

# Multivariate Analysis Lecture 11: Applications of PCA

Zhaoxia Yu  
Professor, Department of Statistics

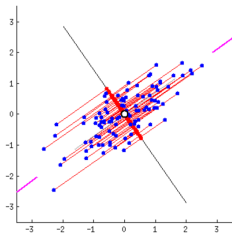
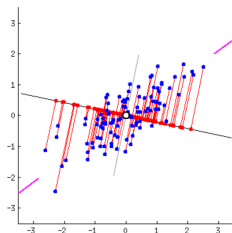
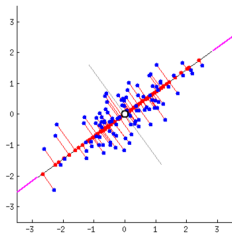
2023-05-09

## Section 1

### Review of PCA

# PCA: Different Directions Have Different Variances

- PCA projects the original data onto a lower-dimensional space using linear combinations of the original features.
- Click the link to see the animated version!



# The Spectral Decomposition of A Covariance Matrix

- Let  $\Sigma_{p \times p}$  be the covariance matrix of a random vector  $\mathbf{X} \in \mathbb{R}$ .
- A Covariance matrix is a positive definite or positive semi-definite.
- Spectral decomposition:

$$\Sigma = \Gamma \Lambda \Gamma^T$$

where  $\Lambda$  is the diagonal matrix of the eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \lambda_p \geq 0$$

- The Eigenvectors are the columns of  $\Gamma = (\gamma_1, \cdots, \gamma_p)$ , where  $\gamma_i$  is the  $i$ th eigenvector.

# First PC

- Let  $Y_1 = a^T \mathbf{X}$  denote the first principal component, which is defined as the linear combination reaches the maximum variance subject to  $\|a\| = 1$ . Mathematically, we are looking for  $a$  s.t.

$$a = \arg \max_{a^T a = 1} a^T \Sigma a$$

- First Principal Component.** Among all the linear combinations of  $\mathbf{X}$ , the one with the maximum variance is  $Y_1 = \gamma_1^T \mathbf{X}$  and the corresponding variance is  $\lambda_1$ .

## 2nd PC

- **Second Principal Component.** Mathematically, we are looking for  $a$  s.t.

$$a = \arg \max_{a^T a=1, a^T \gamma_1=0} a^T \Sigma a$$

- The second PC is

$$Y_2 = \gamma_2^T \mathbf{X}$$

## ith PC

- For the  $i$ th principal component, we are looking for a linear combination in terms of  $a^T \mathbf{X}$  such as

$$a = \arg \max_{a^T a=1, a^T \gamma_1=0, \dots, a^T \gamma_{i-1}=0} a^T \Sigma a$$

- The  $i$ th principal component is

$$Y_i = \gamma_i^T \mathbf{X}$$

# Example

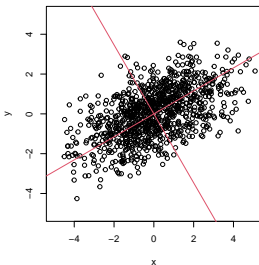
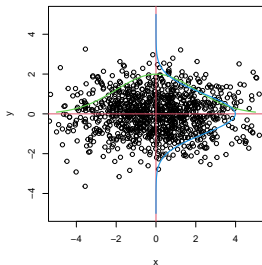
```
n=1000
Sigma1=diag(c(4,1), 2, 2)
Sigma2=diag(c(1,4), 2, 2)
theta=pi/6
R1=matrix(c(cos(theta), sin(theta), -sin(theta), cos(theta)), 2,2)
theta=pi/4+pi/2
R2=matrix(c(cos(theta), sin(theta), -sin(theta), cos(theta)), 2,2)
Sigma3=R1%*%Sigma1%*%t(R1)
Sigma4=R2%*%Sigma1%*%t(R2)
set.seed(1)
X1=data.frame(mvrnorm(n, rep(0,2), Sigma1)); names(X1)=c("x","y")
X2=data.frame(mvrnorm(n, rep(0,2), Sigma2)); names(X2)=c("x","y")
X3=data.frame(mvrnorm(n, rep(0,2), Sigma3)); names(X3)=c("x","y")
X4=data.frame(mvrnorm(n, rep(0,2), Sigma4)); names(X4)=c("x","y")
Sigma3
```

```
##           [,1]      [,2]
## [1,] 3.250000 1.299038
## [2,] 1.299038 1.750000
```



