Module 12. Multivariate Data and Principal Component Analysis

Overview:

In this module we will study multivariate distributions and displaying multivariate data. We will also study another unsupervised learning technique: the principal component analysis (PCA).

Learning Objectives

1. Generate data from the **multivariate normal distribution**
2. **Visualize multivariate data** in R
3. Apply **principle component analysis** for dimension reduction

Readings:

Statistics Using R with Biological Examples pages 288-301.

Applied Statistics for Bioinformatics using R pages 133-141.

Lesson 1: Multivariate Normal Distribution and Multivariate Data Display

Objectives

By the end of this lesson you will have had the opportunity to:

- Generate random samples from a **multivariate normal** distribution
- **Calculate sample statistics** for multivariate data
- **Display multivariate data** in 1D, 2D, and 3D plots

Overview

In this lesson, we will introduce the basic definitions of a multivariate normal distribution. We will review some probability properties of normal distributions. Then we will introduce R commands for generating data from the multivariate normal distribution. Finally, we will summarize and display multivariate data using R commands.

Review of Univariate and Bivariate Normal Distributions

We have covered the univariate and bivariate normal distributions previously.

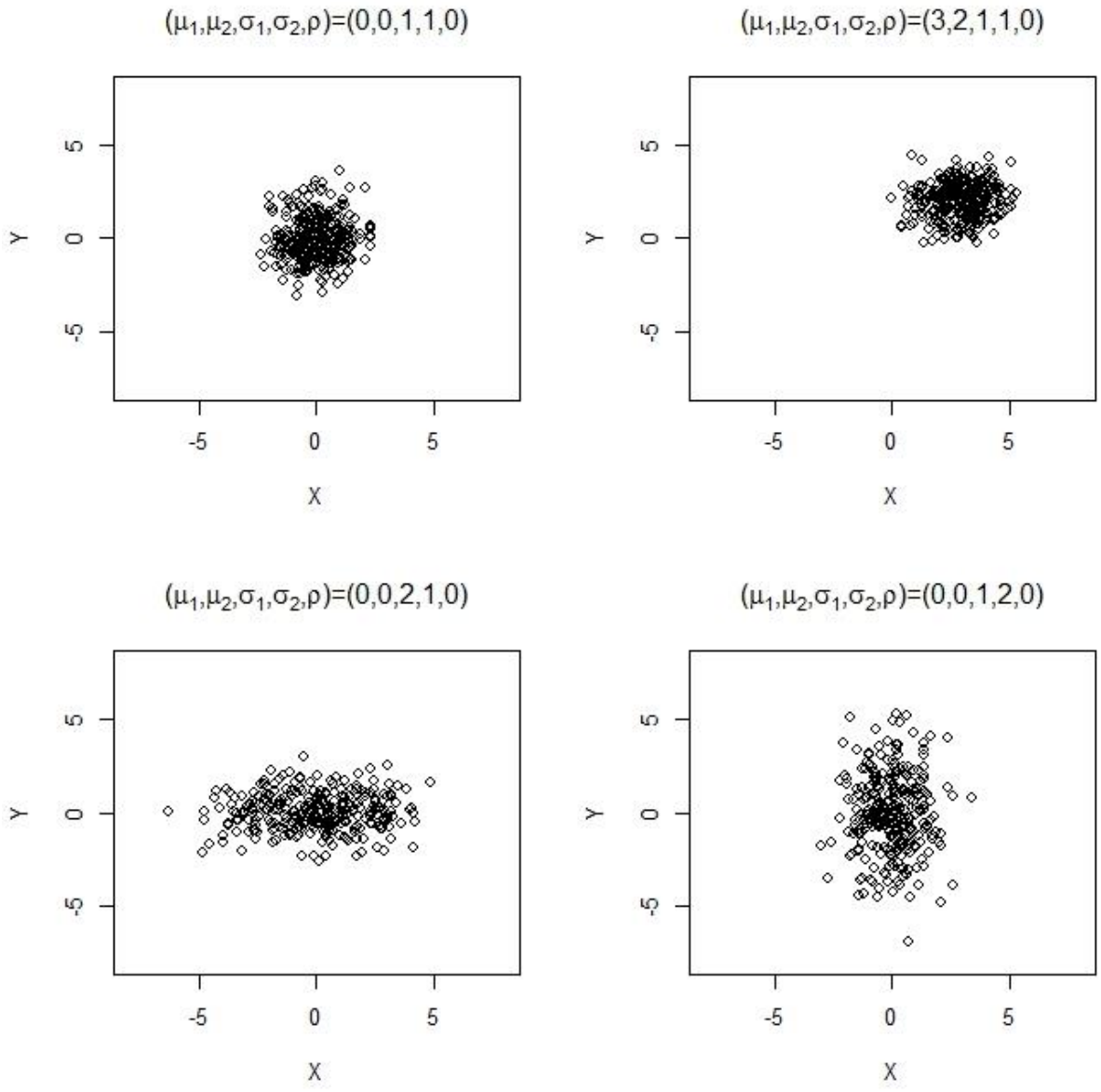
The univariate normal distribution $N(mean = \mu, sd = \sigma)$ is bell-shaped with mean μ and standard deviation σ.

The bivariate normal distribution $N(mean = (\mu_1, \mu_2), \text{var} = \Sigma)$, with variance matrix $\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$, is the distribution of a 2-dimensional random vector (X,Y), where X and Y each follow the univariate normal distribution $N(\mu_1, \sigma_1)$ and $N(\mu_2, \sigma_2)$ respectively, and the covariance between X and Y is σ_{12}. The pdf for the bivariate normal distribution pdf is a hill-shaped 2-dimensional function. We can also characterize the distribution in terms of the correlation $\rho = \frac{\sigma_{12}}{\sigma_1 \sigma_2}$ instead of the covariance σ_{12}. The correlation value is always between -1 and 1. That is, $-\sigma_1 \sigma_2 \le \sigma_{12} \le \sigma_1 \sigma_2$.

We would like to extend this distribution to multivariate case. Before we do that, let us first inspect the effect of the parameters $(\mu_1, \mu_2, \sigma_1, \sigma_2, \rho)$. We create the scatterplots of 300 random samples from the bivariate normal distribution with different parameter $(\mu_1, \mu_2, \sigma_1, \sigma_2, \rho)$ values as done in the examples on page 289 of Statistics Using R with Biological Examples. See the graphs on the next page.

Plots of Bivariate Normal Data

Below are the plots of the bivariate normal distribution using our data.

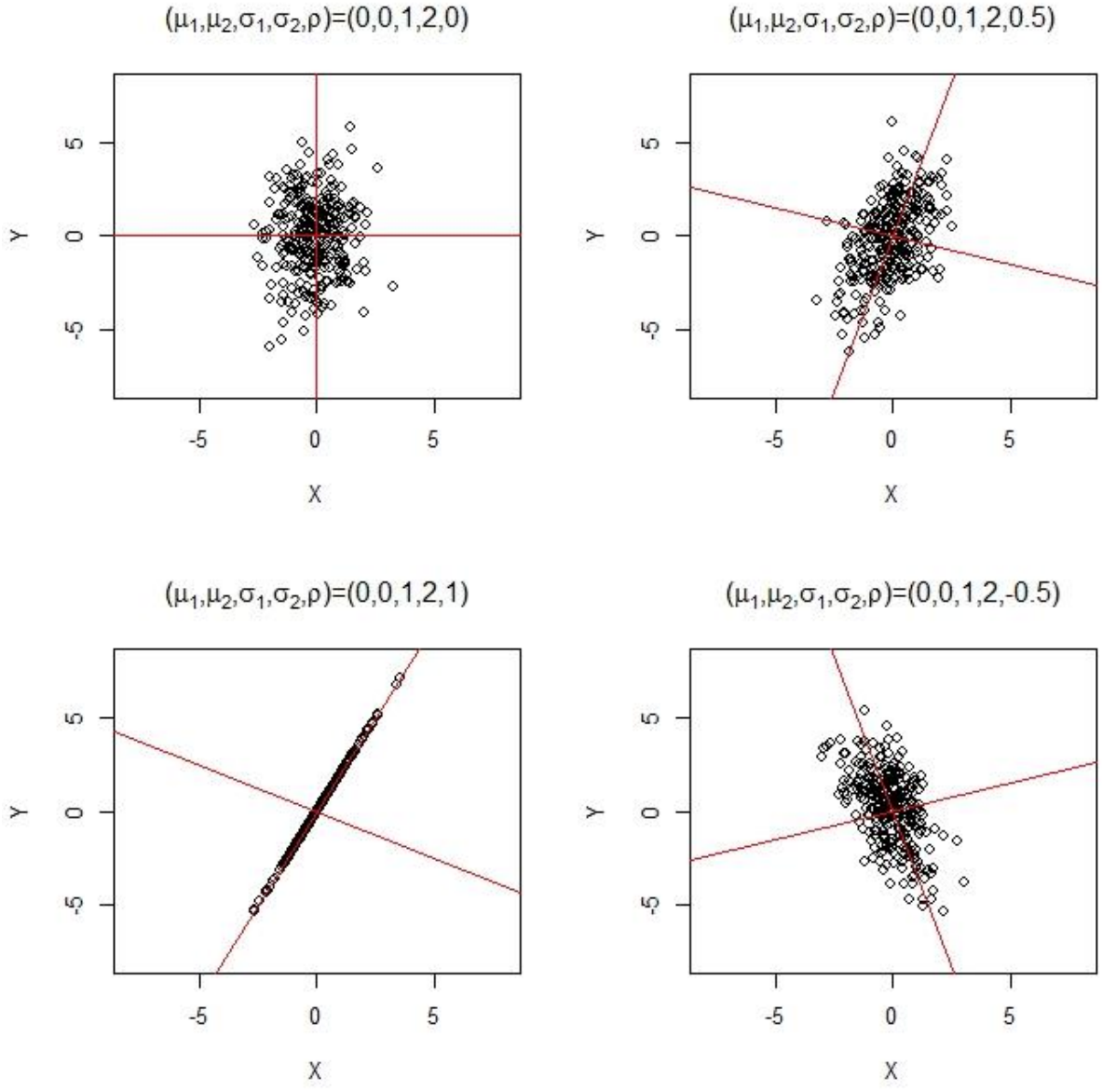


From the plots, we can see that the bivariate normal data is denser in the center, and sparser going out. The mean (μ_1, μ_2) values move the center of the distribution. The σ_1 and σ_2 values stretch the distribution along the X and Y directions, respectively.

