Matrix Transformations & Decomposition

Centering and standardizing, correlation and covariance matrix, projection and rotation matrices, matrix decomposition, Principal Component Analysis (PCA), Singular Value Decomposition (SVD), eigenvalues, eigenvectors

Nagiza F. Samatova, samatova@csc.ncsu.edu
Professor, Department of Computer Science
North Carolina State University

NC STATE Executive Education

NC STATE UNIVERSITY Department of Computer Science

Vector Transformation

CENTERING AND STANDARDIZING

Centering and Standardizing

Let $x = (x_1, x_2, \dots, x_n)$ be the column vector

The mean of the vector:

$$\boxed{\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i}$$

$$\bar{x} = \frac{2.1 + 2.5 + 4.0 + 3.6}{4} = 3.05$$

Example in R:

```
> x <- c(2.1, 2.5, 4.0, 3.6)
> mean(x)
[1] 3.05
```

Example in Python:

```
x = np.array([2.1,2.5,4.0,3.6])
np.mean(x)
```

```
3.05
```

Economic Growth % (x_i)	S & P 500 Returns % (y_i)
2.1	8
2.5	12
4.0	14
3.6	10

Centering

Let $x = (x_1, x_2, \dots, x_n)$ be the column vector

The mean of the vector: $$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{1}{1}\sum_{i=1}^{n} x_i$$ ← **scalar, number**

Centering the vector: center x at its mean

$$\boxed{x_c = x - \bar{x} = (x_1 - \bar{x}, x_2 - \bar{x}, \dots, x_n - \bar{x})}$$ ← **vector**

Note: The mean of the centered vector is zero: $\overline{x_c} = 0$

$$\bar{x} = \frac{2.1 + 2.5 + 4.0 + 3.6}{4} = 3.05$$

$$x_c = (2.1 - 3.05, 2.5 - 3.05, 4.0 - 3.05, 3.6 - 3.05)$$
$$= (-.95, -0.55, 0.95, 0.55)$$

Economic Growth % (x_i)	S & P 500 Returns % (y_i)
2.1	8
2.5	12
4.0	14
3.6	10

Centering in Python and in R

Centering the vector: center x at its mean

$$\boldsymbol{x}_c = \boldsymbol{x} - \overline{\boldsymbol{x}} = (x_1 - \bar{x}, x_2 - \bar{x}, \ldots, x_n - \bar{x})$$ ← **vector**

Example in R:

```
> x <- c(2.1, 2.5, 4.0, 3.6)
> mean(x)
[1] 3.05
> xc <- x - mean(x)
> xc
[1] -0.95 -0.55  0.95  0.55
```

Centering the vector: center x at its mean

Example in Python:

```
xc = x - np.mean(x)
xc
```

```
array([-0.95, -0.55,  0.95,  0.55])
```

Economic Growth % (x_i)	S & P 500 Returns % (y_i)
2.1	8
2.5	12
4.0	14
3.6	10

Standardizing in R

Economic Growth % (x_i)	S & P 500 Returns % (y_i)
2.1	8
2.5	12
4.0	14
3.6	10

Let $x = (x_1, x_2, \dots, x_n)$ **be the column vector**

Centered vector: $\boxed{\boldsymbol{x_c} = \boldsymbol{x} - \bar{\boldsymbol{x}} = (x_1 - \bar{x}, x_2 - \bar{x}, \dots, x_n - \bar{x})}$

Variance: $\boldsymbol{var}\ (\boldsymbol{x}) = \frac{(x_1-\bar{x})^2 + (x_2-\bar{x})^2 + \dots + (x_n-\bar{x})^2}{\boldsymbol{n-1}}$

Standard Deviation: $\boldsymbol{sd}(\boldsymbol{x}) = \sqrt{\boldsymbol{var}(\boldsymbol{x})}$

Standardizing using standard deviation: $\boxed{\boldsymbol{x_s} = \frac{\boldsymbol{x}}{\boldsymbol{sd}(\boldsymbol{x})}}$

Standardizing using mean & standard deviation (Z-score):

$$\boxed{Z\text{-score} = \frac{\boldsymbol{x} - \bar{\boldsymbol{x}}}{\boldsymbol{sd}(\boldsymbol{x})} = \frac{\boldsymbol{x_c}}{\boldsymbol{sd}(\boldsymbol{x})}}$$

Example in R:

```
> x <- c(2.1, 2.5, 4.0, 3.6)
> var(x)
[1] 0.80333
> sd(x)
[1] 0.89629
> sd(x) * sd(x)
[1] 0.80333
> xc * xc
[1] 0.9025 0.3025 0.9025 0.3025
> sum (xc * xc)
[1] 2.41
> sum (xc * xc) / (4-1)
[1] 0.80333
> xs <- x / sd(x)
> xs
[1] 2.3430 2.7893 4.4628 4.0166
> Z.score <- xc / sd(x)
> Z.score
[1] -1.05993 -0.61364  1.05993  0.61364
```

Standardizing in Python

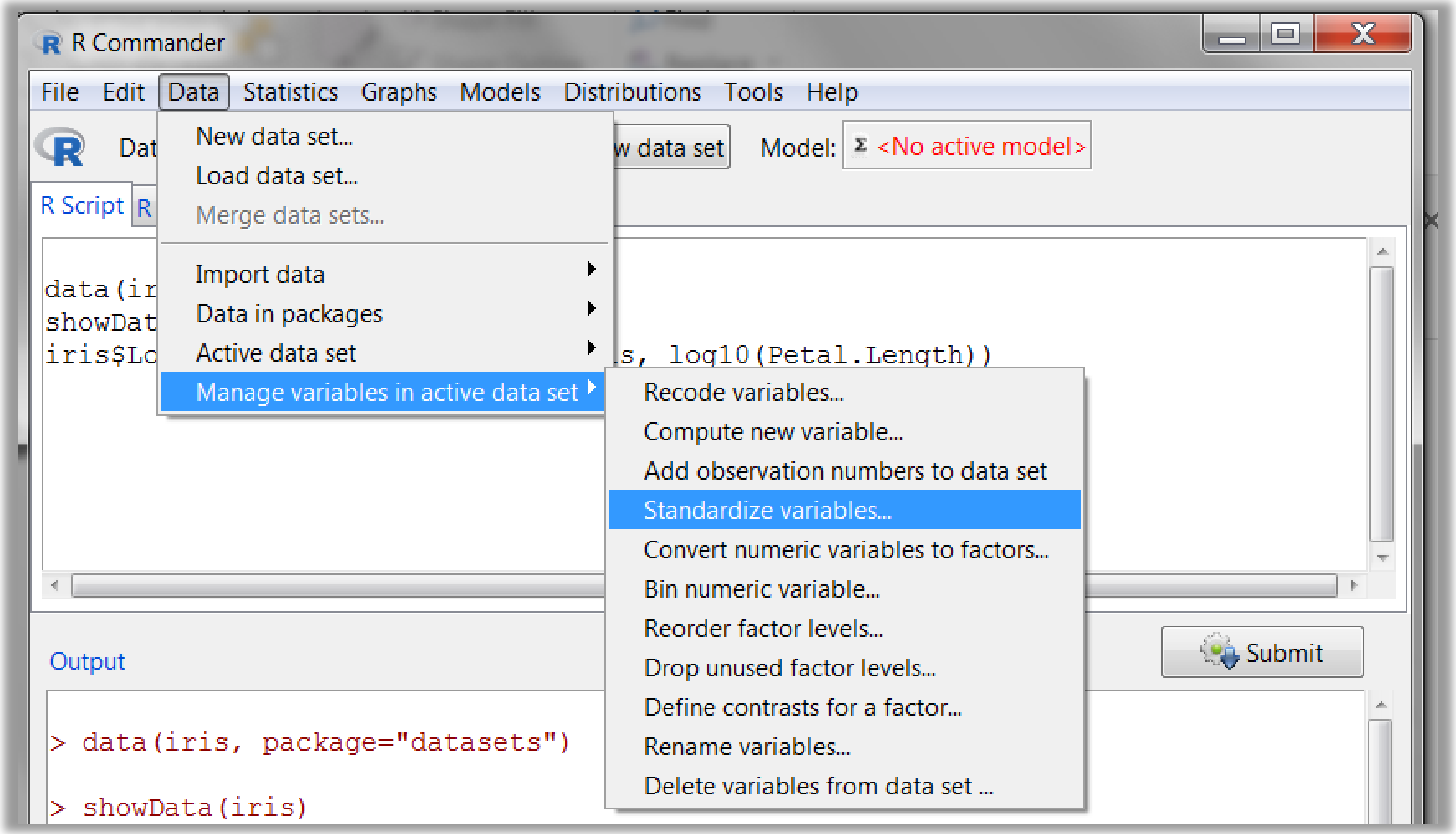
```
x = np.array([2.1,2.5,4.0,3.6])
var_x = np.var(x,ddof=1)
sd = np.std(x,ddof=1)
sd_squared = sd**2
xc_squared = xc**2
sum_xc_squared = np.sum(xc_squared)
var = sum_xc_squared/(len(x)-1)
xs = x/sd
z_score = xc/sd

print(round(var_x, 5))
print(round(sd, 5))
print(round(sd_squared, 5))
print(xc_squared)
print(round(sum_xc_squared, 2))
print(round(var, 5))
print(xs)
print(z_score)
```

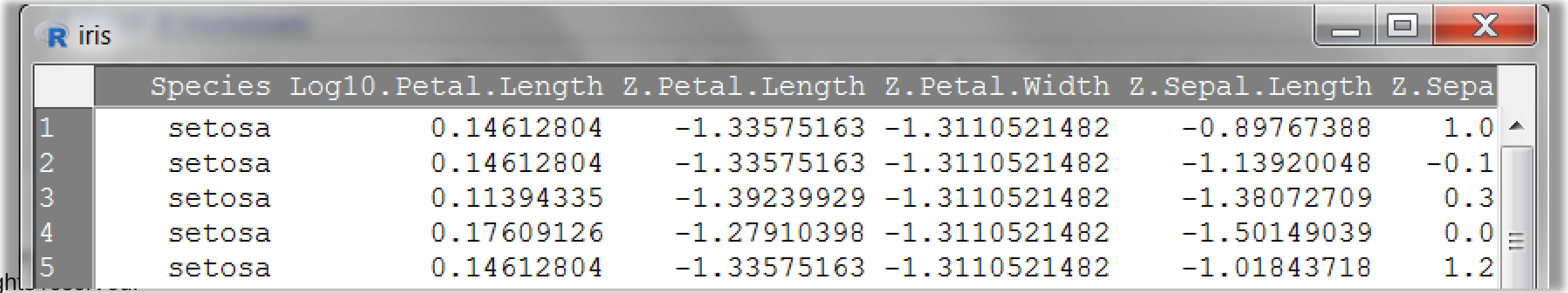
Economic Growth % (x_i)	S & P 500 Returns % (y_i)
2.1	8
2.5	12
4.0	14
3.6	10

Output:

```
0.80333
0.89629
0.80333
[0.9025 0.3025 0.9025 0.3025]
2.41
0.80333
[2.34299521 2.78928001 4.46284802 4.01656322]
[-1.0599264 -0.6136416  1.0599264  0.6136416]
```