# Simulated Data

```
par(mfrow=c(1,2),pty="s")  
plot(X1, xlim=c(-5,5), ylim=c(-5,5));  
abline(0,0, col=2); abline(v=0, col=2)  
lines(seq(-5,5,0.1), 10*dnorm(seq(-5,5,0.1), 0, 2), col=3, lwd=2)  
lines(10*dnorm(seq(-5,5,0.1), 0, 1), seq(-5,5,0.1), col=4, lwd=2)  
plot(X3, xlim=c(-5,5), ylim=c(-5,5));  
abline(0,1/sqrt(3), col=2); abline(0, -sqrt(3), col=2)
```



## Example: 1st PC and 2nd PC

```
gamma1=eigen(Sigma3)$vectors[,1]  
gamma1
```

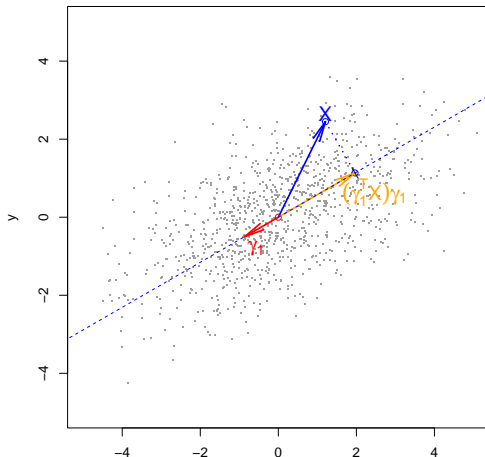
```
## [1] -0.8660254 -0.5000000
```

```
gamma2=eigen(Sigma3)$vectors[,2]  
gamma2
```

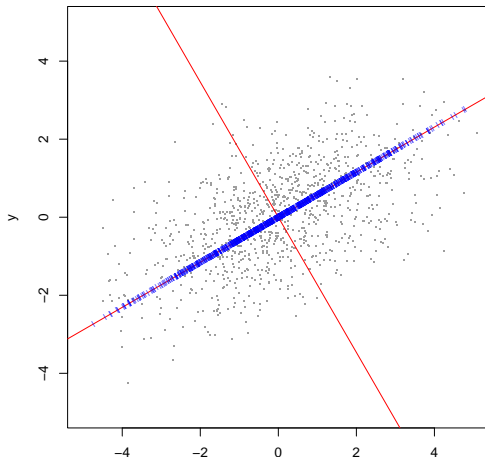
```
## [1] 0.5000000 -0.8660254
```

- 1st PC:  $-0.8660254X_1 - 0.5X_2$
- 2nd PC:  $0.5X_1 - 0.8660254X_2$

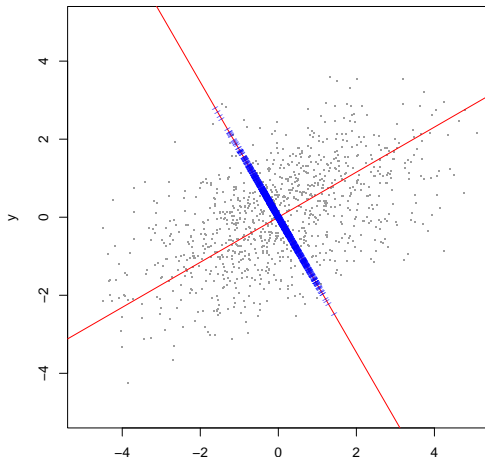
## Example: Project An Observation to 1st PC



## Example: Project All Observations to 1st PC



## Example: Project All Observations to 2nd PC



## Section 2

# Variance Explained

# Dimensionality Reduction Using PCA

- PCA aims to reduce the dimensionality of a dataset while preserving as much variance as possible.
- Dimensionality reduction using PCA can
  - ease visualization and analysis
  - help identify underlying patterns or structure in the data.
  - improve the performance of machine learning algorithms by reducing noise and collinearity.
- PCA is most effective when the majority of the variance can be captured by a small number of principal components.