We observe the effect of correlation ρ values in the next page.

Plots of Bivariate Normal Data (Cont'd)

The below plots show the data distribution under different correlation ρ values.



The red lines show the axis of the oval shape of the distribution. Those are the principal components we will be looking for in the next lesson. We can see that the correlation rotates the distribution angles. Also, the bigger correlation values squeeze the distribution to the long axis of the oval shape.

Review Properties of Normal Distributions

Let us review some properties from Modules 3 and 4 on the bivariate normally distributed (X, Y) $\sim N(mean = (\mu_1, \mu_2), \text{var} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix})$.

(1) The marginal distributions are also normal distributions. That is, $X \sim N(\mu_1, \sigma_1)$ and $Y \sim N(\mu_2, \sigma_2)$.

(2) The means μ_1 and μ_2 denote the central location of the distribution, while σ_1 and σ_2 reflect the scales for the shape of the distribution.

(3) Any linear combination aX + bY also follow a normal distribution, with mean $E(aX + bY) = a\mu_1 + b\mu_2$ and variance $Var(aX + bY) = a^2\sigma_1^2 + b^2\sigma_2^2 + 2ab\sigma_{12}$.

The bivariate normal distribution is totally characterized by the means μ_1 and μ_2, the standard deviations σ_1 and σ_2, and the covariance σ_{12} (or equivalently the correlation ρ).

For more than two variables, we similarly extend the p-dimensional multivariate normal distribution for X_1, X_2, …, X_p. The multivariate normal distribution is characterized by the marginal means μ_1, …, μ_p, the marginal standard deviations σ_1, …, σ_p, and pairwise covariance σ_{ij} for $1 \le i < j \le p$ (or equivalently the correlation ρ_{ij}).

Multivariate Normal Distributions

The p-dimensional multivariate normal distribution for a random vector (X_1, X_2, …, X_p) is denoted as $N(mean = (\mu_1, .., \mu_p), \text{var} = \Sigma)$ with the variance matrix

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1p} & \sigma_{2p} & \cdots & \sigma_p^2 \end{pmatrix}.$$

Each variable X_i has a marginal distribution as a univariate normal distribution $N(\mu_i, \sigma_i)$. And each pair X_i and X_j follows the bivariate normal distribution $N(mean = (\mu_i, \mu_j), \text{var} = \begin{pmatrix} \sigma_i^2 & \sigma_{ij} \\ \sigma_{ij} & \sigma_j^2 \end{pmatrix})$.

In R, the "mvtnorm" package provides the basic functionalities for multivariate normal distribution. The pdf function is calculated by dmvnorm() function. The rmvnorm() function generates random samples from the multivariate normal distribution.

Next, we repeat an example of a 4-dimensional multivariate normal distribution in the textbook Statistics Using R with Biological Examples. From the properties for the multivariate normal distribution listed on the previous page, we know that *each variable* follows the univariate normal distribution and *each pair of variables* follows the bivariate normal distribution.

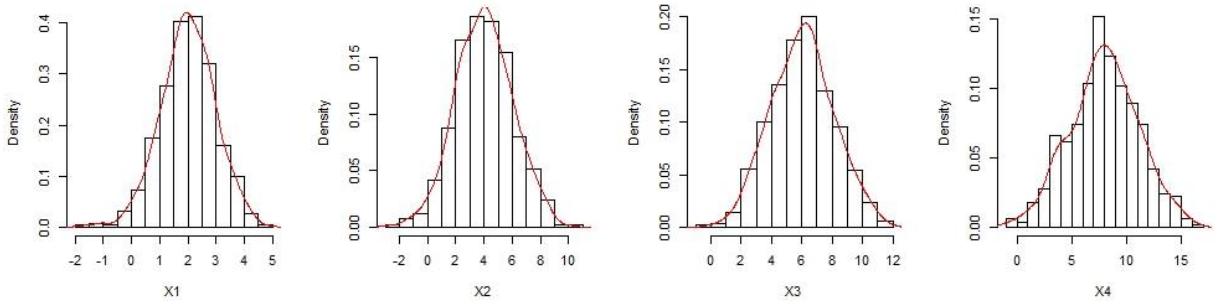
Demonstration: Multivariate Normal Distributions

For illustration, we will generate four random variables X_1, X_2, X_3, X_4 that jointly follow the multivariate normal distribution with mean (2,4,6,8) and variance matrix

$$\Sigma = \begin{pmatrix} 1 & 1 & -0.4 & -0.9 \\ 1 & 4 & 1.6 & -1.2 \\ -0.4 & 1.6 & 4 & 1.8 \\ -0.9 & -1.2 & 1.8 & 9 \end{pmatrix}.$$

```
Sigma <- matrix(c(1,1,-.4,-.9,1,4,1.6,-1.2,-.4,1.6,4, 1.8,-.9,-1.2,1.8,9),
ncol=4)  #Create the 4 by 4 variance matrix.
X <- rmvnorm(500, mean=c(2,4,6,8), sigma=Sigma) #generate 500 random
samples, each a 4-dimensional vector from normal distribution with mean
vector (2,4,6,8) and variance matrix Sigma.
colnames(X)<-paste("X", (1:4), sep="") #variable names X1, X2, X3, X4.
Use empty separator "" to get “X1” instead of “X 1”.
X<-data.frame(X) #Make the samples a data frame. Each column is one
variable containing 500 observations.
par(mfrow=c(1,4)) #Put plots in 1 row and 4 columns
for (i in 1:4) {
        hist(X[,i], main="", nclass=15, freq=FALSE, xlab=colnames(X)[i])
#draw histogram; use 15 blocks instead of default breaks; plot relative
frequency (comparable to density) instead of absolute frequency; use
variables names for label)
        lines(density(X[,i]), col='red') #add a red line for smoothed density
}
```

We plot the histogram of each variable together with their smoothed densities.

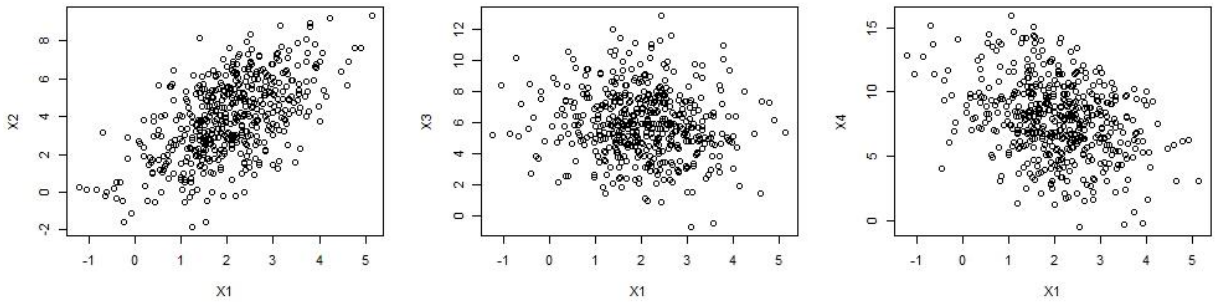


We can observe that each marginal distribution is also normal.

Demonstration: Multivariate Normal Distributions (Cont'd)

We plot the two dimensional scatterplots of X_1 versus X_2, X_3 and X_4 to observe the 2-dimensional marginal distributions.

```
par(mfrow=c(1,3)) #Put plots in 1 row and 3 columns
varlist<-colnames(X)[2:4] #get variable names X2, X3, X4
with (X,                  #Use data set X. Then we can use X1 instead of X$X1
  for (i in varlist) {  #For variables in the list {X2, X3, X4}
     plot(X1, get(i), ylab=i) #plot X1 against i-th variable in the list. Here "i"
is the name "X2"/ "X3"/"X4", use get(i) to get the variable X2, X3 or X4.
  }
)
```



We can observe that the two-dimensional scatterplots are all cloud shaped like bivariate normal distributions. Also from the plots, X_1 and X_2 are positively correlated (leftmost plot), X_1 and X_4 are negatively correlated (rightmost plot).

Next, we illustrate the distributions of two linear combinations of X_1, X_2, X_3, X_4. According to the properties of normal random variables, the linear combinations also jointly follow the normal distribution.