Apply Transformations: Standardize Variables





Other Types of Matrices

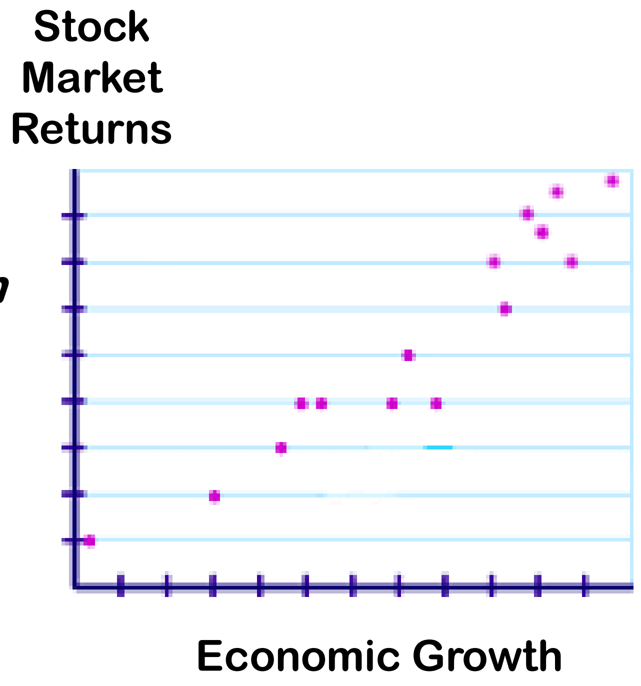
CORRELATION AND COVARIANCE MATRIX

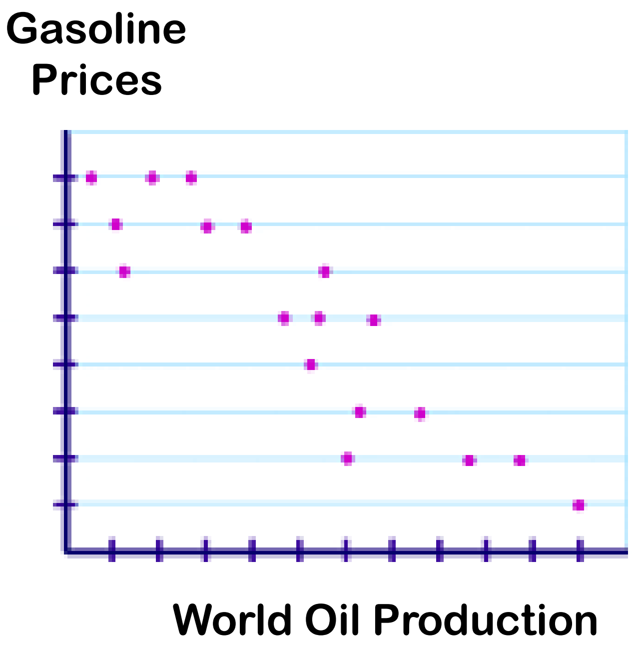
Covariance and Correlation

Covariance and correlation describe how two variables are related.

- **Variables are positively related if they move in the same direction.**
- **Variables are inversely related if they move in opposite directions.**

as economic growth increases, stock market returns also increase





as oil production increases, gas prices fall

Covariance: Formula

$\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ **: the independent variable**

$\boldsymbol{y} = (y_1, y_2, \dots, y_n)$ **: the dependent variable**

$\overline{\boldsymbol{x}}$**: the mean of** x

$\overline{\boldsymbol{y}}$**: the mean of** y

$\boldsymbol{n}$**: the number of points in the sample**

$$\boxed{\boldsymbol{cov}\ (\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{i=1}^{n}(\boldsymbol{x_i} - \overline{\boldsymbol{x}})(\boldsymbol{y_i} - \overline{\boldsymbol{y}})}{\boldsymbol{n} - \mathbf{1}}}$$

$$\boxed{\boldsymbol{cov}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\mathbf{1}}{\boldsymbol{n} - \mathbf{1}}\ \boldsymbol{x_c^T y_c}}$$

Cross-product (inner product) of centered variables normalized by the sample size minus 1 ($n - 1$**).**

Covariance: Example in Python

centering

centering

normalized cross-product

covariance

```
x = np.array([2.1,2.5,4.0,3.6])
xc = x - np.mean(x)
y = np.array([8,12,14,10])
yc = y - np.mean(y)

norm_cross_prod = np.sum(np.matmul(xc.T,yc))/(len(y)-1)
print(norm_cross_prod)
np.cov(x,y)[1,0]
```

```
1.5333333333333332

1.5333333333333332
```

Economic Growth % (x_i)	S & P 500 Returns % (y_i)
2.1	8
2.5	12
4.0	14
3.6	10

Covariance: Example in R

$$cov(x,y) = \frac{1}{n-1}\, x_c^T y_c$$

$$cov\,(x,y) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{n-1}$$

Cross-product (inner product) of centered variables normalized by the sample size minus 1 ($n-1$).

Economic Growth % (x_i)	S & P 500 Returns % (y_i)
2.1	8
2.5	12
4.0	14
3.6	10

centering

centering

normalized cross-product

covariance

```
> x <- c(2.1, 2.5, 4.0, 3.6)
> xc <- x - mean(x)
> y <- c(8, 12, 14, 10)
> yc <- y - mean(y)
>
> sum (t(xc) %*% yc) / (length(y) - 1)
[1] 1.5333
> cov (x, y)
[1] 1.5333
```

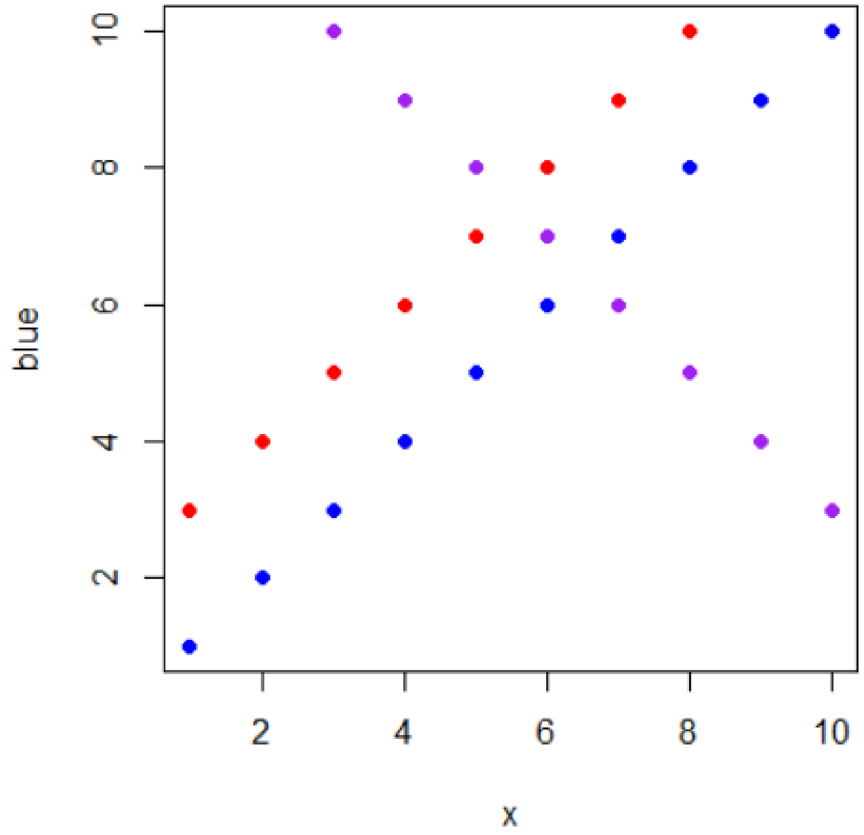
Correlation: Formula

$$\boxed{cor(x,y) = \frac{cov(x,y)}{sd(x) \times sd(y)}}$$

- **Correlation is a scaled version of covariance (i.e., scaled by the standard deviations)**
- **Correlation and covariance always have the same sign (positive, negative, or 0)**
  - when the sign is positive, the variables are said to be positively correlated
  - when the sign is negative, the variables are said to be negatively correlated
  - and when the sign is 0, the variables are said to be uncorrelated
- **Correlation is dimensionless, since the numerator and denominator have the same physical units.**
- **Correlation will always take on a value between 1 and – 1:**
  - If the correlation coefficient is +1, the variables have a perfect positive correlation. This means that if one variable moves a given amount, the second moves proportionally in the same direction.
  - If correlation coefficient is –1, the variables are perfectly negatively correlated (or inversely correlated) and move in opposition to each other. If one variable increases, the other variable decreases proportionally.
  - If correlation coefficient is zero, no relationship exists between the variables. If one variable moves, no predictions about the movement of the other variable can be made.

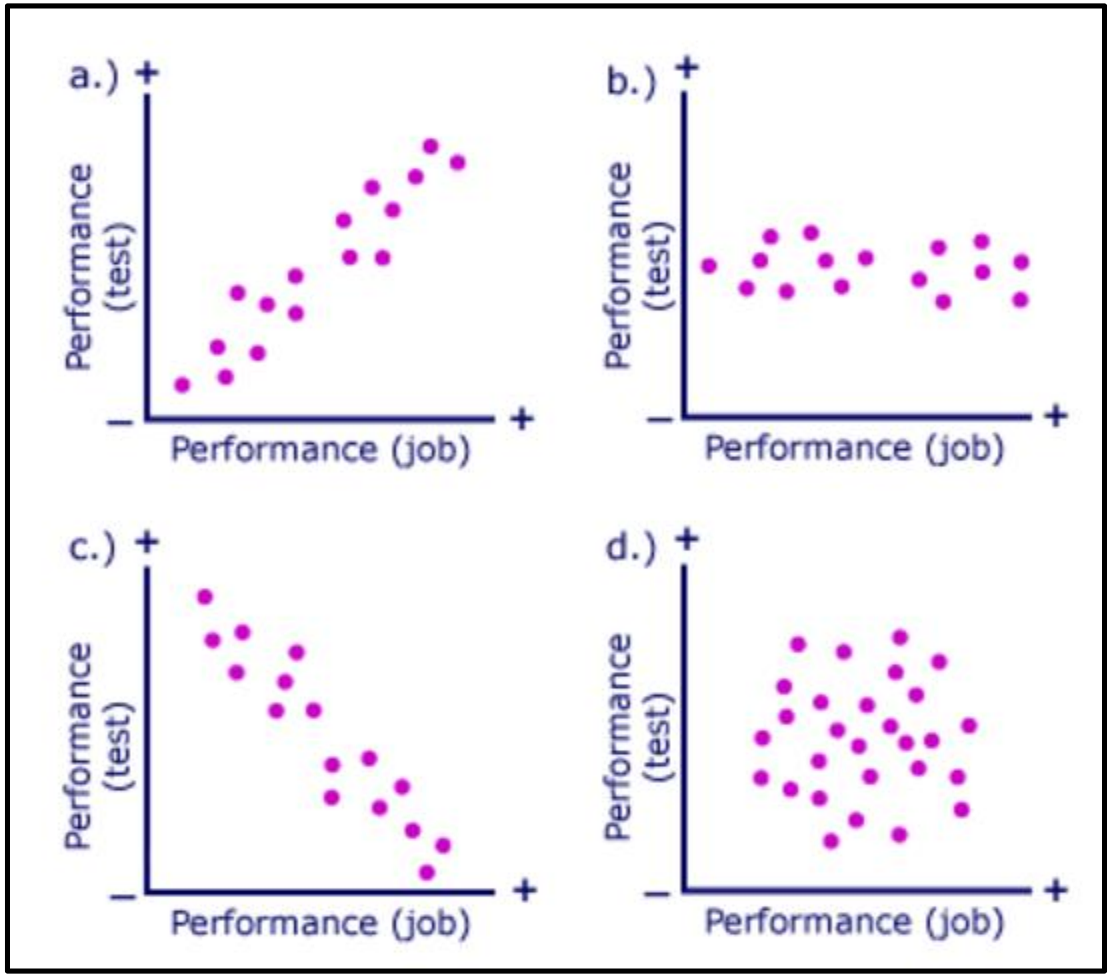
Understanding Correlation (correlation.R)

```
> x <- c(1:10)
> blue <- c(1:10)
> red <- c(3:12)
> purple <- sort(red, decreasing=TRUE)
>
> plot(x,blue,col="blue",pch=19)
> points(red,col="red",pch=19)
> points(purple,col="purple",pch=19)
>
> cor(blue,red)
[1] 1
> cor(red,purple)
[1] -1
```




```
> noise <- abs(rnorm(10, mean=5))
> noise
 [1] 5.0548 6.2234 4.0714 3.4522 4.9278 4.6648 4.8472 5.1168 4.3224 5.5515
> points(noise, col="grey", pch=19)
> cor(blue,noise)
[1] 0.0073982
```

Correlation: Examples



In each of the graphs, are job performance and test performance shown to be positively related, inversely related, or unrelated?

