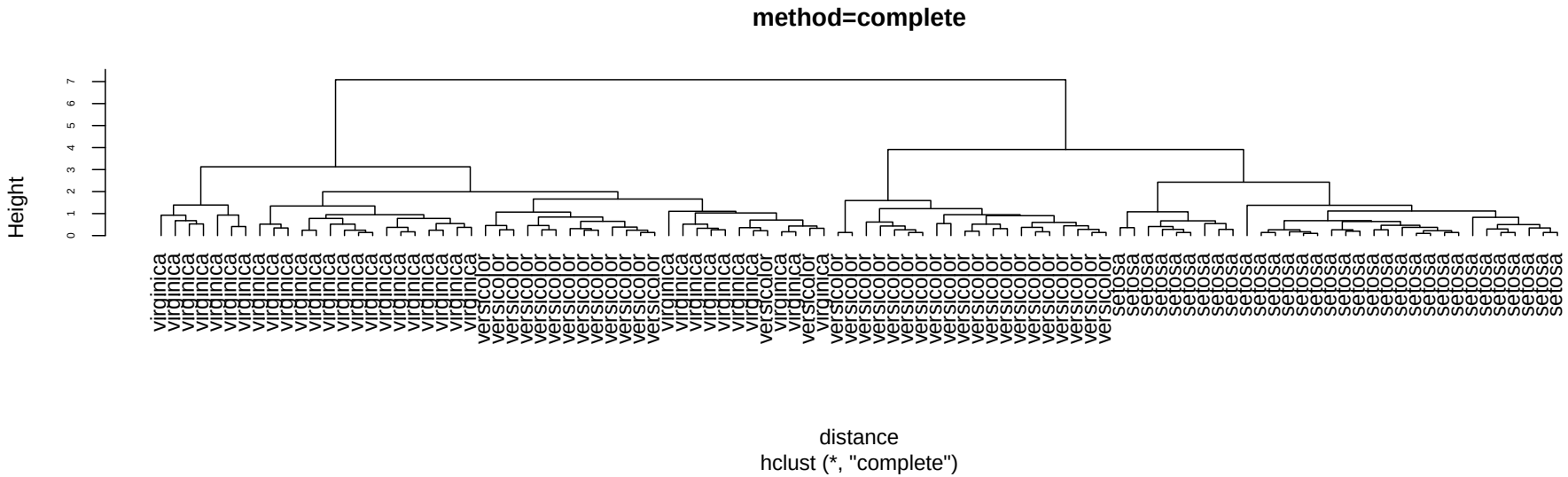
Hierarchical clustering

```
cluster <- hclust(distance, method="single")
plot(cluster, hang=-1, label=s.iris$Species,
main="method=single",
cex.axis=0.5)
```



Hierarchical clustering

```
cluster <- hclust(distance, method="complete")
plot(cluster, hang=-1, label=s.iris$Species,
main="method=complete",cex.axis=0.5)
```



Hierarchical clustering

```
mems = cutree(cluster,k=3)
table(mems,s.iris$Species)

##
## mems setosa versicolor virginica
##    1     32          0         0
##    2      0         15        33
##    3      0         20         0
```