Demonstration: Multivariate Normal Distributions (Cont'd)

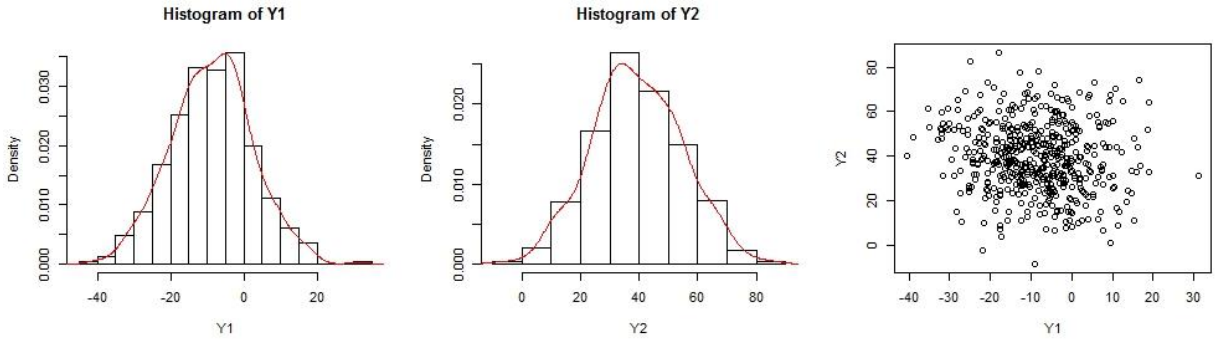
We consider the following two linear combinations:

$$Y_1 = 3 + 2X_1 - 2X_2 + 4X_3 - 4X_4$$

$$Y_2 = -6 - X_1 - X_2 + 6X_3 + 2X_4$$

We plot the univariate histograms and bivariate scatterplot.

```
Y1 <- 3 + 2*X[,1] -2*X[,2] + 4*X[,3] - 4*X[,4] #Y1 as linear combination
Y2 <- -6 - X[,1] - X[,2] + 6*X[,3] + 2*X[,4]
par(mfrow=c(1,3)) #3 plots in one row
hist(Y1,freq=F, nclass=12);lines(density(Y1), col='red') #histogram+density
hist(Y2,freq=F, nclass=12);lines(density(Y2), col='red')
plot(Y1,Y2)        #scatterplot
```



We can see that the two linear combinations look to follow a bivariate normal distribution, as predicted by the properties of normal random variables.

It is easy to find the means of these two linear combinations:

$$E(Y_1) = 3 + 2E(X_1) - 2E(X_2) + 4E(X_3) - 4E(X_4) = 3 + 2(2) - 2E(4) + 4E(6) - 4E(8) = -9$$

$$E(Y_2) = -6 - E(X_1) - E(X_2) + 6E(X_3) + 2E(X_4) = -6 - 2 - 4 + 6(6) + 2(8) = 40$$

We can see these numbers -9 and 40 are the centers of the histograms above.

Multivariate Sample Statistics

The multivariate data is often stored as a data frame in R, and then the summary statistic can be found easily. For the above example, X is the data frame.

```
> summary(X)
       X1                X2               X3               X4
 Min.   :-0.9513   Min.   :-1.334   Min.   : 0.8343   Min.   :-0.6626
 1st Qu.: 1.3700   1st Qu.: 2.757   1st Qu.: 4.6352   1st Qu.: 5.7430
 Median : 2.0854   Median : 4.119   Median : 5.9920   Median : 7.9372
 Mean   : 2.0616   Mean   : 4.067   Mean   : 5.9832   Mean   : 7.8280
 3rd Qu.: 2.6817   3rd Qu.: 5.298   3rd Qu.: 7.2459   3rd Qu.:10.0979
 Max.   : 4.9744   Max.   :10.398   Max.   :13.1665   Max.   :16.3497
```

We can see that the univariate mean, median and quartiles are provided by summary() function.

```
> options(digits=4) #Print only 4 digits in following
> apply(X, 2, sd)    #standard deviation for each column (“2” for columns)
   X1    X2    X3    X4
1.008 1.963 1.963 3.075
> var(X)   #variance here is a 4 by 4 matrix
       X1     X2      X3     X4
X1  1.0162  1.011 -0.5173 -0.912
X2  1.0112  3.853  1.3417 -1.242
X3 -0.5173  1.342  3.8533  1.876
X4 -0.9120 -1.242  1.8764  9.458
> cor(X) #correlation here is also a 4 by 4 matrix
       X1      X2      X3      X4
X1  1.0000  0.5110 -0.2614 -0.2942
X2  0.5110  1.0000  0.3482 -0.2057
X3 -0.2614  0.3482  1.0000  0.3108
X4 -0.2942 -0.2057  0.3108  1.0000
```

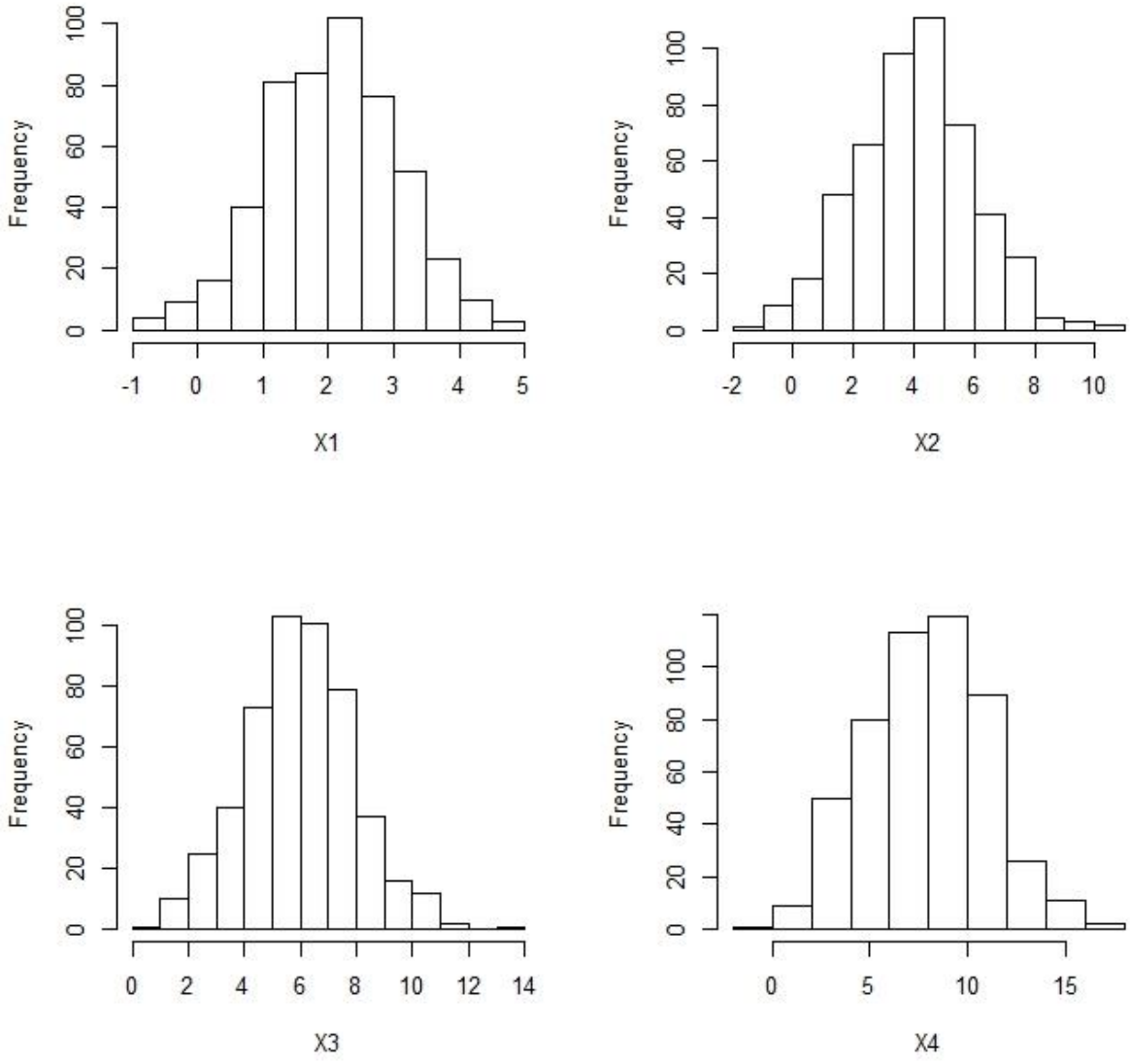
We observe that the sample variance matrix and correlation matrix are close to the true variance matrix $\Sigma = \begin{pmatrix} 1 & 1 & -0.4 & -0.9 \\ 1 & 4 & 1.6 & -1.2 \\ -0.4 & 1.6 & 4 & 1.8 \\ -0.9 & -1.2 & 1.8 & 9 \end{pmatrix}$ and the true correlation matrix

$$\begin{pmatrix} 1 & 0.5 & -0.2 & -0.3 \\ 0.5 & 1 & 0.4 & -0.2 \\ -0.2 & 0.4 & 1 & 0.3 \\ -0.3 & -0.2 & 0.3 & 1 \end{pmatrix}.$$