Answers:
a) positively related
b) unrelated
c) inversely related
d) unrelated

Exercise: Compute Covariance & Correlation

Month	Return of Stock A	Return of Market Index
1	2.3	1.3
2	2.5	5.0
3	1.9	0.8
4	2.4	1.9
5	2.1	1.1

1. Using the table, show your calculations and R codes for computing the correlation of Stock A's returns and the return of the market index.

2. Do the same for the covariance.

Exercise: Solution

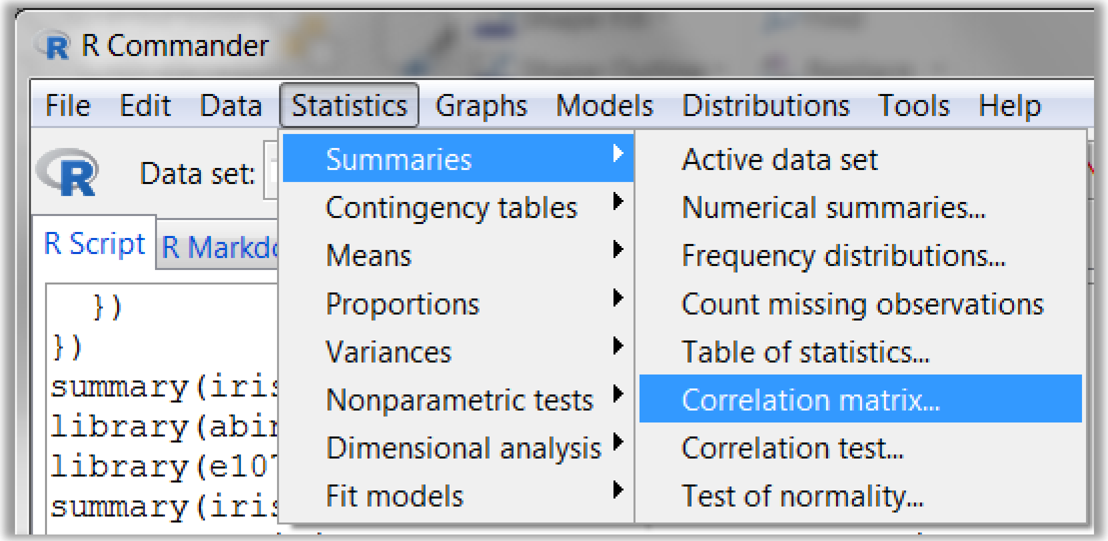
Month	Return of Stock A	Return of Market Index
1	2.3	1.3
2	2.5	5.0
3	1.9	0.8
4	2.4	1.9
5	2.1	1.1

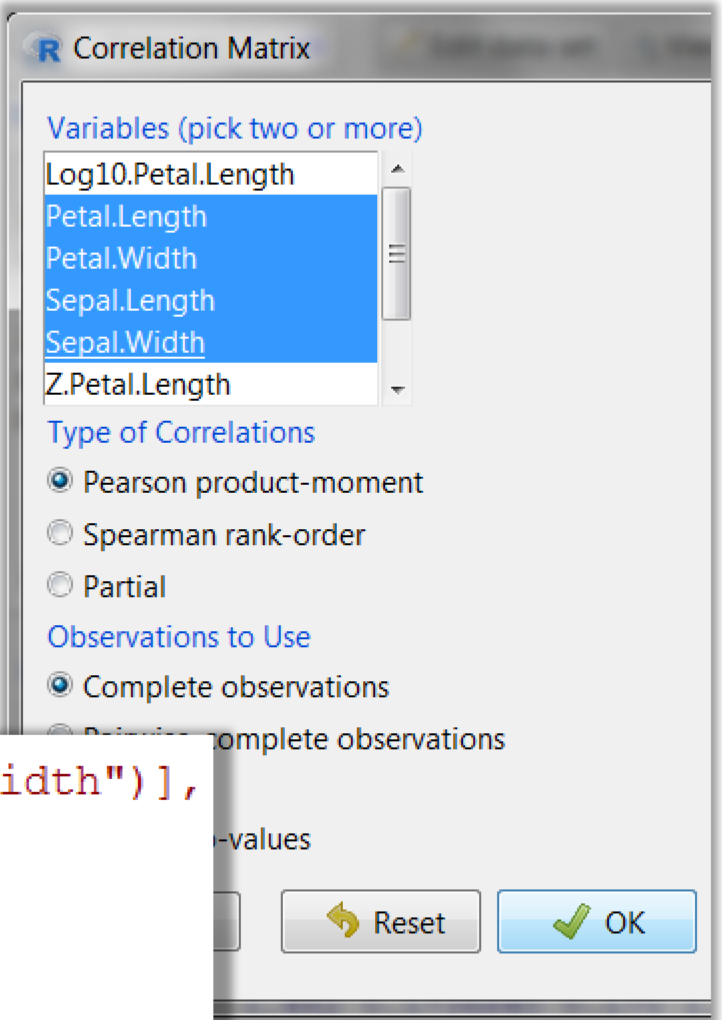
			step 1	step 2
	Stock A	Market Return	$(x_i - \bar{x})^2$	$(y_i - \bar{y})^2$
	2.30	1.30	0.0036	0.5184
	2.50	5.00	0.0676	8.8804
	1.90	0.80	0.1156	1.4884
	2.40	1.90	0.0256	0.0144
	2.10	1.10	0.0196	0.8464
Sum			0.2320	11.7480
Average	2.24	2.02		
Sum ÷ 4			0.0580	2.9370
Standard deviation			0.2408	1.7138

$$\boldsymbol{cor(x,y) = \frac{cov(x,y)}{sd(x)sd(y)} = \frac{0.31}{(0.24)(1.71)} = \frac{0.31}{0.41} = 0.76}$$

```
> x <- c(2.3,2.5,1.9,2.4,2.1)
> y <- c(1.3,5.0,0.8,1.9,1.1)
> mean(x)
[1] 2.24
> mean(y)
[1] 2.02
> sd(x)
[1] 0.24083
> sd(y)
[1] 1.7138
> cov(x,y)
[1] 0.314
> cor(x,y)
[1] 0.76079
> cov(x,y) / (sd(x)*sd(y))
[1] 0.76079
```

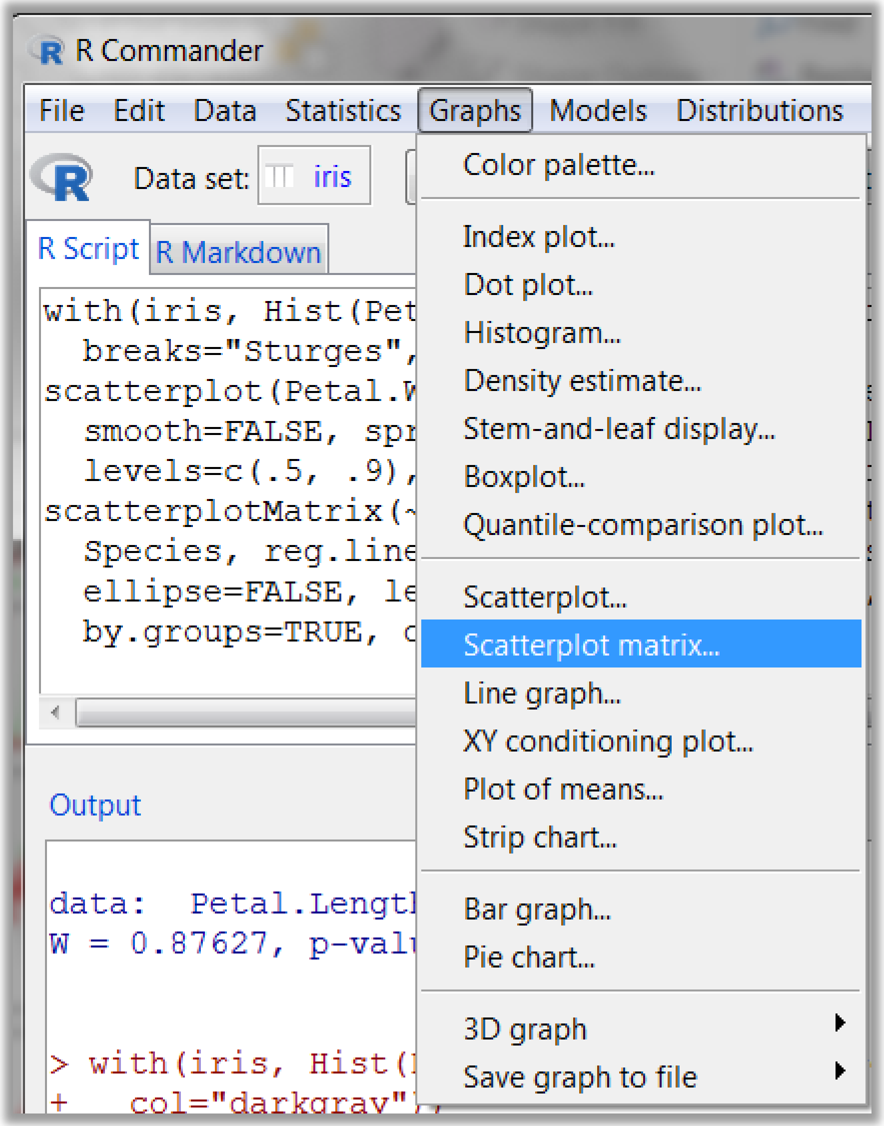
Correlation Matrix with R Commander

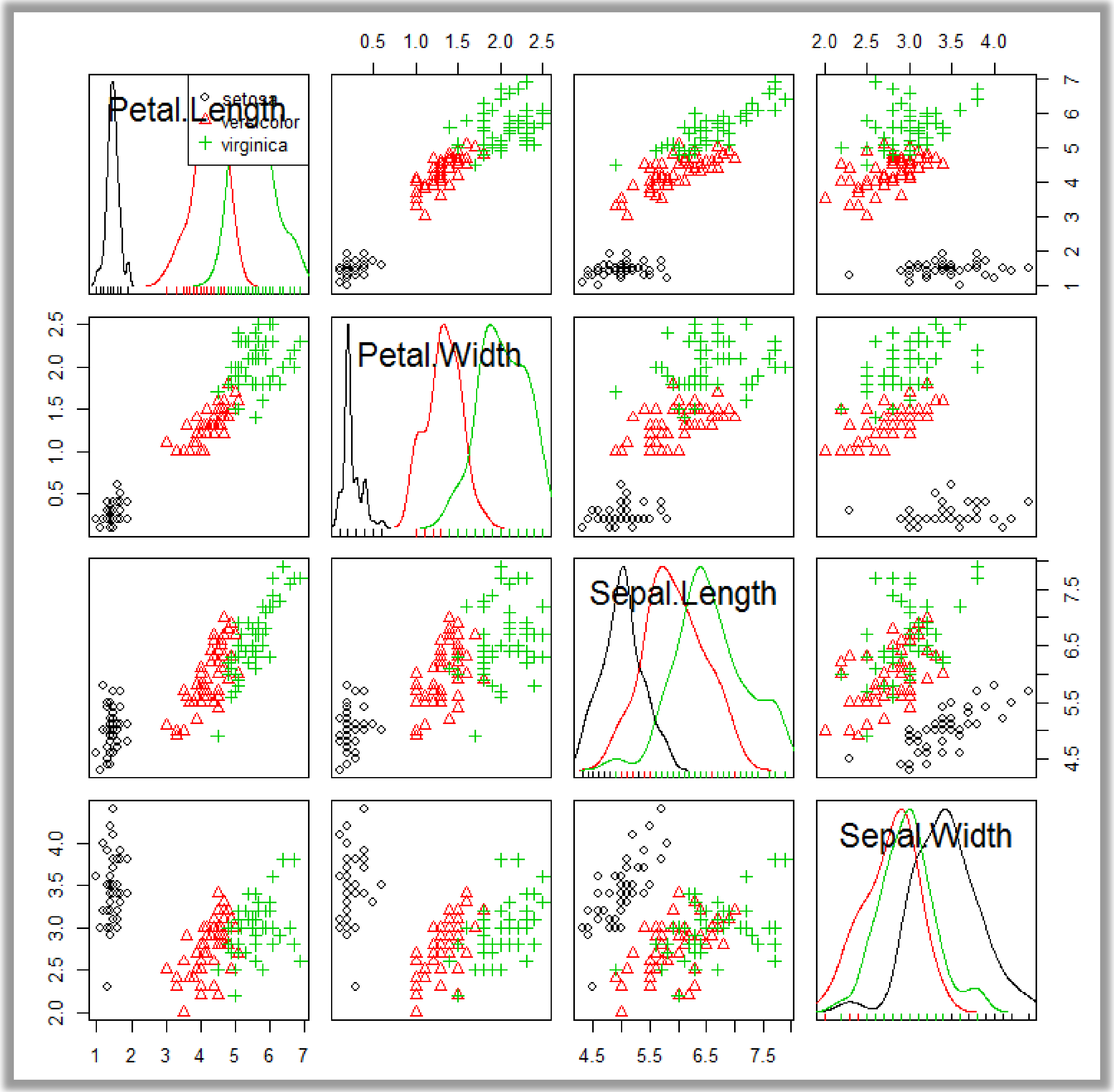





```
> cor(iris[,c("Petal.Length","Petal.Width","Sepal.Length","Sepal.Width")],
+   use="complete")
             Petal.Length Petal.Width Sepal.Length Sepal.Width
Petal.Length    1.0000000   0.9628654    0.8717538  -0.4284401
Petal.Width     0.9628654   1.0000000    0.8179411  -0.3661259
Sepal.Length    0.8717538   0.8179411    1.0000000  -0.1175698
Sepal.Width    -0.4284401  -0.3661259   -0.1175698   1.0000000
```