# Total Variance

- There are at least two justifications to use  $\sum_{i=1}^p \lambda_i$  as the total variance:
  - ① One way to quantify the total variance in  $\mathbf{X}$  is the trace of the covariance matrix  $\mathbf{\Sigma}$ 
    - Recall that  $tr(\mathbf{\Sigma}) = \sum_{i=1}^n \lambda_i$
  - ② The variance of  $i$ th PC is  $Var(Y_i) = Var(\gamma_i^T \mathbf{X}) = \lambda_i$ . Thus, the total variance of PCs is

$$\sum_{i=1}^p Var(Y_i) = \sum_{i=1}^p \lambda_i$$



# Variance Explained

- The proportion of the variance explained by  $i$ th PC explains is

$$\lambda_i / \sum_{i=1}^p \lambda_i$$

- Cumulative Explained Variance is the proportion of the total variance explained by the first  $k$  PCs is

$$\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^p \lambda_i}$$

# Variance Explained: Example

```
lambda=eigen(Sigma3)$values  
# Prop of Variance explained by 1st PC  
lambda[1]/sum(lambda)
```

```
## [1] 0.8
```

```
# Prop of Cumulative Variance explained by two PCs  
sum(lambda[1:2])/sum(lambda)
```

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

- In this example, 1st PC explains 80% of variance, 2nd PC explains 20% of variance.

## Section 3

### Apply PCA to Data

# Estimate $\Sigma$ .

- We have discussed how to find PCs for a random vector  $\mathbf{X}_{p \times 1} \sim (0, \Sigma)$ , where  $\Sigma$  is a known covariance matrix.
- In practice,
  - the observed data is an  $n \times p$  data matrix  $\mathbf{X}_{n \times p}$
  - $\Sigma$  is unknown, which can be estimated by the sample covariance matrix  $\mathbf{S}$ .

## Apply PCA to A Data Set: Steps

- 1 Estimate  $\Sigma$  by the sample covariance matrix  $\mathbf{S}$
- 2 Compute the eigenvectors of  $\mathbf{S}$ , and denote them by  $\gamma_1, \dots, \gamma_p$
- 3 Compute the PCs
  - PC1:  $Y_1 = \mathbf{X}\gamma_1$ , which is a  $n \times 1$  vector
  - PC2:  $Y_2 = \mathbf{X}\gamma_2$ , which is a  $n \times 1$  vector
  - .....
  - PC $i$ :  $Y_i = \mathbf{X}\gamma_i$ , which is a  $n \times 1$  vector
  - Equivalently, compute

$$Y_{pc} = \mathbf{X}\Gamma$$

which is the  $n \times p$  matrix with the  $i$ th column being the  $i$ th PC.

## Section 4

### Example: Iris Data

# Outline

- PCA using raw data
- PCA using centered data
- PCA using standardized data

## Subsection 1

### PCA Using Raw Data



# Estimate $\Sigma$

```
#####  
## iris data  
##rearrange the data such as the response matrix is an n-by  
Y=cbind(SepalL=c(iris3[,1,1],iris3[,1,2],iris3[,1,3]),  
SepalW=c(iris3[,2,1],iris3[,2,2],iris3[,2,3]),  
PetalL=c(iris3[,3,1],iris3[,3,2],iris3[,3,3]),  
PetalW=c(iris3[,4,1],iris3[,4,2],iris3[,4,3]))  
#for unknown reasons, data.frame won't work but cbind works  
#alternatively, we can use the following way to define y  
#y=aperm(iris3,c(1,3,2));dim(y)=c(150,4)  
S=cov(Y)
```

# Compute PCs

```
eigen.vec=eigen(S)$vectors
eigen.val=eigen(S)$values
gamma1=eigen.vec[,1]
gamma2=eigen.vec[,2]
gamma3=eigen.vec[,3]
gamma4=eigen.vec[,4]
# The four pcs:
pc1=Y%*%gamma1
pc2=Y%*%gamma2
pc3=Y%*%gamma3
pc4=Y%*%gamma4
# Equivalently, we can obtain the nx4 matrix of PCs
PCs=Y%*%eigen.vec
```