Demonstration: Displaying Multivariate Data

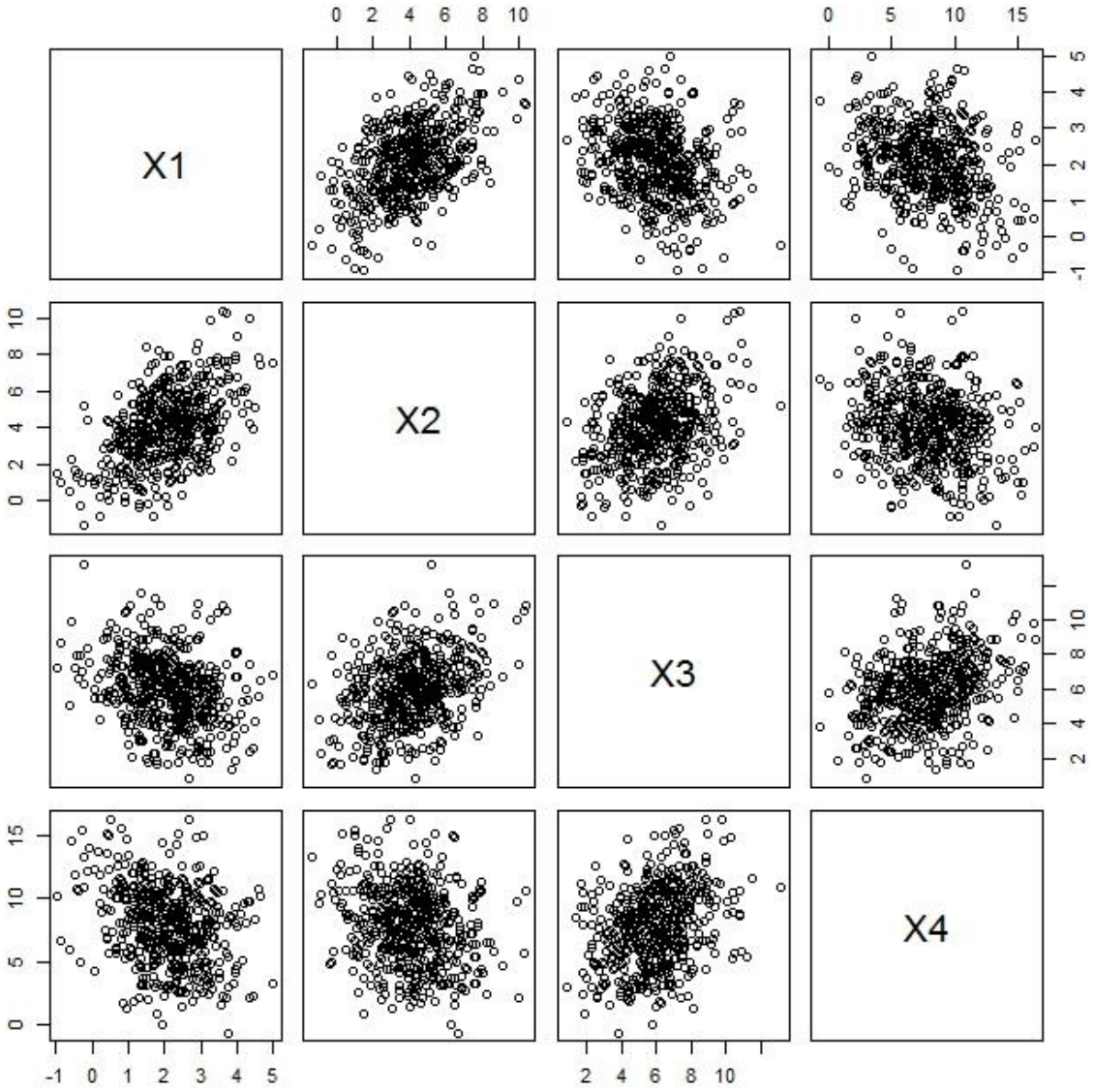
We use the data frame X in the above example to illustrate visual display of multivariate data. We have learned how to display one-dimensional and two-dimensional data previously. We usually first display all the univariate and bivariate marginal sample distributions to get some visualization. The univariate histograms can be drawn as:

```
par(mfrow=c(2,2)) #plots in two rows by two columns
for (i in 1:4) {
        hist(X[,i], xlab=colnames(X)[i], main=NULL) #histogram
}
```



Demonstration: Displaying Multivariate Data (Cont'd)

In R, we can use pairs() function to get all pairwise bivariate scatterplots.
pairs(X)

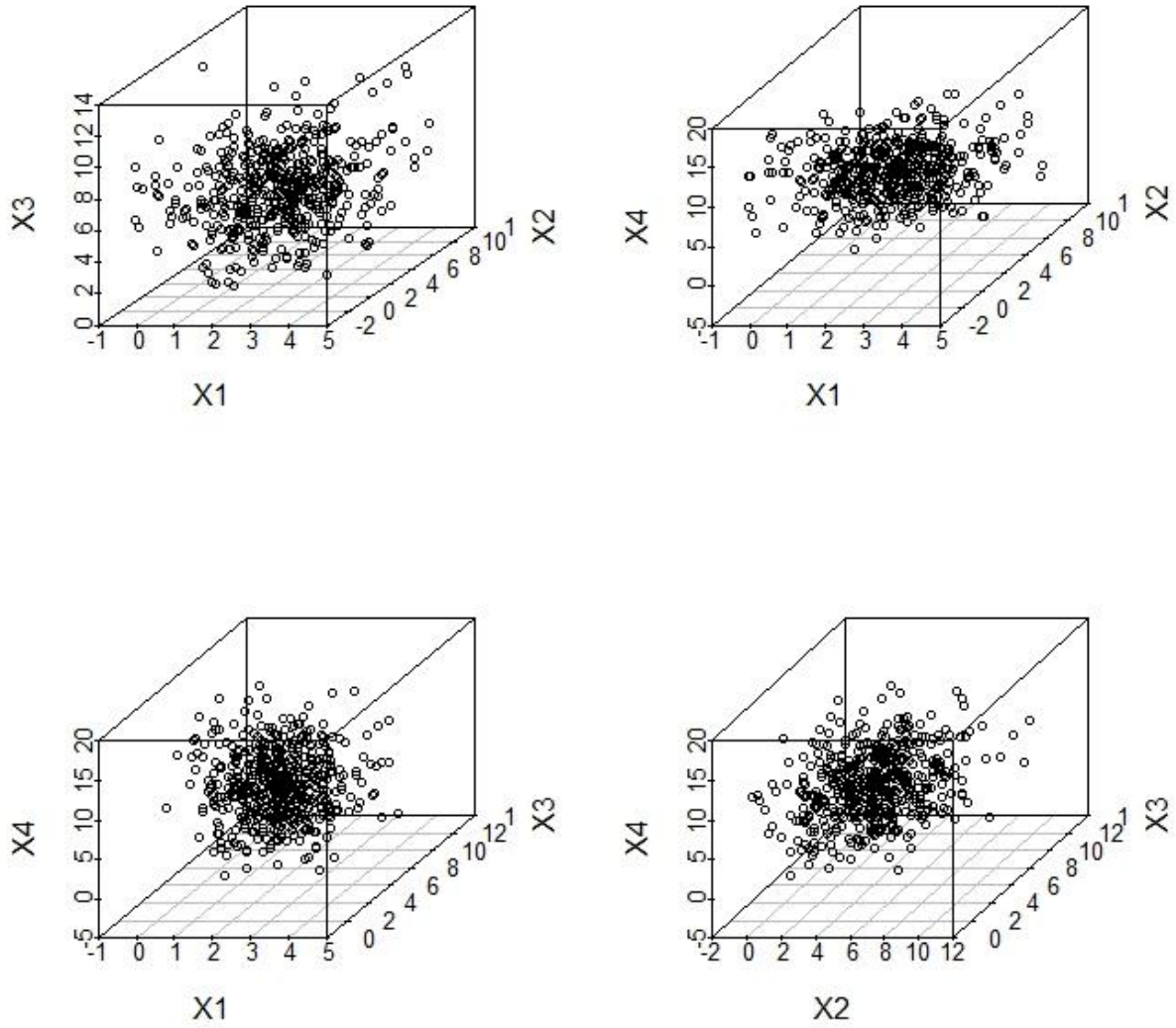


We can observe that X_1 is positively correlated with X_2 and negatively correlated with X_3 and X_4; X_2 is positively correlated with X_3 and negatively correlated with X_4; X_3 is positively correlated with X_4.

Demonstration: Displaying Multivariate Data (Cont'd)

We can also display 3D-scatterplots nowadays in R. We can do this using package "scatterplot3d" (install it using command install.packages). We do this for the above example.