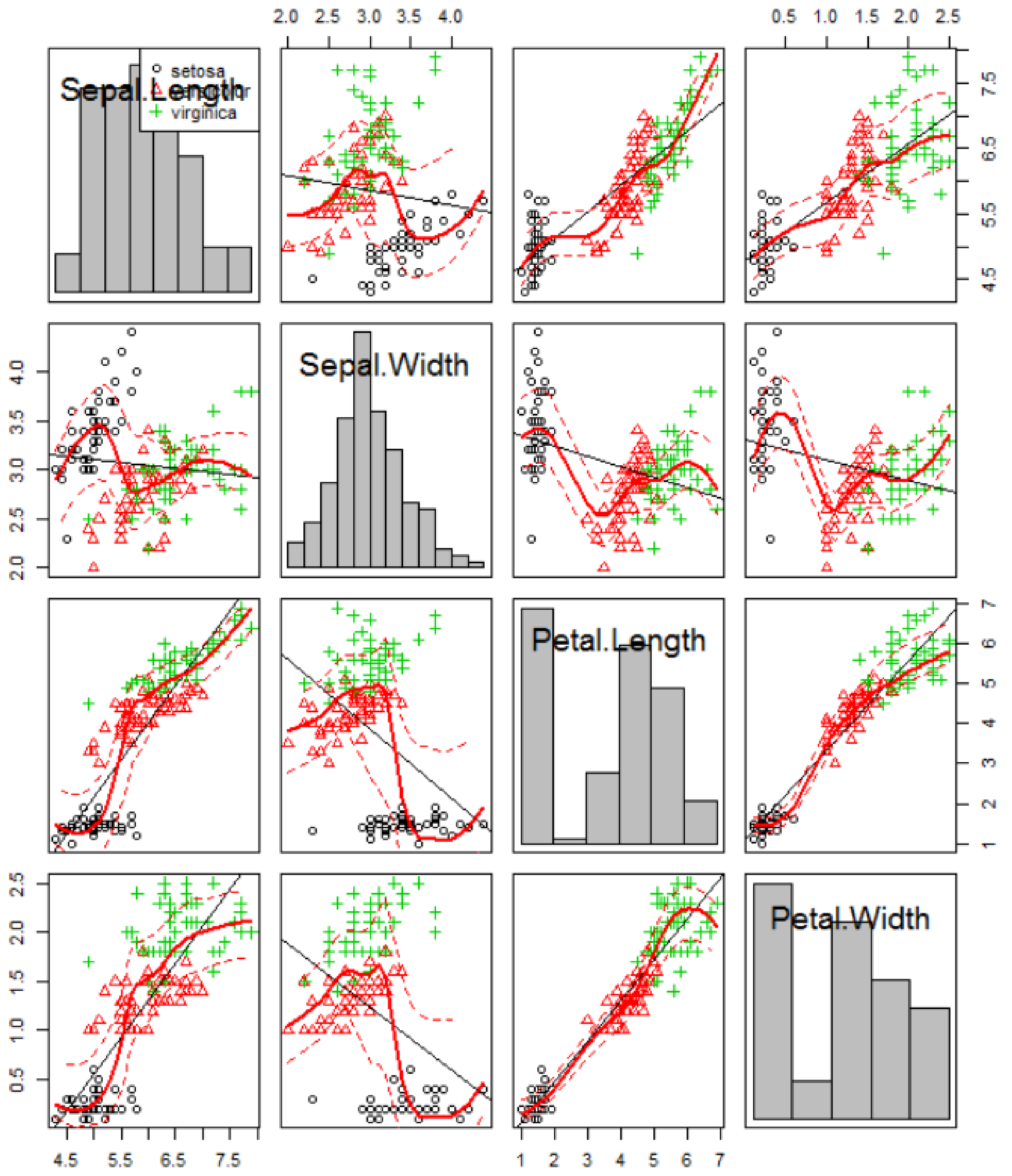
Scatterplot Matrix





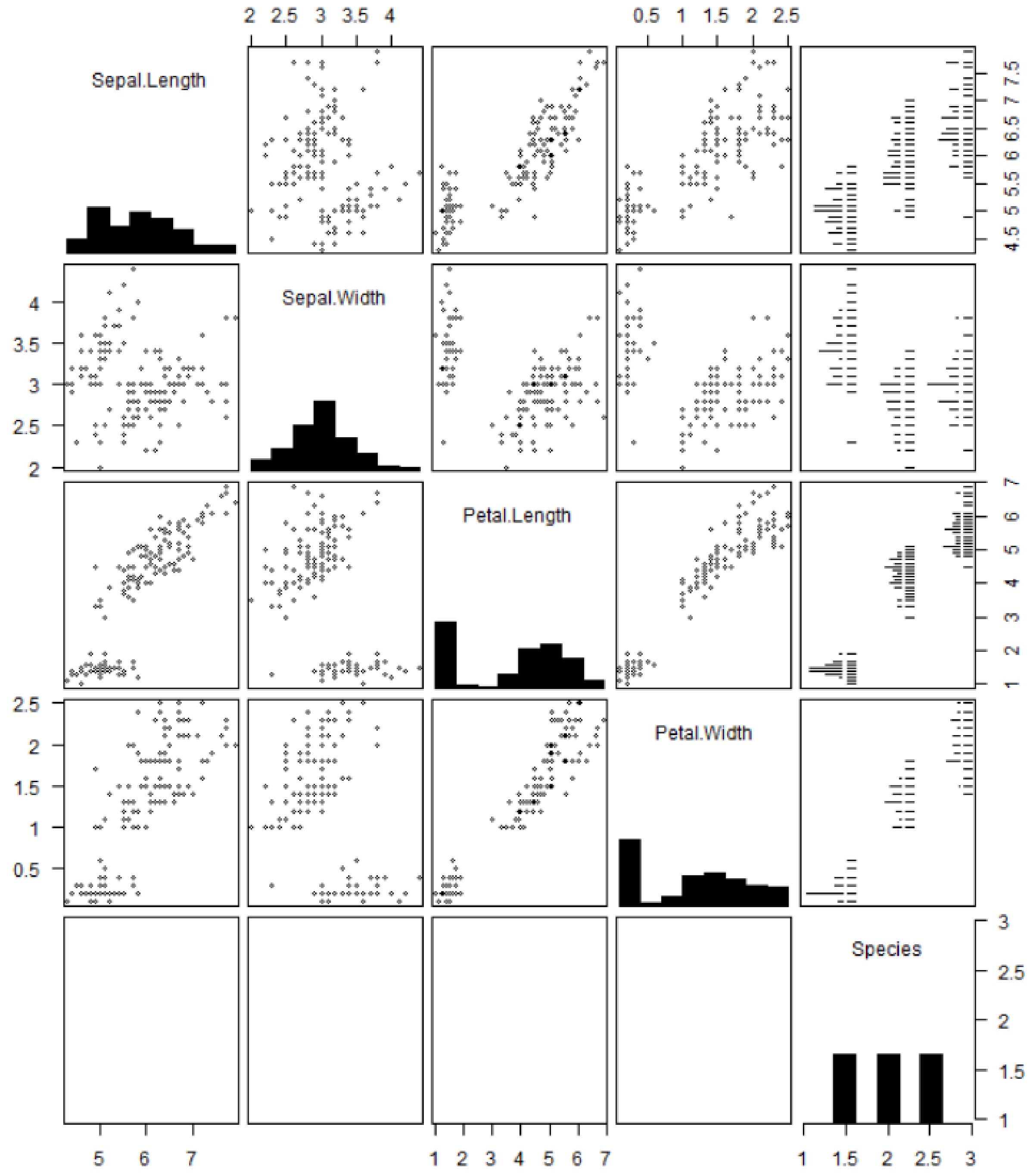
Visualizing Relationships: Continuous Variables

install.packages (“car")
library (car)
scatterplotMatrix (~
Sepal.Length + Sepal.Width +
Petal.Length + Petal.Width |
Species, data=iris,
diagonal="histogram")



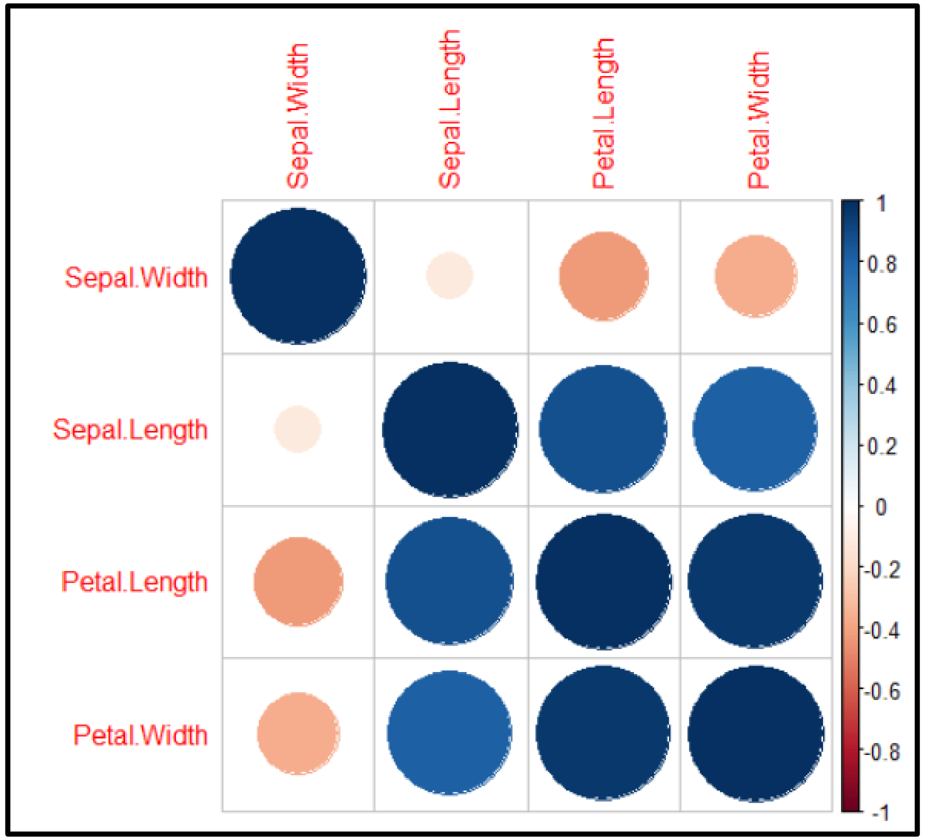
Visualizing Relationships: Continuous & Discrete

```
install.packages ("gpairs")
library (gpairs)
gpairs (iris[ , 1:5])
help (gpairs)
```