# The Loadings (Weights)

gamma1

```
## [1]  0.36138659 -0.08452251  0.85667061  0.35828920
```

gamma2

```
## [1] -0.65658877 -0.73016143  0.17337266  0.07548102
```

gamma3

```
## [1] -0.58202985  0.59791083  0.07623608  0.54583143
```

gamma4

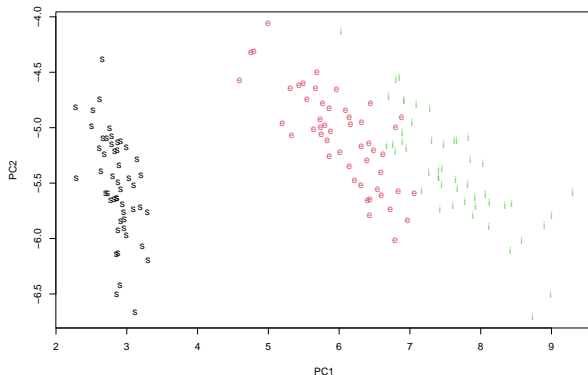
```
## [1]  0.3154872 -0.3197231 -0.4798390  0.7536574
```

# The Four PCs

- PC 1:  $Y_1 = 0.36SL - 0.08SW + 0.86PL + 0.36PL$
- PC 2:  $Y_2 = -0.66SL - 0.73SW + 0.17PL + 0.08PL$
- PC 3:  $Y_3 = -0.60SL - 0.60SW + 0.08PL + 0.55PL$
- PC 4:  $Y_4 = 0.32SL - 0.32SW - 0.48PL + 0.75PL$
- Note that PC1 loads the most on PL. This is not surprising because PL has the largest variance among all the four features.

# Visualize PC1 and PC2

```
plot(pc1,pc2,xlab="PC1", ylab="PC2", type="n")  
points(pc1[1:50], pc2[1:50], col=1, pch="s")  
points(pc1[51:100], pc2[51:100], col=2, pch="e")  
points(pc1[101:150], pc2[101:150], col=3, pch="i")
```

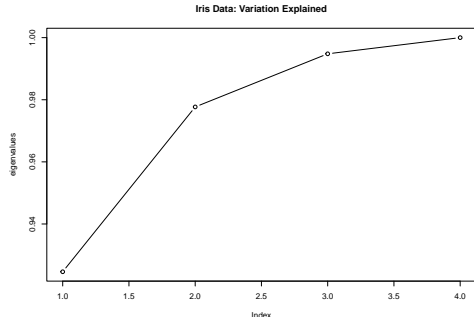


# Proportions of Variation Explained (Cumulative)

```
cumsum(eigen.val/sum(eigen.val))
```

```
## [1] 0.9246187 0.9776852 0.9947878 1.0000000
```

```
plot(cumsum(eigen.val)/sum(eigen.val), type="b", main="Iris Data: Variation Explained",  
ylab="eigenvalues")
```



## Subsection 2

### PCA Using Centered Data

# PCA Using Centered Data

*# The four pcs:*

```
pc1_c=scale(Y, scale=FALSE)%*%gamma1
```

```
pc2_c=scale(Y, scale=FALSE)%*%gamma2
```

```
pc3_c=scale(Y, scale=FALSE)%*%gamma3
```

```
pc4_c=scale(Y, scale=FALSE)%*%gamma4
```

*# or*

```
PCs_c=scale(Y, scale=FALSE)%*%eigen(S)$vectors
```



# The PCs from R

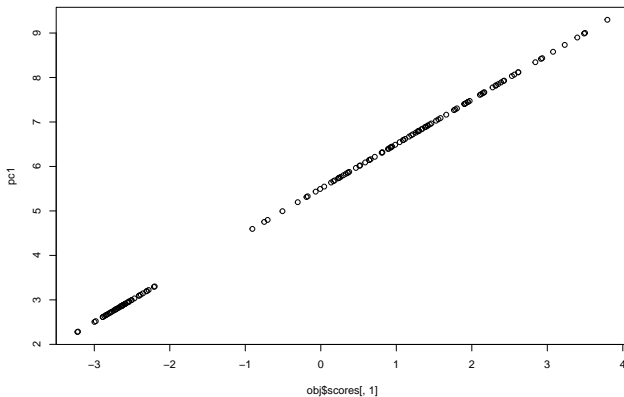
```
#help(princomp), pay attention to fix_sign  
obj=princomp(Y)  
names(obj)
```

```
## [1] "sdev"      "loadings" "center"   "scale"    "n.obs"    "scores"   "call"
```

```
#help(loadings), pay attention to cutoff  
#check loadings and scores to verify that  
#they are same as what we calculated
```