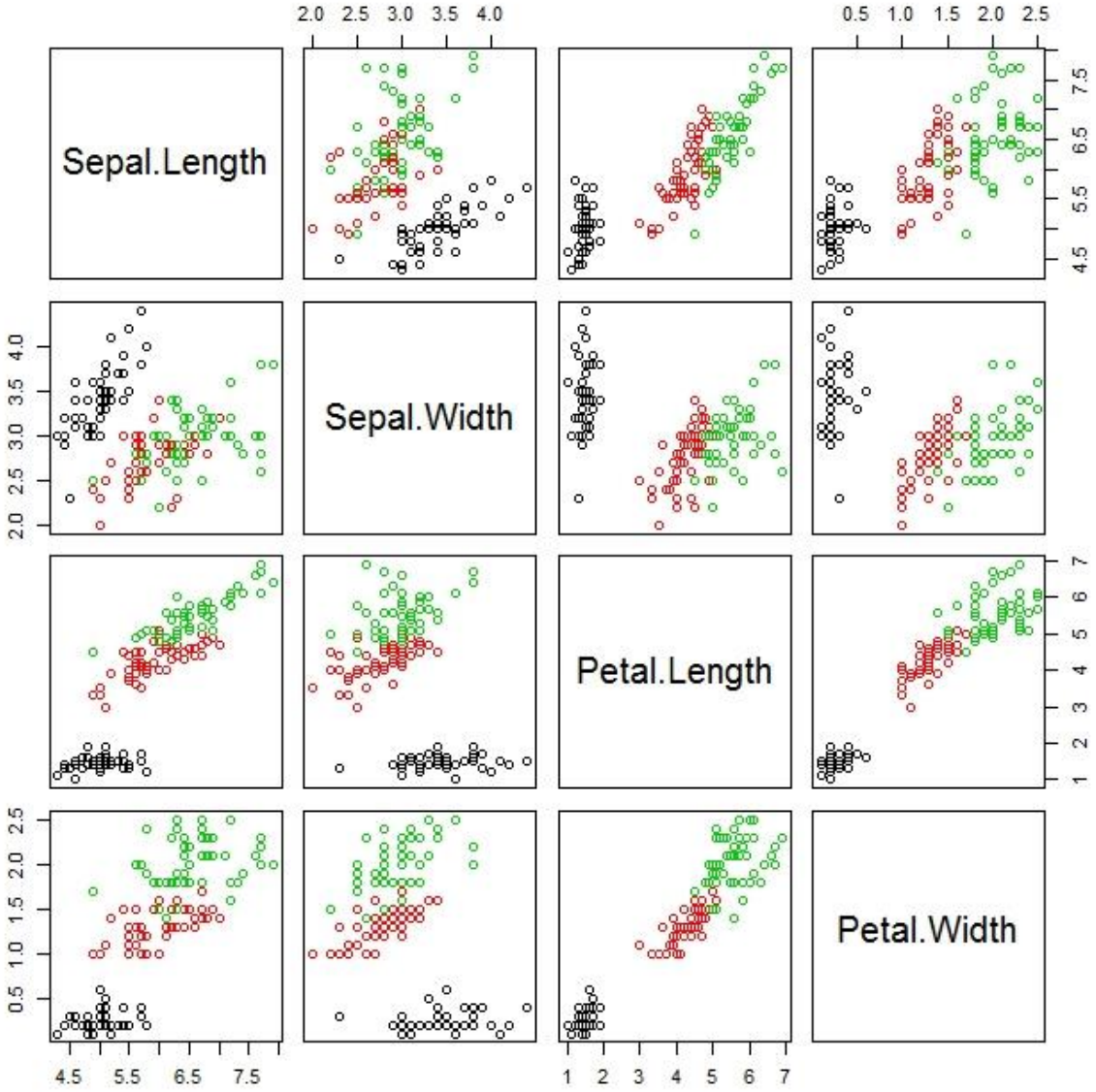
```
require(scatterplot3d) #load the library 'scatterplot3d'
par(mfrow=c(2,2))     #plots in two rows by two columns
for (i in (4:1)) scatterplot3d(X[,-i]) #four 3d-plot, each plot variables
(columns) in X except the i-th column.
```



Demonstration: Iris Data

We can further demonstrate the methods of displaying multivariate data on the iris data set. Recall that the first four (columns) variables are numerical, while the last (column) variable denotes the species of the flowers. We first draw 2D scatterplots for the first four variables, and use different colors to indicate species.

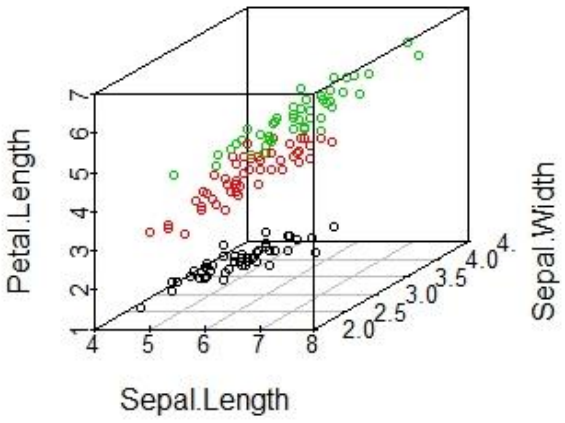
```
pairs(iris[,1:4],col=iris$Species) #pairwise scatterplot, color by species
```

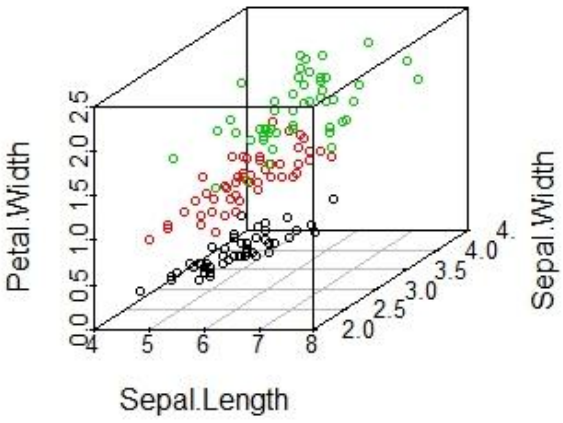


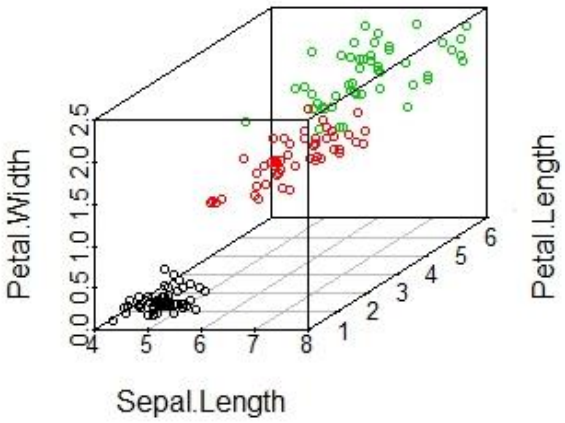
Demonstration: Iris Data (Cont'd)

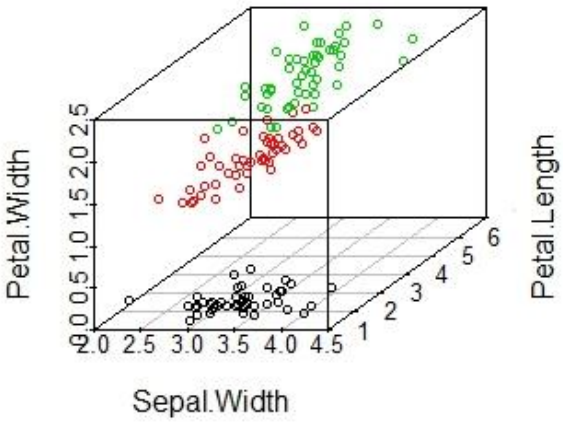
We now draw 3D scatterplots for the first four variables, and use different colors to indicate species.

```
irisNum<-iris[,1:4]   #only take first 4 variables (columns)
par(mfrow=c(2,2))
for (i in (4:1)) scatterplot3d(irisNum[,-i], color=as.numeric(iris$Species))
#3d-scatterplots without i-th variable. Color with species info.
```









The color can also be used as a fourth dimension in the plot. But the 4D scatterplot is not as intuitive as the 3D scatterplot.

Summary

This lesson first introduced multivariate normal distribution. We reviewed useful properties about linear combination of normal random variables. We introduced R functions to generate multivariate normal distribution. We then introduced R commands for summarizing multivariate data, including visual displays in 1-dimensional, 2-dimensional and 3-dimensional.

The next lesson covers principal components analysis, a dimension reduction method for multivariate data.

Lesson 2: Principal Components Analysis (PCA)

Objectives

By the end of this lesson you will have had the opportunity to:

- Determine the **principal components**
- Visually display the first two principal components with **biplot**
- Select principal components to **reduce dimensionality** and carry further data analysis on reduced data set

Overview

In this lesson, we will focus further on the principal component analysis (PCA). We will start the definition of principal components. We then introduce the R routine to carry out the PCA. The analysis will be illustrated with two examples: The randomly generated multivariate normal data of last lesson and the Golub data from the textbook.

Principal Components Analysis

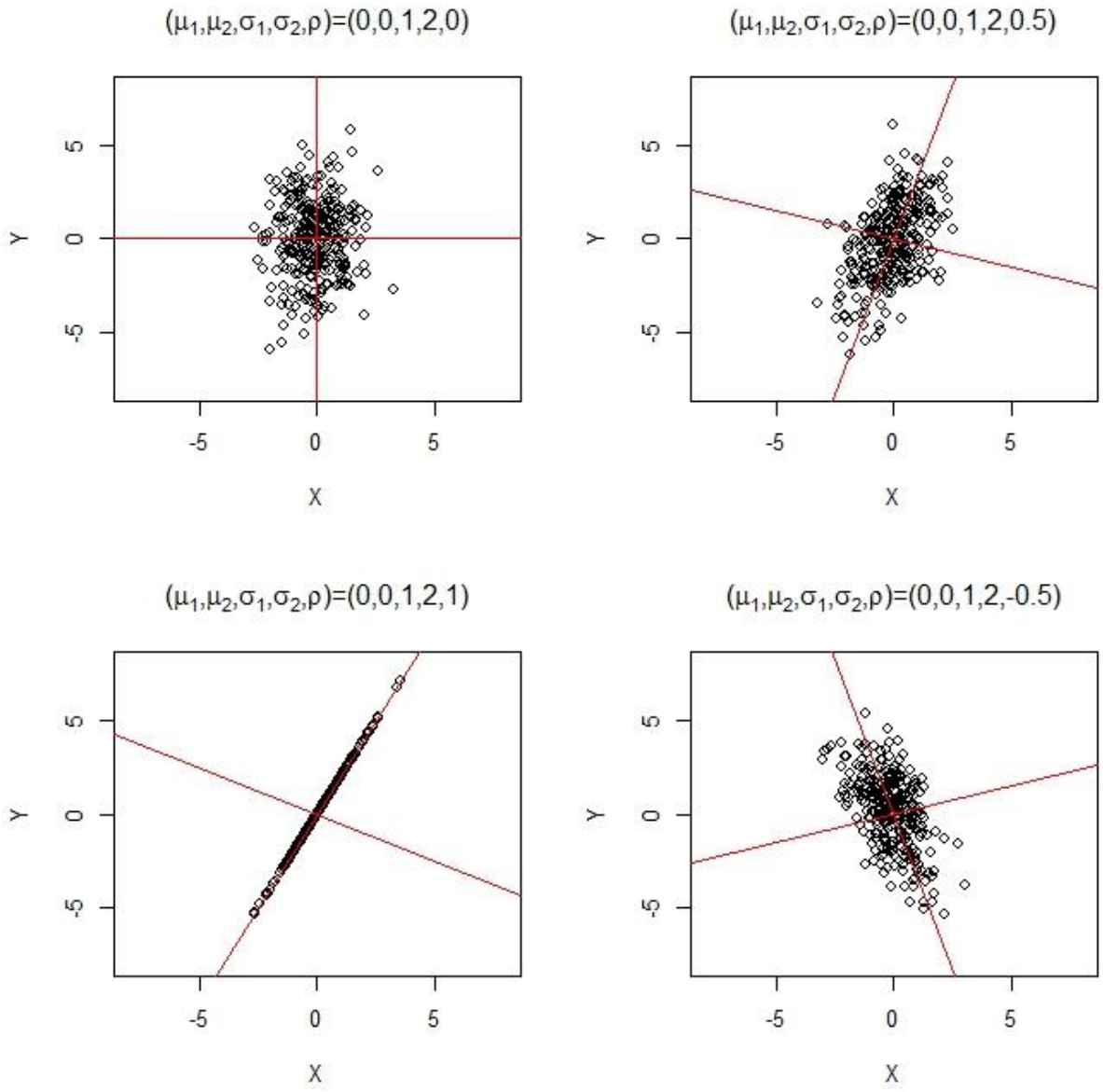
Principal component analysis (PCA) is one kind of dimension reduction technique where we try to represent the data with fewer variables. The primary goal of PCA is to describe the variability in a multivariate data of p correlated variables by a smaller set of derived variables that are uncorrelated. The derived variables, called principal components, are linear combinations of the original variables, and are usually listed in the order of their respective importance. In other words, the first principal component (PC1) explains the largest amount of variation, the second principal component (PC2) is uncorrelated to PC1, and explains the second largest amount of variation, and so on. The aim is to discard subsequent principal components after a large percentage of the variation has been explained by the first k principal components, where k < p.