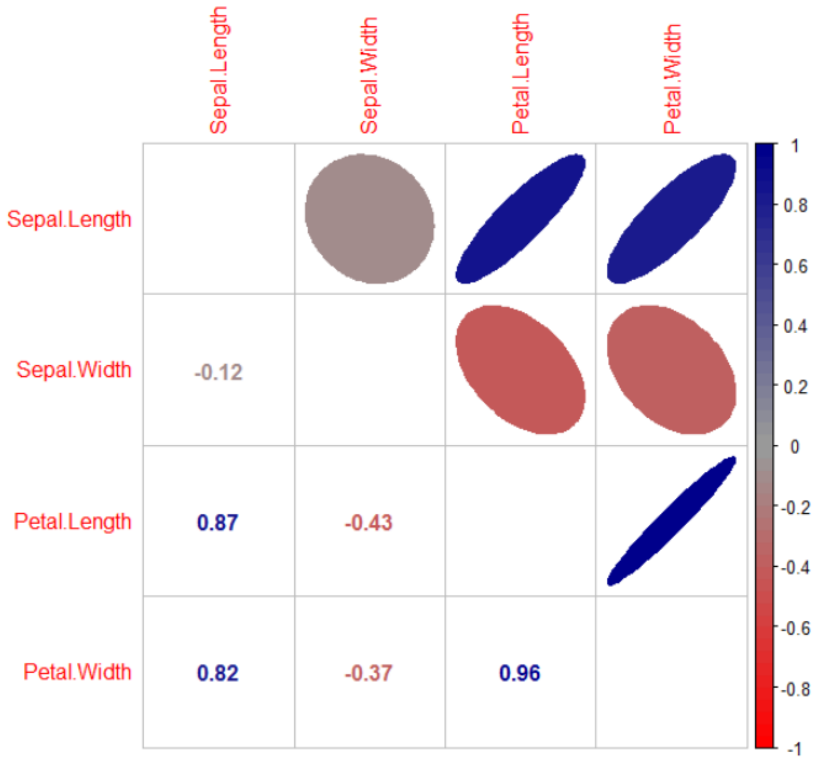


Visualizing Correlation Matrix

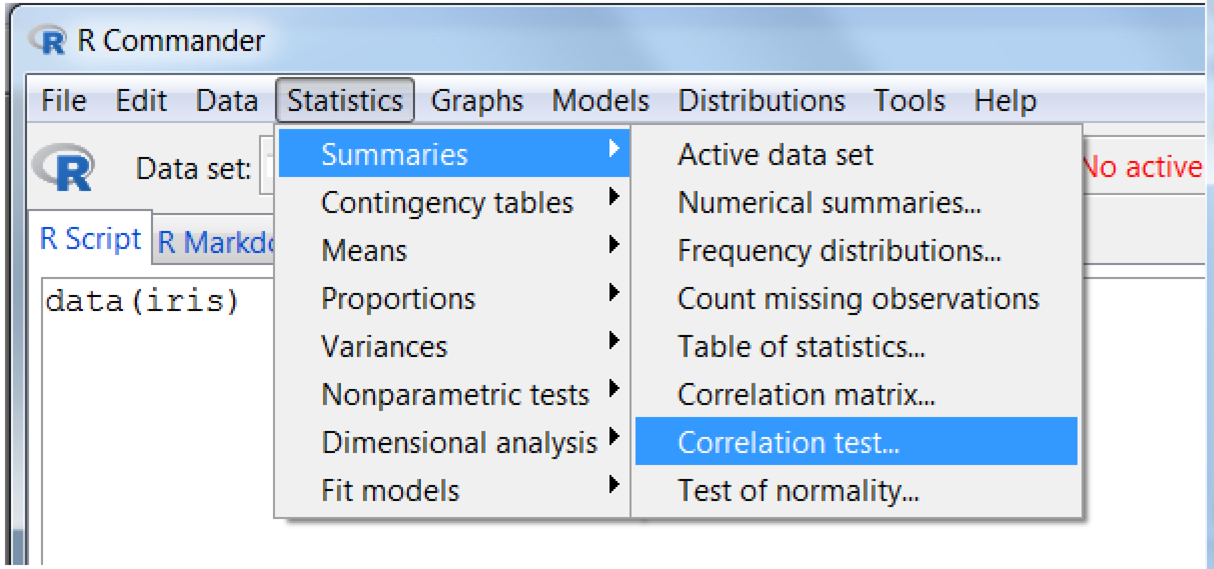
```
install.packages ("corrplot")
library (corrplot)
corrplot (cor (iris[ , 1:4], order="hclust")
help (corrplot)
```

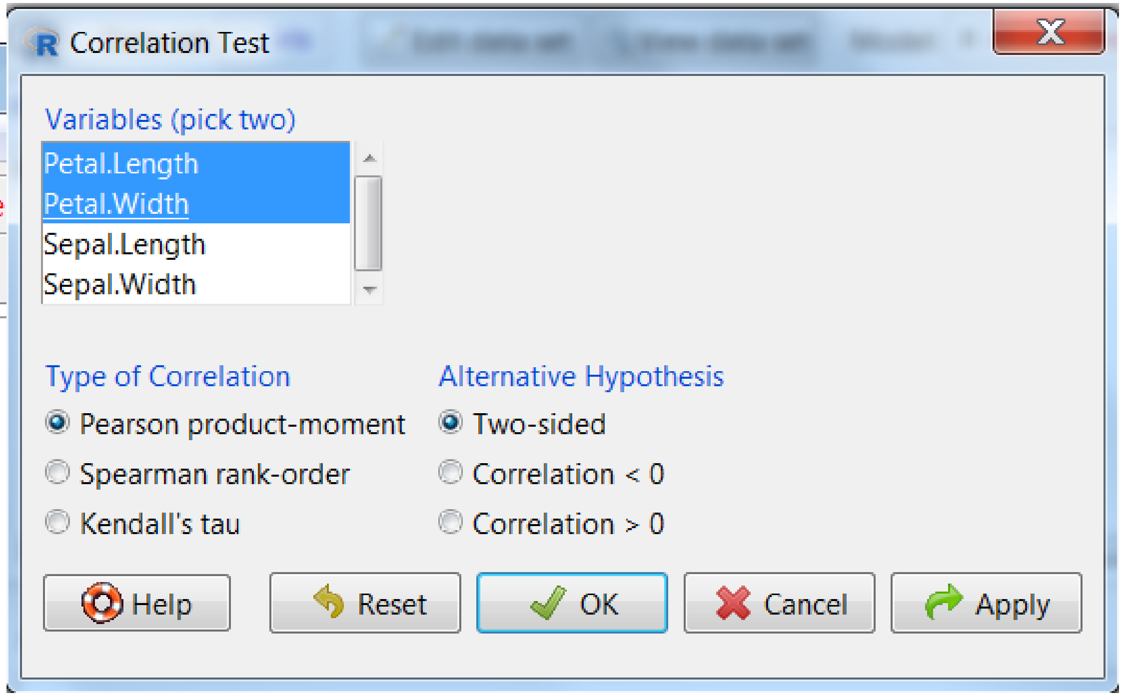



```
library (corrplot)
library(gplots)
corrplot.mixed (cor (iris[ , 1:4]),
                upper="ellipse", tl.pos="lt",
                col=colorpanel(50, "red", "grey60", "blue4"))
```



Statistical Significance of Correlations






```
        Pearson's product-moment correlation

data:  Petal.Length and Petal.Width
t = 43.387, df = 148, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9490525 0.9729853
sample estimates:
      cor
0.9628654
```

***p*-value < 0.05: statistically significant**

Linear Transformation Matrix

SCALING, REFLECTION, ROTATION

Linear Transformation Matrix

Matrix-based Transformation of Vectors

$$V_{old} \xrightarrow[\text{matrix, } \boldsymbol{A}]{} V_{new}$$

via vector-matrix multiplication

- Scaling Matrix
- Reflection Matrix
- Rotation Matrix
- Projection Matrix

Notation

$$\boldsymbol{u} = [u_1, u_2, ..., u_p] - \text{ row vector}$$

$$\boldsymbol{v} = \begin{bmatrix} v_1 \\ ... \\ v_m \end{bmatrix} - \text{ column vector}$$

$$\boldsymbol{A} = \boldsymbol{A}_{m\times p} = \begin{bmatrix} a_{11} & ... & a_{1p} \\ ... & ... & ... \\ a_{m1} & ... & a_{mp} \end{bmatrix} - m \times p \text{ matrix}$$

$$\boldsymbol{I} = \boldsymbol{I}_{m\times m} = \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & 0 & 1 \end{bmatrix} - \text{ identity matrix}$$

Scaling Matrix

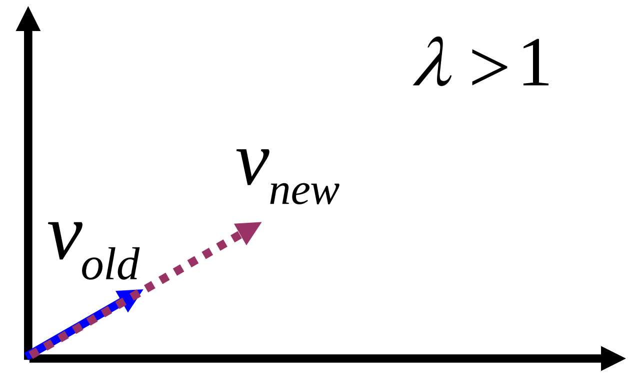
Row vectors:

$$v_{old} \in R^p \xrightarrow[\text{scalar, } \lambda \in R^+]{\text{matrix, } \boldsymbol{A}} v_{new} = \lambda \cdot v_{old} \in R^p$$

- **Unchanged:**
  - **Direction** of a vector if $\boldsymbol{\lambda} > \mathbf{0}$
- **Changed:**
  - Vector **norm/length** $\|\boldsymbol{v_{new}}\| = \boldsymbol{\lambda} \cdot \|\boldsymbol{v_{old}}\|$
  - **Direction** of a vector if $\boldsymbol{\lambda} < \mathbf{0}$

$$\boldsymbol{A}_\lambda = \lambda \cdot \boldsymbol{I}_{p \times p} = \begin{bmatrix} \lambda & 0 & \dots & 0 \\ 0 & \lambda & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \lambda \end{bmatrix}$$ – **Scaling Matrix**

$$\boxed{\boldsymbol{v_{new}} = \boldsymbol{v_{old}} \times \boldsymbol{A_\lambda} = \boldsymbol{v_{old}} \cdot \boldsymbol{\lambda} \cdot \boldsymbol{I_{p \times p}}}$$



Reflection Matrix

Row vectors:

$$v_{old} \in R^p \xrightarrow[\text{matrix, } \boldsymbol{A}]{} v_{new} = v_{old} \cdot \boldsymbol{A} \in R^p$$