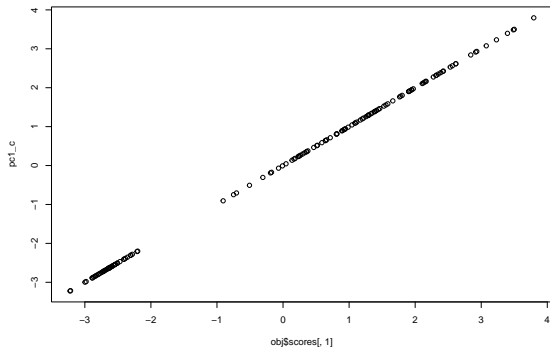
# The PCs from R vs our PCs

```
plot(obj$scores[,1], pc1)
```



# R Uses Centered Data to Compute PCA

```
plot(obj$scores[,1], pc1_c)
```



- They are identical

## Subsection 3

### PCA Using Standardized Data

## Covariance of Standardized Data

- The covariance matrix using standardized data

```
cov(scale(Y, scale=TRUE))
```

```
##           SepalL      SepalW      PetalL      PetalW
## SepalL  1.0000000 -0.1175698  0.8717538  0.8179411
## SepalW -0.1175698  1.0000000 -0.4284401 -0.3661259
## PetalL  0.8717538 -0.4284401  1.0000000  0.9628654
## PetalW  0.8179411 -0.3661259  0.9628654  1.0000000
```

# Correlation Matrix

- The correlation matrix is the covariance matrix of standardized data

```
cor(Y)
```

```
##           SepalL      SepalW      PetalL      PetalW
## SepalL  1.0000000 -0.1175698  0.8717538  0.8179411
## SepalW -0.1175698  1.0000000 -0.4284401 -0.3661259
## PetalL  0.8717538 -0.4284401  1.0000000  0.9628654
## PetalW  0.8179411 -0.3661259  0.9628654  1.0000000
```

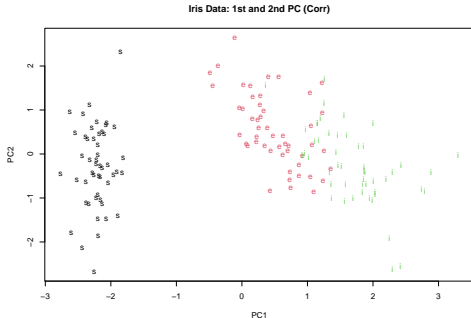
# PCA Based on Correlation Matrix

```
gamma1_s=eigen(cor(Y))$vectors[,1]  
gamma2_s=eigen(cor(Y))$vectors[,2]  
gamma3_s=eigen(cor(Y))$vectors[,3]  
gamma4_s=eigen(cor(Y))$vectors[,4]
```

## PCA Using Standardized Data

## The PCs based on Correlation Matrix

```
PCs_s=scale(Y)%*% eigen(cor(Y))$vectors
plot(PCs_s[,1:2], xlab="PC1", ylab="PC2",
     main="Iris Data: 1st and 2nd PC (Corr)", type="n")
points(PCs_s[1:50, 1:2], col=1, pch="s")
points(PCs_s[51:100, 1:2], col=2, pch="e")
points(PCs_s[101:150, 1:2], col=3, pch="i")
```





# PCA with Standardized Data in R

- The princomp function in R has an option to use correlation matrix rather than covariance matrix

```
obj.cor=princomp(Y, cor=TRUE)
```

# Standardized vs Not-Standardized

## ● Loadings

```
obj$loadings[,1:4]
```

##		Comp.1	Comp.2	Comp.3	Comp.4
##	SepalL	0.36138659	0.65658877	0.58202985	0.3154872
##	SepalW	-0.08452251	0.73016143	-0.59791083	-0.3197231
##	PetalL	0.85667061	-0.17337266	-0.07623608	-0.4798390
##	PetalW	0.35828920	-0.07548102	-0.54583143	0.7536574

```
obj.cor$loading[,1:4]
```