It is very instructive to look at the multivariate normal distribution which are characterized by the mean vector $\vec{\mu} = (\mu_1, ..., \mu_p)$ and the variance matrix Σ. The **first principal component** (PC1) is a normalized linear combination $Y_1 = a_{11}X_1 + ... + a_{p1}X_p$ with largest variance. By *normalized*, we mean that $a_{11}^2 + ... + a_{p1}^2 = 1$. The weights $(a_{11}, ..., a_{p1})$ are called the **loadings** of PC1. Then the second principal component (PC2) $Y_2 = a_{12}X_1 + ... + a_{p2}X_p$ is the normalized linear combination with largest variance, and also uncorrelated to PC1 $Y_1 = a_{11}X_1 + ... + a_{p1}X_p$. The $(a_{12}, ..., a_{p2})$ are the loadings of PC2, and so on. Therefore, these principal components can be found by the eigenvalue decomposition of the variance matrix Σ. Mathematically, the PCs are up to a sign change. If all loadings' signs are changed (times a "-1"), the direction remains the same: $-Y_1$ and Y_1 have the same variance, and are considered the same PC1.

We will first review the bivariate scatterplots of the bivariate normal distribution to get a visualized geometric interpretation of the principal components.

Principal Components in Plots of Bivariate Normal Data

Here we have the plots of bivariate normal data with various parameters from last lesson.



The long axis of the oval shape is the PC1, the other axis is PC2. Notice that for bigger correlation values $|\rho|$, the PC1 accounts for a bigger proportion of total variation. For $\rho = 1$, the data can in fact be represented totally along PC1, and the dimensionality is reduced to one from two. In high dimensional data, we hope that a few axis will account for most of the total variation. Then we can approximate

the data with the principal components along those few axis.

Finding the Principal Components

Knowing the variance matrix Σ, the principal components can be found by the eigenvalue decomposition of Σ. In practice, we do not know the true variance matrix Σ, but we have the sample variance matrix. The principal component analysis (PCA) is calculated using the eigenvalue decomposition of the *sample variance matrix*, and is programmed as the prcomp() function in R.

In R, there are there functions such as princomp() that also finds the PCA. prcomp() and princomp() have similar outputs. They use different numerical algorithms in implementation, but that really does not matter for us users.

In this course we will be using prcomp(). The other function princomp() has an annoying feature of not printing out any loading coefficients < 0.1.

Next, we will demonstrate finding the principle components on the example of four-dimensional multivariate normal data in last lesson.

Demonstration: PCA on Multivariate Normal Data

Let us look at R output for the following session:

```
> Sigma <- matrix(c(1,1,-.4,-.9,1,4,1.6,-1.2,-.4,1.6,4, 1.8,-.9,-
1.2,1.8,9),ncol=4)
> X <- rmvnorm(500,mean=c(2,4,6,8),sigma=Sigma)
> X<-data.frame(X); colnames(X)<-paste("X",(1:4),sep="")
> options(digits=4) #Print only 4 digits in following
> eigen(Sigma)       #True PC: eigenvalues and eigenvetors
$values
[1] 9.8017 5.6277 2.1134 0.4571
$vectors
       [,1]    [,2]    [,3]     [,4]
[1,]  0.1251  0.1152  0.4450  0.87924
[2,]  0.1449  0.7483  0.5220 -0.38286
[3,] -0.2621  0.6513 -0.6536  0.28270
[4,] -0.9459 -0.0506  0.3199 -0.02072
> pc.X<- prcomp(X) #Sample PC
> pc.X      #display the PC fit (sdev and loadings)
Standard deviations:
[1] 3.1394 2.4906 1.3889 0.6871
Rotation:
      PC1      PC2     PC3      PC4
X1  0.1254  0.10897 -0.4268 -0.88896
X2  0.1223  0.75032 -0.5363  0.36669
X3 -0.3056  0.64465  0.6450 -0.27376
X4 -0.9359 -0.09785 -0.3379  0.01822
```

We can see that the loadings calculated by `prcomp(X)` from the sample variance matrix is very close to the eigenvectors of the true variance matrix. The PC1 is $0.125X_1 + 0.122X_2 - 0.306X_3 - 0.936X_4$, in contrast to the true PC1 $0.125X_1 + 0.145X_2 - 0.262X_3 - 0.946X_4$.

The eigenvalues of Σ reflect the true variances for the principal components. We can see that they are very close to the estimated variances (squares of standard deviations).

```
> pc.X$sdev^2   #square to get variances of each PC

[1] 9.8560 6.2031 1.9290 0.4722
```

Demonstration: PCA on Multivariate Normal Data (Cont'd)

In fact, we can see that the PCs estimated by prcomp(X) are exactly those PCs from the eigenvalue analysis of the sample variance matrix eigen(cov(X)).

```
> pc.X        #display the PC fit
Standard deviations:
[1] 3.1394 2.4906 1.3889 0.6871
Rotation:
       PC1      PC2     PC3      PC4
X1  0.1254  0.10897 -0.4268 -0.88896
X2  0.1223  0.75032 -0.5363  0.36669
X3 -0.3056  0.64465  0.6450 -0.27376
X4 -0.9359 -0.09785 -0.3379  0.01822
> pc.X$sdev^2
[1] 9.8560 6.2031 1.9290 0.4722
```

```
> eigen(cov(X)) #eigenvalues and eigenvectors of sample variance matrix
$values
[1] 9.8560 6.2031 1.9290 0.4722
$vectors
        [,1]    [,2]    [,3]     [,4]
[1,] -0.1254  0.10897  0.4268  0.88896
[2,] -0.1223  0.75032  0.5363 -0.36669
[3,]  0.3056  0.64465 -0.6450  0.27376
[4,]  0.9359 -0.09785  0.3379 -0.01822
```

Notice that the eigenvalues of the sample covariance matrix cov(X) are exactly the squares of standard deviations of the PCA. And the eigenvectors are exactly (up to a sign change) those of the loadings in PCA. For PCA, changing all loadings by a sign does not change the principal component. (0.1254, 0.1223, -0.3056, -0.9359) is the same principal component as (-0.1254, -0.1223, 0.3056, 0.9359)

Demonstration: PCA on Multivariate Normal Data (Cont'd)

We can find the proportion of variances that are accounted for by different principal components, by applying the summary() function on the R object fitted from prcomp().

```
> summary(pc.X) #summary of the PC fit, instead display PC fit directly
Importance of components:
                          PC1   PC2   PC3    PC4
Standard deviation      3.139 2.491 1.389 0.6871
Proportion of Variance  0.534 0.336 0.104 0.0256
Cumulative Proportion   0.534 0.870 0.974 1.0000
```

The first row lists the sample standard deviation for each principal component. Squares of those are the variances and are converted into proportions of total variances listed in the second row.

```
> pc.X$sdev^2/sum(pc.X$sdev^2) #divide by sum to get proportions
[1] 0.53390 0.33603 0.10449 0.02558
```

We can see that our calculated proportions are the same numbers in the second row of the summary(pc.X) output.

The last row in summary(pc.X) output lists the proportions of total variance accounted for by the first k principal components cumulatively. For example, the first two principal components account for 0.534+0.336 =0.870 proportion of total variance.

We demonstrate the bootstrap inference next.