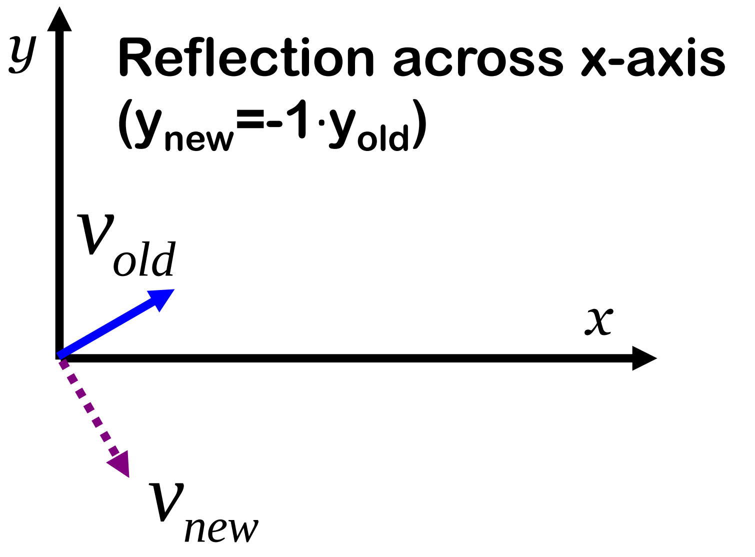
- Changed:
  - Direction of a vector
  - Reflects across one/more axis
- Unchanged:
  - Vector norm/length

$$\boldsymbol{A} = \boldsymbol{I}'_{p \times p} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 1 \end{bmatrix} -$$

Reflection Matrix: Identity Matrix with some diagonal elements set to -1

$$v_{new} = v_{old} \cdot \boldsymbol{A}$$

Equivalent to: multiplying one or more vector components by -1



Rotation Matrix

Row vectors:

$$v_{old} \in R^p \xrightarrow[\substack{\text{matrix, } \boldsymbol{A} \\ \boldsymbol{\det}(\boldsymbol{A})=1 \\ \boldsymbol{A}^T=\boldsymbol{A}^{-1}}]{} v_{new} = v_{old} \cdot \boldsymbol{A} \in R^p$$

- **Changed:**
  - **Direction of a vector**
- **Unchanged:**
  - **Vector norm/length**

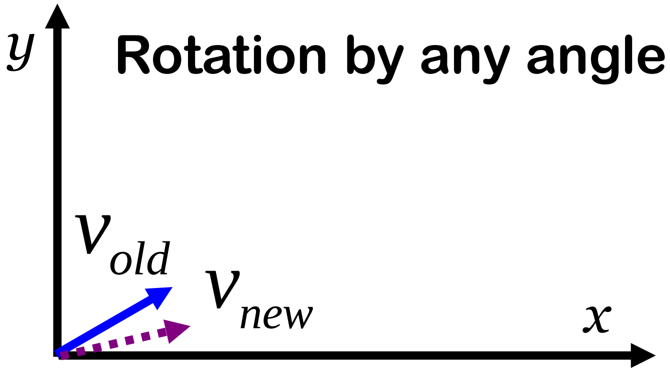
$$\boldsymbol{A} = \boldsymbol{A}_{p \times p} = \begin{bmatrix} a_{11} & \dots & a_{1p} \\ \dots & \dots & \dots \\ a_{p1} & \dots & a_{pp} \end{bmatrix}$$ – **Rotation Matrix: $\boldsymbol{p} \times \boldsymbol{p}$ orthogonal matrix**

$$\boldsymbol{A}^T = \boldsymbol{A}^{-1} - \text{orthogonal matrix}$$

$$\det(\boldsymbol{A}) = 1$$

Example: $\boldsymbol{p} = 2$

$$A(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$ – **$\boldsymbol{\theta}$-angle rotation: clockwise ($\boldsymbol{A} \times \boldsymbol{v_{old}}$) or counterclockwise ($\boldsymbol{v_{old}} \times \boldsymbol{A}$)**



Linear Transformation Matrix

PROJECTION

Projection Matrix

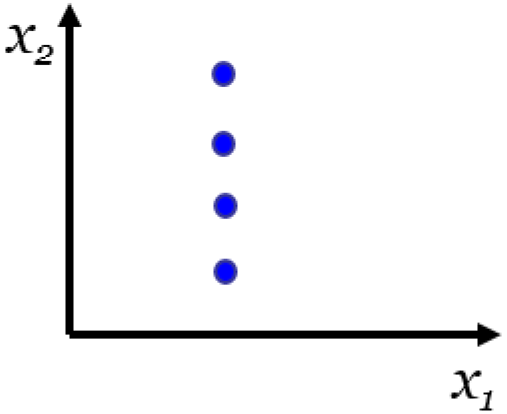
Row vectors:

$$\boldsymbol{v_{old}} \in \mathbb{R}^{\boldsymbol{p}} \longrightarrow \boldsymbol{v_{new}} = \boldsymbol{v_{old}} \times \boldsymbol{A_{p \times d}} \in \mathbb{R}^{\boldsymbol{d}}, \boldsymbol{d} < \boldsymbol{p}$$

- **Changed:**
  - **Dimensionality of a vector**
  - **Direction of a vector**
  - **Vector norm/length**

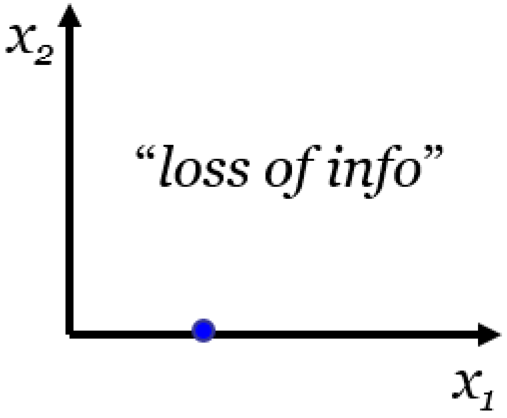
Example 1: p=2→d=1

Original data, *p*=2



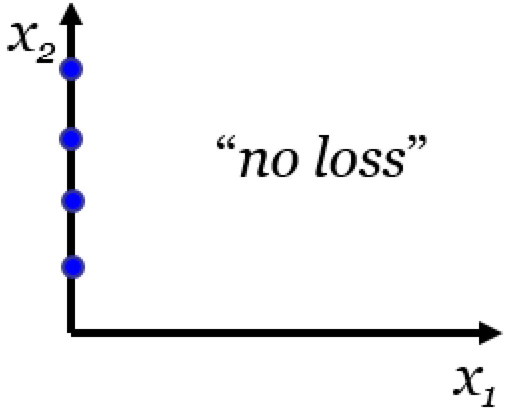
$$X_{4\times 2}$$

x_1 -projection,



$$X'_{4\times 2} = X_{4\times 2} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

x_2 -projection, *k*=1



$$X'_{4\times 2} = X_{4\times 2} \cdot \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

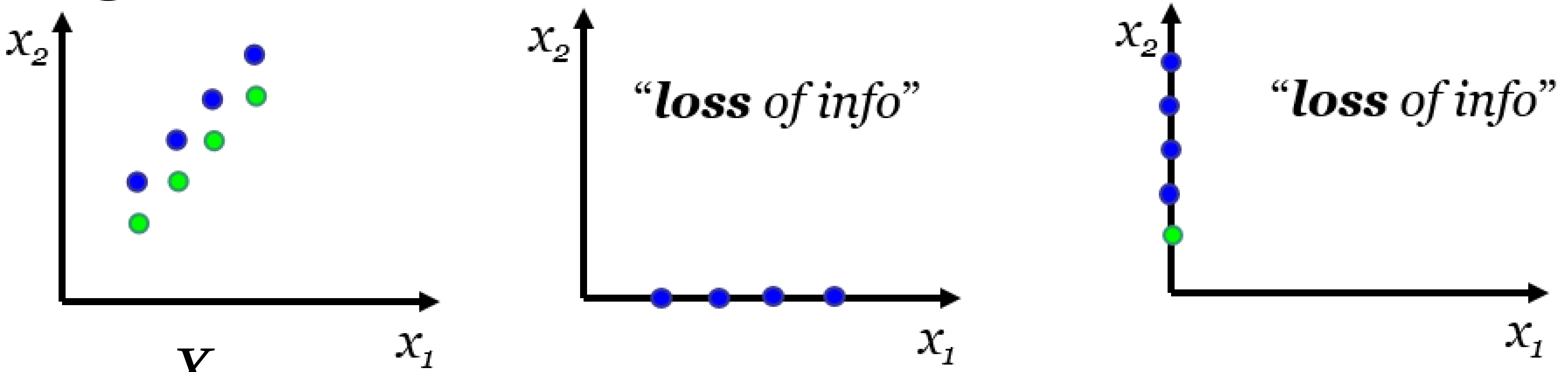
$$X'_{m\times d} = X_{m\times d} \cdot P_{d\times d}$$

$$\boldsymbol{P}_{d\times d} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 1 \end{bmatrix} - \text{ projection matrix; some diagonal elements are 0}$$

Which projection is "better"?

Example 2: Linear, Orthogonal Projection

Original data, d=2 | $\boldsymbol{x}_1$ -projection, k=1 | $\boldsymbol{x}_2$ -projection, k=1



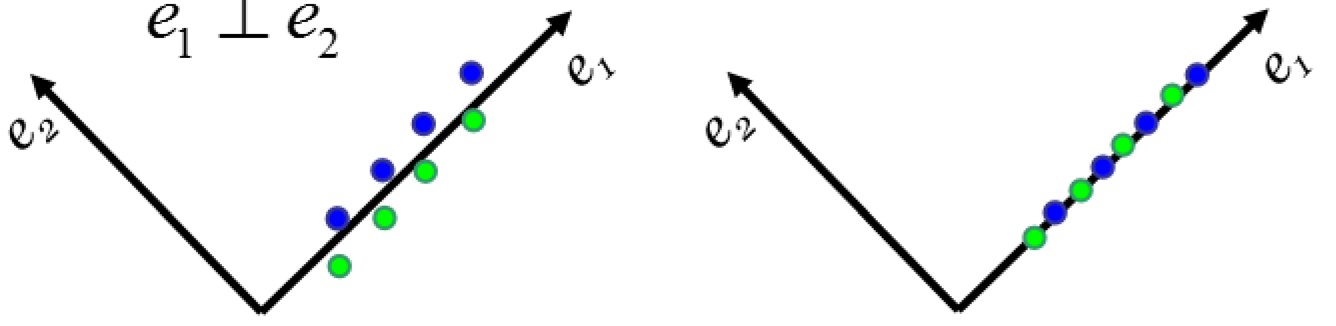
$X_{8\times 2}$

Is there a "better" projection?

Another **basis**, d=2 (rotate coord. system)

$e_1 \perp e_2$

$\boldsymbol{e}_1$-projection, k=1



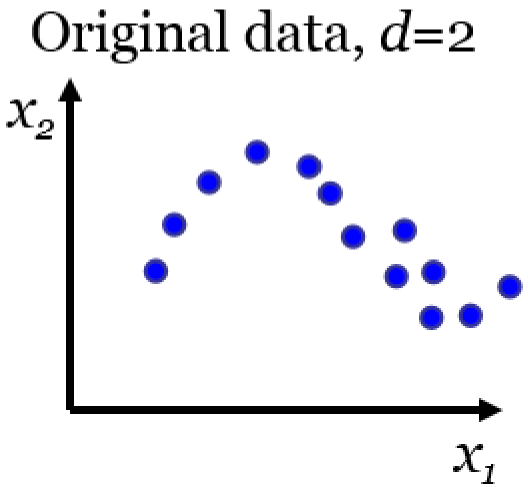
Projection:

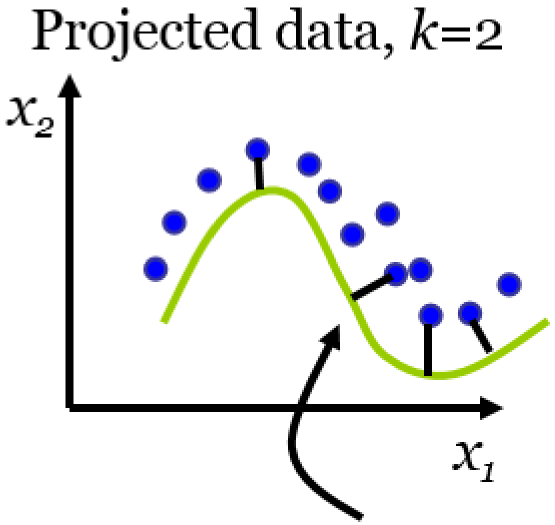
- **Linear**, $\boldsymbol{e}_1$ – line
- **Orthogonal**

$$e_1 \perp e_2$$

$\boldsymbol{e}_1$, $\boldsymbol{e}_2$ – **eigenvectors** of $\boldsymbol{X}$

Example 3: Non-linear projection





NON-linear projection

Example 4: Projection for Labeled Data

Ideal for machine learning (ML) algorithm:

- **Points within the same group come from a Gaussian distribution (spherical shapes)**
- **Points from different groups are linearly separable**

Summary: Types of Dimension Reduction

- Linear vs. NON-linear
- Orthogonal vs. non-orthogonal
- Unsupervised (unlabeled data) vs. supervised (labeled data)

Linear dimension reduction:
- Interpretable in original space
- Preserves non-linearity for visualization
- Orthogonal projections
- Non-orthogonal projections
- Favored for structure discovery

Non-linear dimension reduction:
- Lower-dimensional representation
- Interpretable w.r.t. Non-linear transformation
- 1 to 3 orders of magnitude more computation
- Favored for prediction or classification

- A lower-dimensional representation that contains the essence of the high dimensional data
- Blessing of dependence/correlation that saves us from the curse of dimensionality

Linear Orthogonal Projection

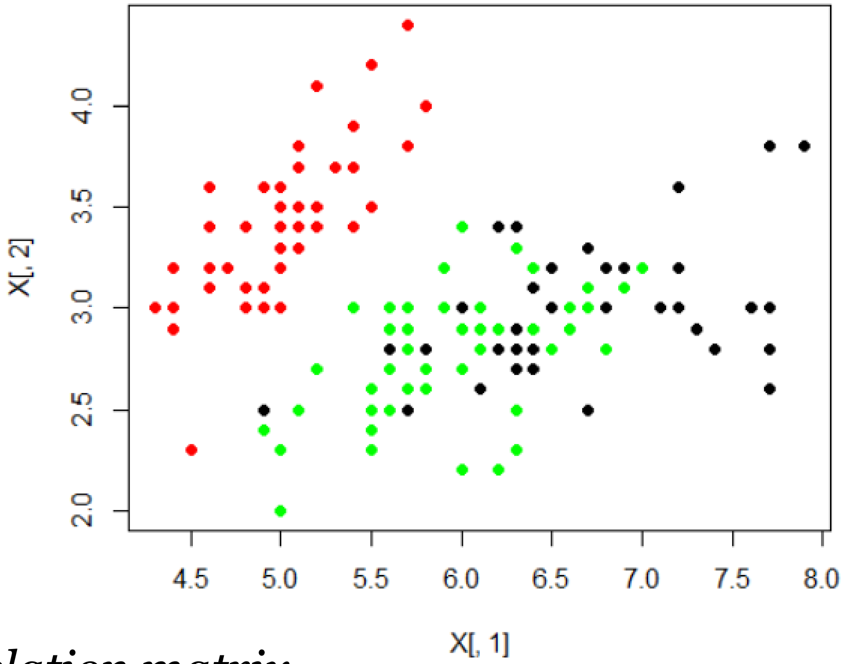
PCA: PRINCIPAL COMPONENT ANALYSIS

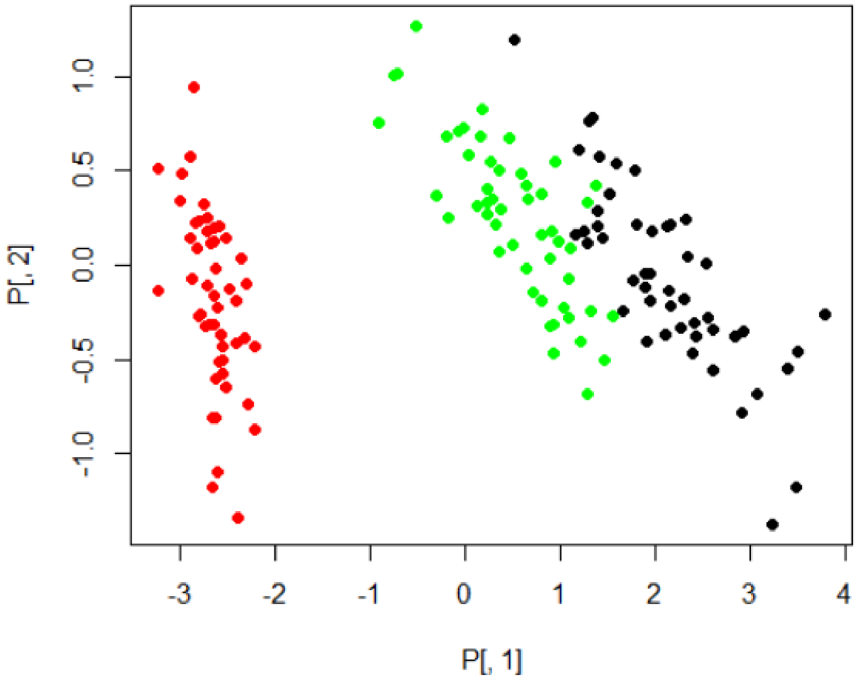
What is “better” projection: $d \rightarrow k$ ($k<d$)?

- Many definitions are possible
- Definition 1:
  - Projection that **maximizes the VARIANCE** of the original d-dimensional data upon its projection onto the target k-dimensions ($k < d$)

R Example: PCA with prcomp()

```
data (iris)
View (iris)
X <- iris [ , 1:4]
plot (X[ ,1],  X[ ,2], pch=19)
points (X [1:50, 1], X[1:50,2], col="red", pch=19)
points (X [51:100, 1], X[51:100,2], col="green", pch=19)

pca.model <- prcomp (X)
pca.model$sdev              # square roots of eigenvalues covariance/correlation matrix
plot (pca.model, type="l")  # screeplot
pca.model$rotation          # matrix of eigenvector columns
pca.model
summary (pca.model) # check proportion of variance

P <- pca.model$x           # projection of X onto eigenvectors
dim (P)
head (P)
plot (P[ ,1], P[ ,2], pch=19)
points (P [1:50, 1], P[1:50,2], col="red", pch=19)
points (P [51:100, 1], P[51:100,2], col ="green", pch=19)
```

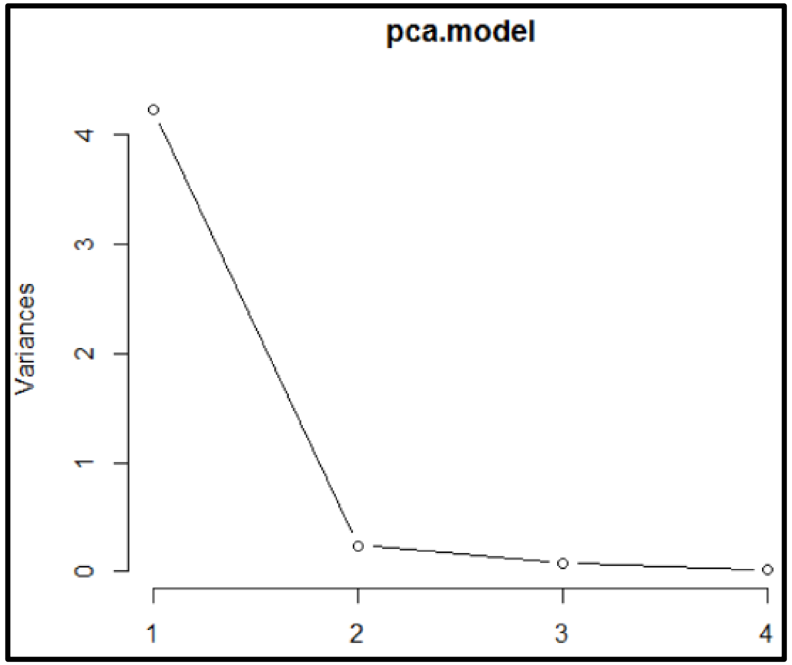




Visualizing PCA results

Screeplot

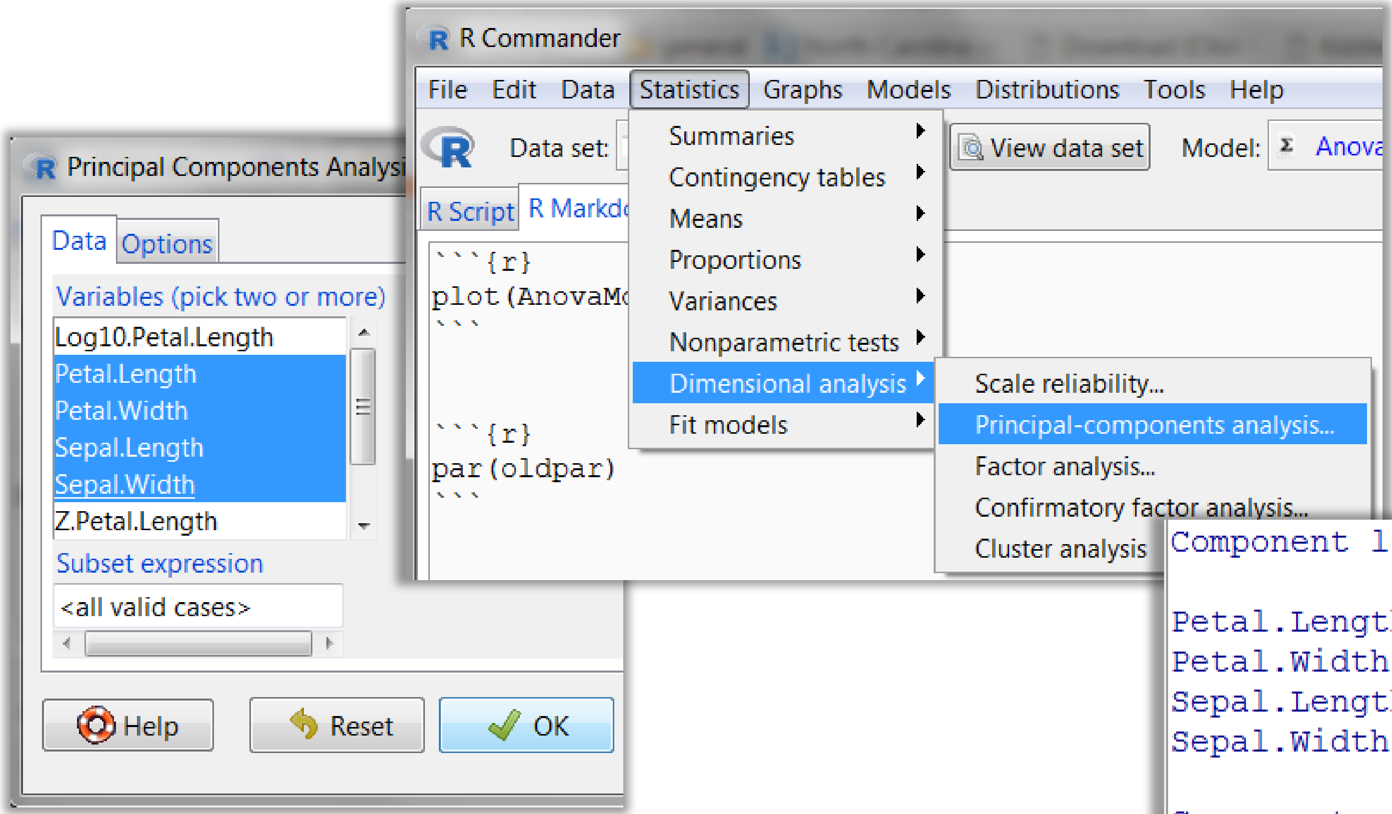
plot (pca.model, type="l")



Successive variance accounted by each component

biplot (pca.model)

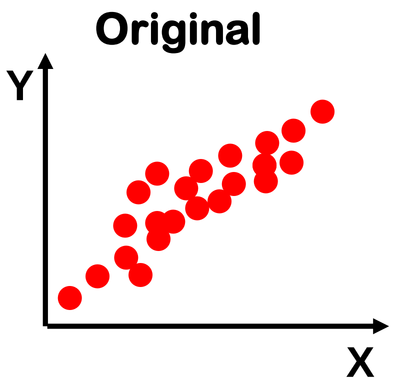
Principal Component Analysis with R Commander