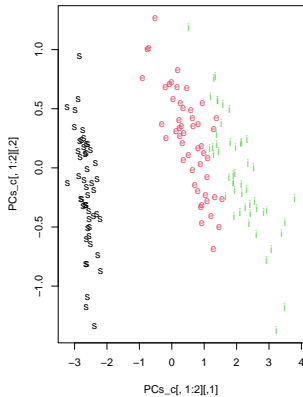
##		Comp.1	Comp.2	Comp.3	Comp.4
##	SepalL	0.5210659	0.37741762	0.7195664	0.2612863
##	SepalW	-0.2693474	0.92329566	-0.2443818	-0.1235096
##	PetalL	0.5804131	0.02449161	-0.1421264	-0.8014492
##	PetalW	0.5648565	0.06694199	-0.6342727	0.5235971

# PCA based on Cov vs Cor: 1st and 2nd PC

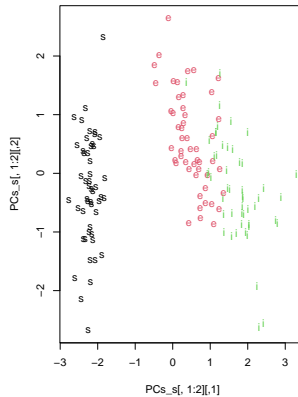
```
par(mfrow=c(1,2))
plot(PCs_c[,1:2], main="PCA using Cov", type="n")
points(PCs_c[1:50, 1:2], col=1, pch="s")
points(PCs_c[51:100, 1:2], col=2, pch="e")
points(PCs_c[101:150, 1:2], col=3, pch="i")
plot(PCs_s[,1:2], main="PCA using Cor", type="n")
points(PCs_s[1:50, 1:2], col=1, pch="s")
points(PCs_s[51:100, 1:2], col=2, pch="e")
points(PCs_s[101:150, 1:2], col=3, pch="i")
```

## PCA Using Standardized Data

PCA using Cov



PCA using Cor



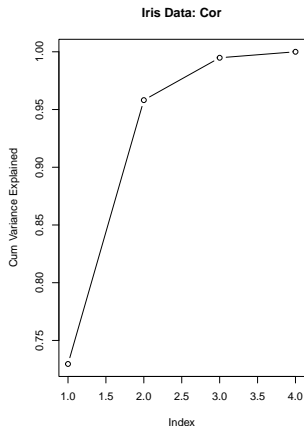
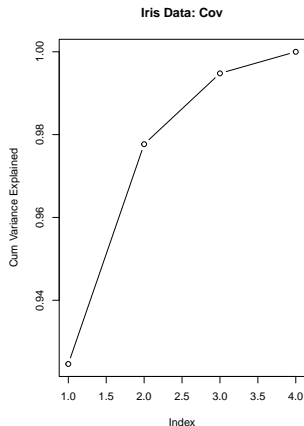
# PCA based on Cov vs Cor: Variance Explained (Cumulatively)

```
par(mfrow=c(1,2))
cumsum(eigen(cov(Y))$values)/sum(eigen(cov(Y))$values)
plot(cumsum(eigen(cov(Y))$values)/sum(eigen(cov(Y))$values),
     type="b", main="Iris Data: Cov",ylab="Cum Variance Explained")
cumsum(eigen(cor(Y))$values)/sum(eigen(cor(Y))$values)
plot(cumsum(eigen(cor(Y))$values)/sum(eigen(cor(Y))$values),
     type="b", main="Iris Data: Cor",ylab="Cum Variance Explained")
```

## PCA Using Standardized Data

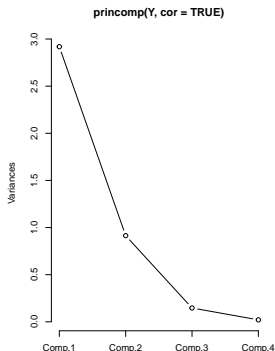
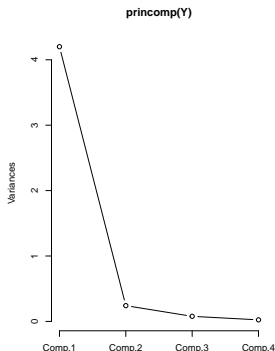
```
## [1] 0.9246187 0.9776852 0.9947878 1.0000000
```

```
## [1] 0.7296245 0.9581321 0.9948213 1.0000000
```