Demonstration: PCA on Multivariate Normal Data (Cont'd)

By resampling data, we can obtain confidence intervals for quantities in the PCA fit. Here we resample nboot=1000 times to calculate the bootstrap confidence intervals for the standard deviations of the principal components.

```
data <- X; p <- ncol(data); n <- nrow(data) ; nboot<-1000 #define quantities
sdevs <- array(dim=c(nboot,p)) #empty nboot by p matrix to save sdev for
each PC in bootstrapped samples
for (i in 1:nboot) {
        dat.star <- data[sample(1:n,replace=TRUE),] #bootstrap data
        sdevs[i,] <- prcomp(dat.star)$sdev    #sdev for PCs in bootrap data
}
print(names(quantile(sdevs[,1], c(0.025,0.975)))) #display "2.5%"  "97.5%"
for (j in 1:p) cat(j, as.numeric(quantile(sdevs[,j], c(0.025,0.975))),"\n") #for
j-th PC, print "j" then the 95% CI limits in one line. "\n" starts a new line
```

If we run the above R script, we get the 95% confidence intervals for the standard deviations of every principal components.

```
[1] "2.5%"  "97.5%"
1 2.918 3.383
2 2.352 2.636
3 1.296 1.467
4 0.6401 0.7293
```

We can see that, in this case, the four 95% CIs all contain the true values.

```
> sqrt(eigen(Sigma)$values) #True PC sdev
[1] 3.1308 2.3723 1.4538 0.6761
```

Run the above R script, we get the 95% confidence intervals for the standard deviations of every principal component.

Demonstration: PCA on Multivariate Normal Data (Cont'd)

Recall when we plot the bivariate normal data in the last lesson, increasing the standard deviation value in one variable will stretch the scatterplot along the direction of the variable. For an easier interpretation of the principal components, often the variables are all scaled to have standard deviation of one, before applying the PCA. Mathematically, this is equivalent to do the eigenvalue analysis on the sample correlation matrix (instead of on the sample covariance matrix before).

```
> prcomp(X, scale=TRUE) #Do PCA on X with scaled variables.

Standard deviations:
[1] 1.2949 1.2096 0.7942 0.4790
Rotation:
       PC1      PC2      PC3     PC4
X1  0.64679 -0.09616  0.61890  0.4352
X2  0.58107  0.47626 -0.07198 -0.6560
X3 -0.02042  0.77579 -0.26162  0.5738
X4 -0.49357  0.40259  0.73711 -0.2258
> eigen(cor(X)) #eigenvalues and eigenvectors for sample correlation matrix
$values
[1] 1.6767 1.4631 0.6307 0.2294
$vectors
        [,1]     [,2]     [,3]    [,4]
[1,] -0.64679 -0.09616  0.61890 -0.4352
[2,] -0.58107  0.47626 -0.07198  0.6560
[3,]  0.02042  0.77579 -0.26162 -0.5738
[4,]  0.49357  0.40259  0.73711  0.2258
```

We can see the PCA for scaled variables can be done in R by setting the option "scale=TRUE" inside the prcomp() function. And it gives the same values as the eigenvalue analysis on the sample correlation matrix.

Demonstration: PCA on Golub data

Let us now look at another example. We apply the PCA on the Golub (1999) data as in the textbook.

```
> pr.golub <- prcomp(golub, scale=TRUE) #save (scaled) PCA fit on golub
> summary(pr.golub)    #summary of the PCA fit
Importance of components:
                            PC1    PC2     PC3    PC4     PC5    PC6     PC7    PC8
Standard deviation        5.044 1.4407 1.1173 1.0351 0.8582 0.7440 0.7210 0.6923
Proportion of Variance **0.669 0.0546** 0.0328 0.0282 0.0194 0.0146 0.0137 0.0126
Cumulative Proportion 0.669 0.7240 0.7569 0.7851 0.8045 0.8190 0.8327 0.8453
                             PC9   PC10    PC11     PC12    PC13     PC14    PC15
Standard deviation        0.6382 0.6363 0.56700 0.55263 0.53868 0.52011 0.49568
Proportion of Variance 0.0107 0.0106 0.00846 0.00804 0.00764 0.00712 0.00647
Cumulative Proportion 0.8561 0.8667 0.87518 0.88321 0.89085 0.89797 **0.90443**
……
```

We can see that the first principal component PC1 accounts for almost 67% of total variance, the next PC2 accounts for about 5%, and the other principal components account less and less. The first 15 principal components (out of 38 components) accounts for 90% of total variance.

Similar to textbook example in Applied Statistics for Bioinformatics using R, we conduct a bootstrap procedure on the Golub data. We print out the bootstrap CIs for the first five components' standard deviations.

```
data <- golub; p <- ncol(data); n <- nrow(data) ; nboot<-1000
sdevs <- array(dim=c(nboot,p)) #matrix to save resampled sdev for p PCs
for (i in 1:nboot) {
        dat.star <- data[sample(1:n,replace=TRUE),] #resample rows
        sdevs[i,] <- prcomp(dat.star)$sdev  #sdev for PCA on resampled data
}
print(names(quantile(sdevs[,1],c(0.025,0.975)))) #print out "2.5%"  "97.5%"
for (j in 1:5) cat(j, as.numeric(quantile(sdevs[,j], c(0.025,0.975))), "\n")
##print out j, boostrap CI for j-th component, then new line ("\n")
```

Run it to get

```
[1] "2.5%"  "97.5%"
1 4.901 5.186
2 1.394 1.494
3 1.071 1.18
4 0.997 1.066
```

5 0.829 0.8928

We can observe that the CIs are tight. Thus, the estimated proportion of variances accounted by the PCs are accurate.

Demonstration: PCA on Golub data (Cont'd)

From the PCA fit on last page, the first two principal components account for 72.4% of total variance. Concentrating on these two principal components results in a dimension reduction from 38 to 2. We print out the loadings for the first two principal components.

```
> print(t(pr.golub$rotation[,1:2]), digits=3) #print out the loadings for the first two PCs, transpose so that the loadings are shown in rows instead of columns, display only 3 digits for each entry
        [,1]    [,2]   [,3]  [,4]  [,5]   [,6]   [,7]     [,8]   [,9]  [,10]
PC1 0.172  0.1691 0.1650 0.173 0.166 0.1669 0.1686 0.160244 0.1649 0.1688
PC2 0.104 -0.0369 0.0691 0.101 0.171 0.0283 0.0324 0.000506 0.0936 0.0235
        [,11]   [,12] [,13]  [,14] [,15] [,16]  [,17]  [,18] [,19] [,20] [,21]
PC1 0.1654  0.1694 0.163 0.1661 0.165 0.172 0.1559 0.1600 0.168 0.149 0.127
PC2 0.0754 -0.0894 0.233 0.0779 0.238 0.184 0.0782 0.0416 0.115 0.247 0.202
          [,22]  [,23] [,24]   [,25] [,26]  [,27]  [,28]  [,29]  [,30]  [,31]
PC1  0.1621 0.1644 0.165  0.1659 0.169 0.1540  0.169  0.154  0.152  0.169
PC2 -0.0141 0.0379 0.211 -0.0445 0.122 0.0214 -0.189 -0.175 -0.244 -0.165
        [,32]   [,33]    [,34]   [,35]   [,36]    [,37]   [,38]
PC1  0.168  0.145  0.168  0.164  0.151  0.148  0.152
PC2 -0.150 -0.344 -0.158 -0.131 -0.278 -0.345 -0.223
```

We can see that the loadings for PC1 are all positive and have very similar size (all between 0.13 and 0.17). So PC1 is essentially the average of all variables. The loadings of PC2 have an interesting pattern: Almost all of the first 27 loadings are positive and all of the last 11 loadings are negative.

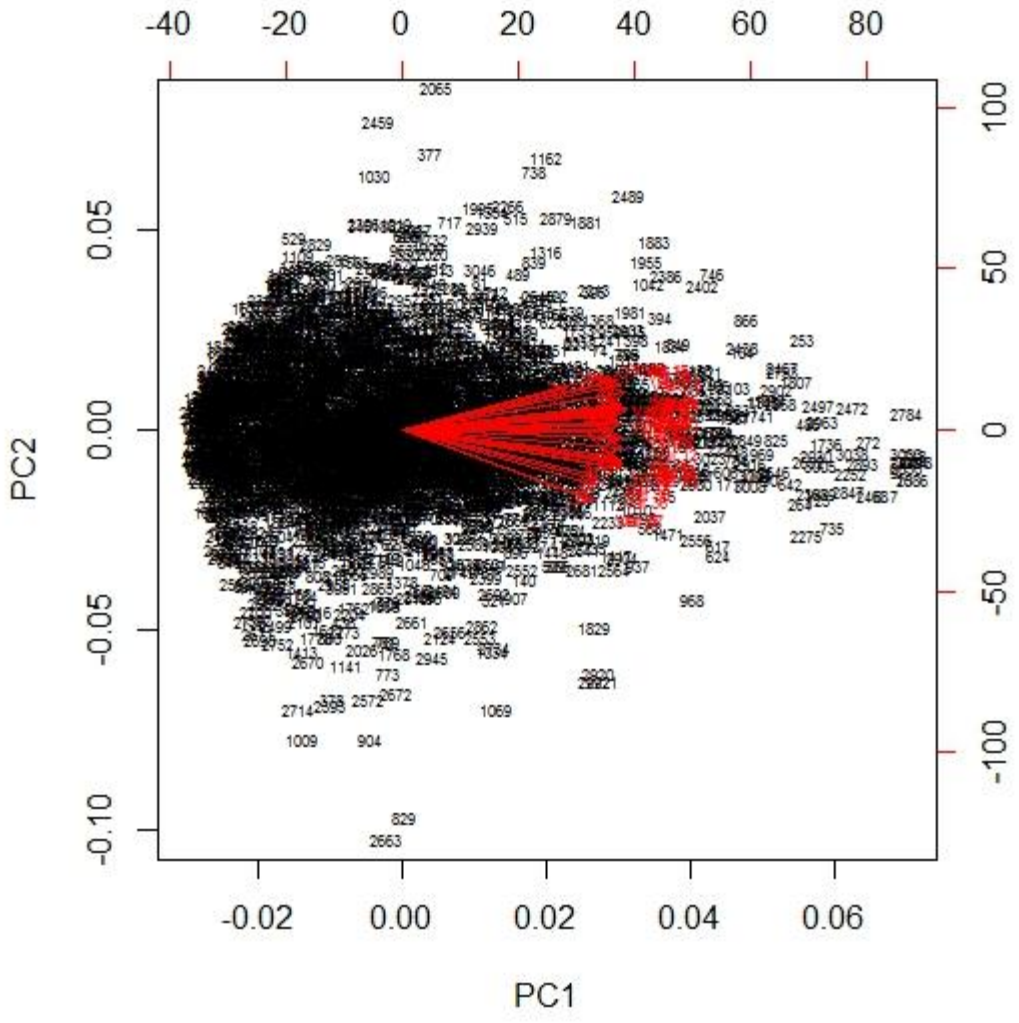
Recall that the first 27 are ALL patients and the last 11 are AML patients. So PC2 mostly contrasts the ALL patients against the AML patients. Therefore, besides the overall average in PC1, the contrasting of ALL and AML patients explains the largest amount of variance in data. This discovery is in line with findings in Golub et al. (1999).