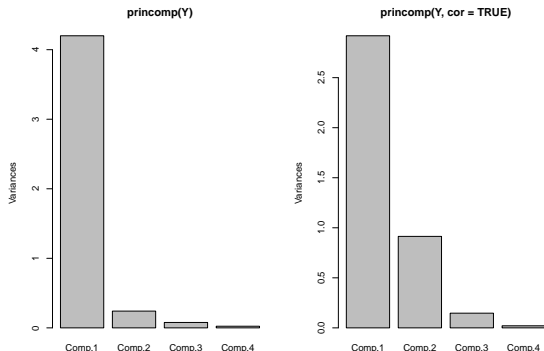
# PCA based on Cov vs Cor: Variances (Scree Plot)

```
par(mfrow=c(1,2))  
plot(princomp(Y), type="l")  
plot(princomp(Y, cor=TRUE), type="l")
```



# PCA based on Cov vs Cor: Variances (Scree Plot)

```
par(mfrow=c(1,2))  
plot(princomp(Y))  
plot(princomp(Y, cor=TRUE))
```





## Section 5

### Example: Wine Data

# The Wine Data

```
wine = read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv")
dim(wine)
```

```
## [1] 4898 12
```

```
head(wine)
```

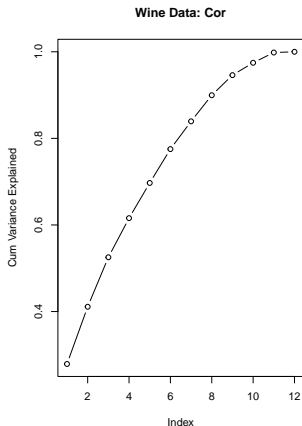
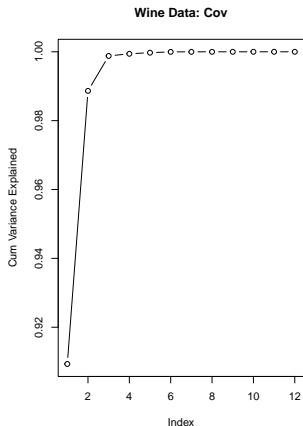
```
## fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1          7.0           0.27      0.36          20.7      0.045
## 2          6.3           0.30      0.34          1.6      0.049
## 3          8.1           0.28      0.40          6.9      0.050
## 4          7.2           0.23      0.32          8.5      0.058
## 5          7.2           0.23      0.32          8.5      0.058
## 6          8.1           0.28      0.40          6.9      0.050
## free.sulfur.dioxide total.sulfur.dioxide density pH sulphates alcohol
## 1          45          170 1.0010 3.00      0.45      8.8
## 2          14          132 0.9940 3.30      0.49      9.5
## 3          30           97 0.9951 3.26      0.44     10.1
## 4          47          186 0.9956 3.19      0.40      9.9
## 5          47          186 0.9956 3.19      0.40      9.9
## 6          30           97 0.9951 3.26      0.44     10.1
## quality
## 1          6
## 2          6
## 3          6
## 4          6
## 5          6
## 6          6
```

# PCA based on Cov vs Cor: Variance Explained

```
par(mfrow=c(1,2))
cumsum(eigen(cov(wine))$values)/sum(eigen(cov(wine))$values)
plot(cumsum(eigen(cov(wine))$values)/sum(eigen(cov(wine))$values),
     type="b", main="Wine Data: Cov",ylab="Cum Variance Explained")
cumsum(eigen(cor(wine))$values)/sum(eigen(cor(wine))$values)
plot(cumsum(eigen(cor(wine))$values)/sum(eigen(cor(wine))$values),
     type="b", main="Wine Data: Cor",ylab="Cum Variance Explained")
```

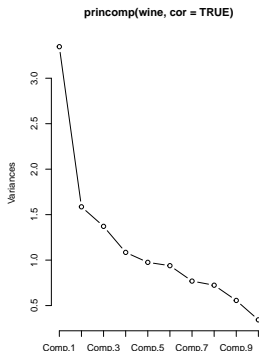
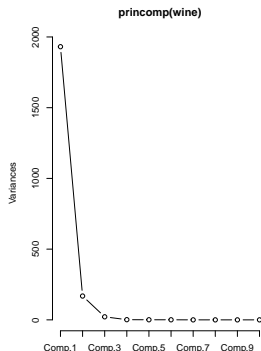
```
## [1] 0.9093312 0.9886453 0.9987968 0.9994164 0.9997397 0.9999753 0.9999839
## [8] 0.9999906 0.9999960 0.9999998 1.0000000 1.0000000

## [1] 0.2788891 0.4110633 0.5253276 0.6157327 0.6970063 0.7752366 0.8393406
## [8] 0.8996651 0.9460402 0.9746180 0.9982942 1.0000000
```



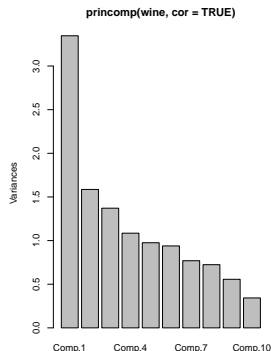
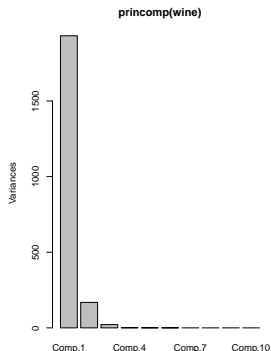
# PCA based on Cov vs Cor: Variances (Scree Plot)

```
par(mfrow=c(1,2))  
plot(princomp(wine), type="l")  
plot(princomp(wine, cor=TRUE), type="l")
```



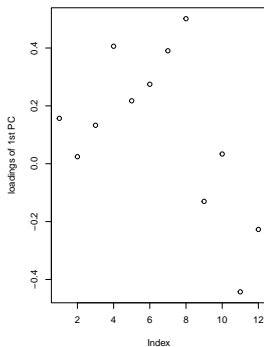
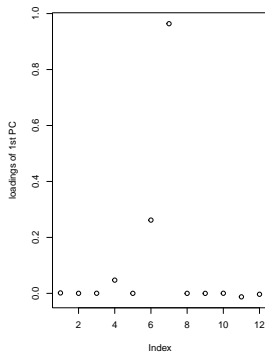
# PCA based on Cov vs Cor: Variances (Scree Plot)

```
par(mfrow=c(1,2))  
plot(princomp(wine))  
plot(princomp(wine, cor=TRUE))
```



# PCA based on Cov vs Cor: Loadings of 1st PC

```
par(mfrow=c(1,2))  
plot(princomp(wine)$loadings[,1], ylab="loadings of 1st PC")  
plot(princomp(wine, cor=TRUE)$loadings[,1], ylab="loadings of 1st PC")
```



## Wine Data: PCA\_COV and PCA\_COR Are Different

- In the wine data, raw data and standardized data give very different results
- Raw Data: the first PC dominates the rest
- Standardized data: the variances of the PCs are less different
- Why?



## Wine Data: The measurements

- There are 12 measurements / features in the wine data
- The variances are very different

```
apply(wine, 2, var)
```

```
##      fixed.acidity  volatile.acidity      citric.acid
##      7.121136e-01      1.015954e-02      1.464579e-02
##      residual.sugar      chlorides  free.sulfur.dioxide
##      2.572577e+01      4.773337e-04      2.892427e+02
## total.sulfur.dioxide      density      pH
##      1.806085e+03      8.945524e-06      2.280118e-02
##      sulphates      alcohol      quality
##      1.302471e-02      1.514427e+00      7.843557e-01
```

# Standardized vs Not-Standardized

- PCA depends on the measurement scale
- When features are not standardized, the leading PCs tend to give larger loadings for features with large variances
- When similar variables are measured using different units, standardization is typically recommended before PCA

## Section 6

### Choose $k$

## Choose the Number of PCs

- A rule of thumb? Choose the PC's that contain more information than the average amount of information per PC.
- People often choose  $k$  such that at least a certain percentage of the total variance is explained. E.g., 80%, 90%, 95%, 99%
- Scree Plots:
  - A scree plot is a plot of the eigenvalues against the number of PCs
  - we look for an “elbow point” where the decrease in eigenvalues becomes less steep. The number of PCs corresponding to the elbow point can be chosen.
- No unified solution. The choice of the number of PCs may depend on the specific problem

# Choose the Number of PCs

- Balancing dimensionality reduction and information retention.
- Too few PCs may result in significant loss of information
- Too many PCs is not efficient in dimension reduction application or research question

## The Remaining Lectures Will Cover

- Lec 12: Linear Discriminant Analysis
- Lec 13: Linear Discriminant Analysis
- Lec 14: Factor Analysis
- Lec 15: Cluster Analysis
- Lec 16: Canonical Analysis
- Lec 17: Structural Equation Modeling
- Lec 18: Conclusion