Next, we use a common visualization tool 'biplot' to plot the genes on the first two principal components.

Demonstration: PCA on Golub data (Cont'd)

A common visualization tool is the 'biplot' that plots the data on the first two principal components. It also displays the loadings of the first components with red arrows.

```
> biplot(pr.golub, xlim=c(-0.03,0.07), ylim=c(-0.1,0.08), cex=0.5) #biplot display of the PCA fit, set ranges of x and y, use smaller font (cex=0.5)
```



We see the red arrows have about the same horizontal lengths. This reflects the fact that PC1 is essentially the average of the patients, as discussed in last page. The main purpose of the biplot is to show patterns on the first two PCs. For example, gene 2784 has a big PC1 value, indicating that it has large average expression values among the 38 patients. The PC2 reflects the distinction of ALL versus AML patients. So the genes with large (positive or negative) PC2 values, such as 2065

and 2663, have the potential to distinguish the two patient groups.

Demonstration: PCA on Golub data (Cont'd)

We can find the ten genes with largest positive PC2 values and the ten genes with largest negative PC2 values (which is stored in pr.golub$x) as the following.

```
> o <- order(pr.golub$x[,2]) #order the PC2 values
> golub.gnames[o[1:10],2]   #first 10 ordered gene names (stored in 2nd column)
 [1] "INTERLEUKIN-8 PRECURSOR"
 [2] "Interleukin 8 (IL8) gene"
 [3] "CST3 Cystatin C (amyloid angiopathy and cerebral hemorrhage)"
 [4] "DF D component of complement (adipsin)"
 [5] "LGALS3 Lectin, galactoside-binding, soluble, 3 (galectin 3) (NOTE:
redefinition of symbol)"
 [6] "Azurocidin gene"
 [7] "GRO2 GRO2 oncogene"
 [8] "MACROPHAGE INFLAMMATORY PROTEIN 1-ALPHA PRECURSOR"
 [9] "LYZ Lysozyme"
[10] "CYSTATIN A"
> golub.gnames[o[3042:3051],2] #last 10 ordered gene names
 [1] "Cytoplasmic dynein light chain 1 (hdlc1) mRNA"
 [2] "RETINOBLASTOMA BINDING PROTEIN P48"
 [3] "Adenosine triphosphatase, calcium"
 [4] "C-myb gene extracted from Human (c-myb) gene, complete primary cds, and
five complete alternatively spliced cds"
 [5] "IGB Immunoglobulin-associated beta (B29)"
 [6] "Terminal transferase mRNA"
 [7] "MB-1 gene"
 [8] "GB DEF = (lambda) DNA for immunoglobin light chain"
 [9] "CD24 signal transducer mRNA and 3' region"
[10] "TCL1 gene (T cell leukemia) extracted from H.sapiens mRNA for Tcell
leukemia/lymphoma 1"
```

Many of these genes are related to leukemia (Golub, et al., 1999).

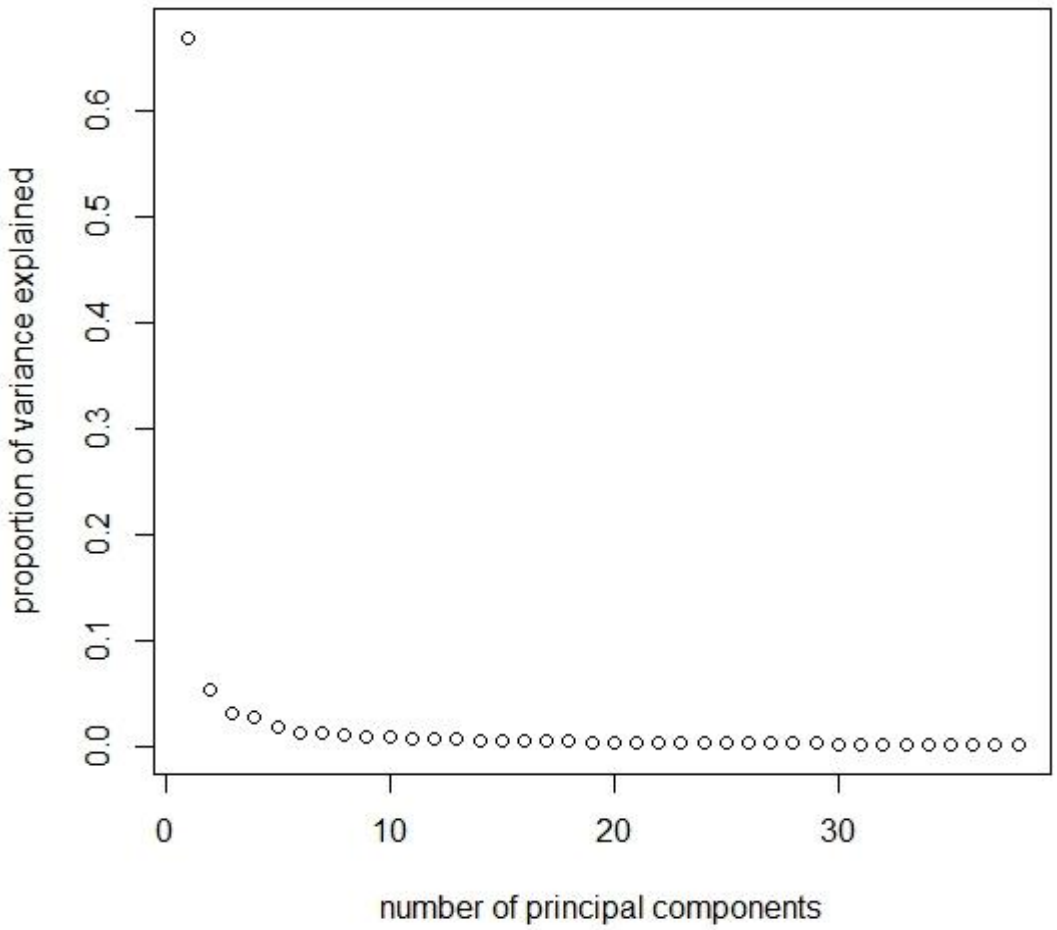
Number of the Principal Components to Use

The main purpose of the PCA is to reduce the dimension by using first k principal components (reduce dimension from n to k). But how many k should we choose? Since PCA is an unsupervised learning method, there is no well-accepted objective ways to choose k. However, there are reasonable common approaches. One approach is to first choose a threshold, say 90%, then choose the first k components that explain at least the proportion of total variance exceeding the threshold.

Another approach is to plot the amount of variance explained by each component and observe the pattern, similar to the selection of the number of clusters in the previous module. We demonstrate this on the Golub data example.

Number of the Principal Components to Use (Golub data)

```
> PropVar<-summary(pr.golub)$importance[2,] #save the proportion of variance of each PC (2nd row). 1st row has sdev, 3rd row cumulative proportions
> plot(1:length(PropVar), PropVar, xlab='number of principal components', ylab='proportion of variance explained') #Plot PropVar against index: 1,2,…
```



It appears that the first components explain most of the variance, and there is a sharp drop off afterwards. The curve flattens after k=5. So a choice between 1 and 5 principal components seems reasonable.

Purpose of the PCA

As we explained in this lesson, the main purpose of the PCA is to reduce the data into a lower dimension. In the lower dimension, it is easier to analyze and interpret the data. Other analysis, such as the clustering analysis can be applied to the reduced data on the lower dimension. For example, we can do cluster analysis for the genes on the first two principle components of the Golub data (See Example 4 on page 140 of Applied Statistics for Bioinformatics using R).

As other unsupervised learning methods, the PCA is mainly used for exploration. That is, we try to discover interesting patterns, instead of doing rigorous statistical inferences like hypothesis testing. It generally serves as a first step in data analysis. Other procedures are applied after the dimension reduction.

Summary

In this lesson we learned how to do the principal component analysis. You should know how to carry out the PCA in R. You should know how to reduce the dimensionality by focusing on the first k principal components. We also learned some methods of selecting the number of components to use, and to do some analysis on the reduced data of first k components.

Module Summary

This module covered multivariate data analysis. You should know basic methods of displaying and summarizing multivariate data. You should know how to carry out the principal component analysis (PCA) in R. You should know the basic visualization for the PCA, and carry out further analysis on PCA reduced data set. You should know how to choose the number of components.

The next module will cover some classification methods. Those are supervised learning methods, in contrast to the unsupervised learning method such as PCA and clustering analysis.