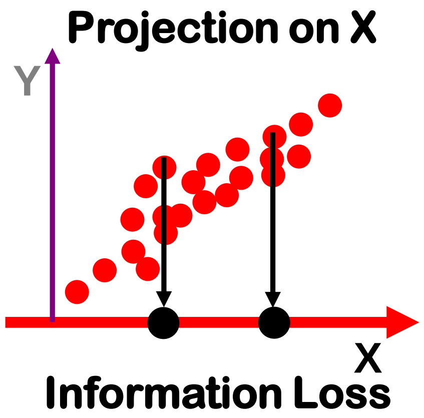



```
Component loadings:
                 Comp.1     Comp.2     Comp.3     Comp.4
Petal.Length  0.5804131 0.02449161  0.1421264  0.8014492
Petal.Width   0.5648565 0.06694199  0.6342727 -0.5235971
Sepal.Length  0.5210659 0.37741762 -0.7195664 -0.2612863
Sepal.Width  -0.2693474 0.92329566  0.2443818  0.1235096

Component variances:
    Comp.1     Comp.2     Comp.3     Comp.4
2.91849782 0.91403047 0.14675688 0.02071484

Importance of components:
                          Comp.1    Comp.2     Comp.3      Comp.4
Standard deviation     1.7083611 0.9560494 0.38308860 0.143926497
Proportion of Variance 0.7296245 0.2285076 0.03668922 0.005178709
Cumulative Proportion  0.7296245 0.9581321 0.99482129 1.000000000
```

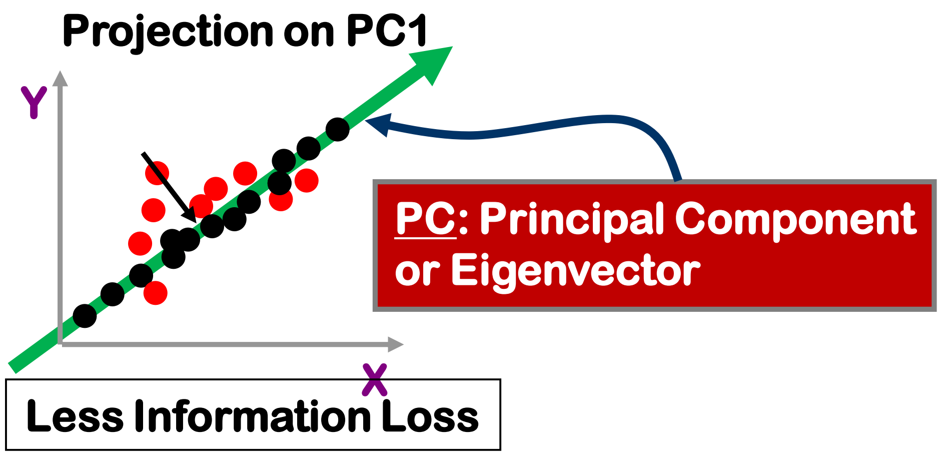
PCA: Linear Orthogonal Dimension Reduction

Principal Component Analysis (PCA) finds intrinsic dimensionality and allows for low-dimensional representation.





Principal Component Analysis is the Spectral Decomposition of the Covariance Matrix.



Matrix Decomposition

PCA: SVD OF COVARIANCE MATRIX

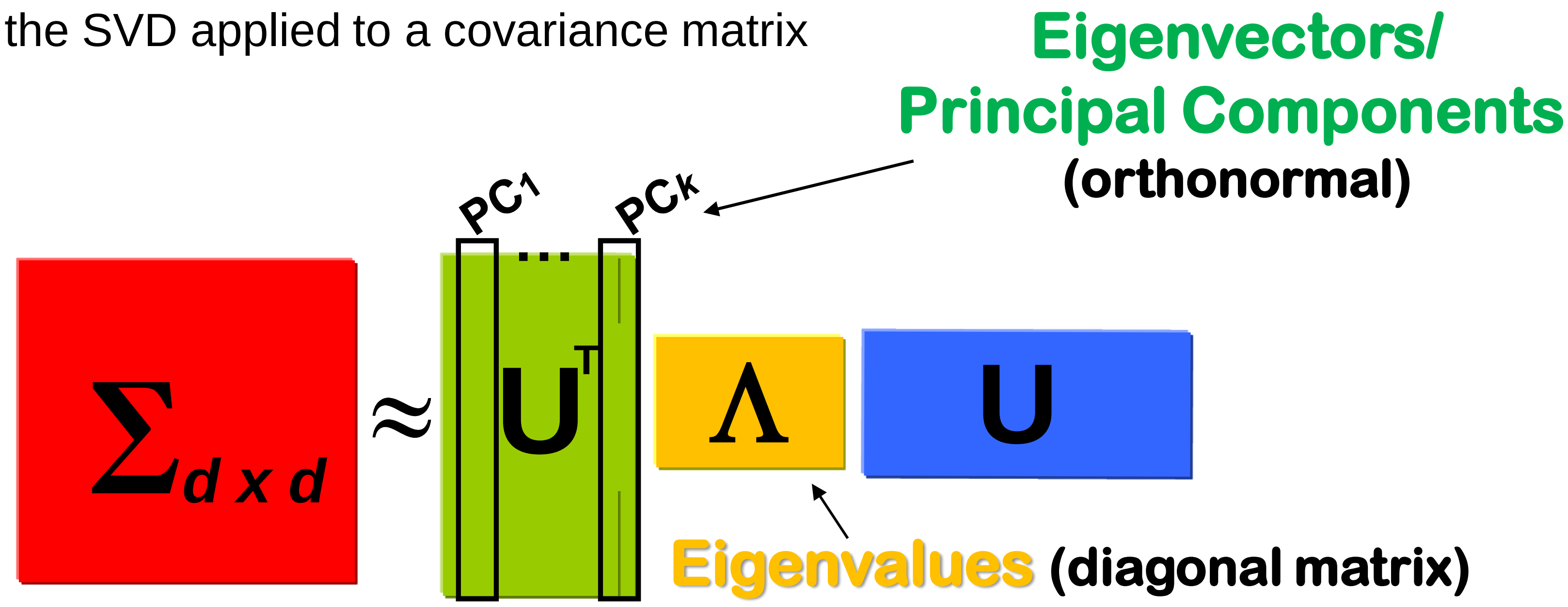
PCA is the SVD of Covariance/Correlation Matrix

Singular Value Decomposition (SVD)

The technique underlying PCA analysis

PCA = SVD (Covariance Matrix)

PCA is the SVD applied to a covariance matrix



$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \geq \ldots \geq \lambda_d \geq 0$$

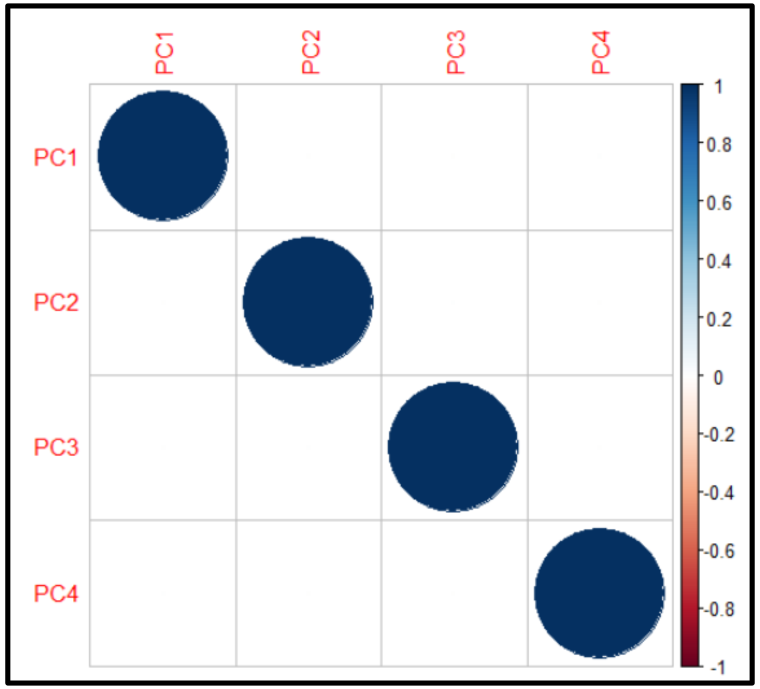
Details

`princomp` is a generic function with `"formula"` and `"default"` methods.

The calculation is done using [`eigen`] on the correlation or covariance matrix, as determined by [`cor`].

Extracted Principal Components (PCs) are Uncorrelated

```
> cor (pca.model$x)
        PC1    PC2    PC3    PC4
PC1   1e+00 -1e-16 -1e-15  2e-15
PC2  -1e-16  1e+00  2e-16 -5e-16
PC3  -1e-15  2e-16  1e+00 -1e-15
PC4   2e-15 -5e-16 -1e-15  1e+00
> corrplot (cor (pca.model$x))
```



Preserved Variability for Top-k PCs

Percentage of variability preserved
if the first **k** PCs are used for **projection**:

$$\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{d}\lambda_i} = \frac{\sum_{i=1}^{k}\lambda_i}{\text{trace}\,(\Sigma)}$$

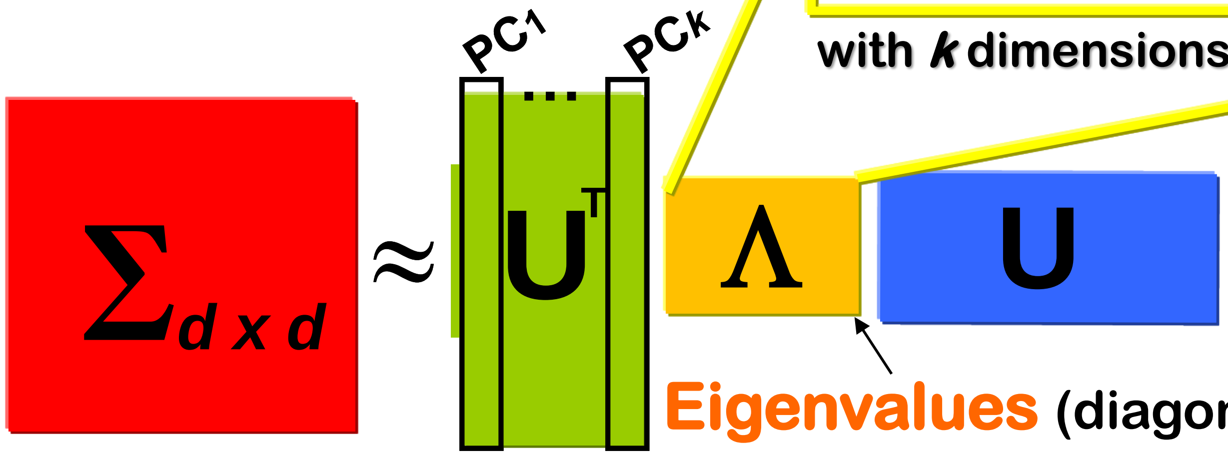
SORTED Eigenvalues

$$\underbrace{\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k}\ \geq \ldots \geq \lambda_d \geq 0$$

$$\text{Preserved variability} = \frac{\text{Sum of } \boldsymbol{k} \text{ eigenvalues}}{\text{Sum of } \boldsymbol{all} \text{ eigenvalues}}$$

with ***k*** dimensions

covariance matrix



Eigenvalues (diagonal matrix)

Key Points about PCA

- **PCA = SVD (Covariance Matrix, ∑)**
- **Linear, orthogonal projection:**
  - **"Best" linear orthogonal *k*-dimensional (*k* < *d*) view of data**
- **How "good" the *k*-dimensional view is:**

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} = \frac{\sum_{i=1}^{k} \lambda_i}{\text{trace}\,(\Sigma)}$$

Eigenvalues: Importance of eigenvectors

$$\boxed{\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \geq \ldots \geq \lambda_d \geq 0}$$

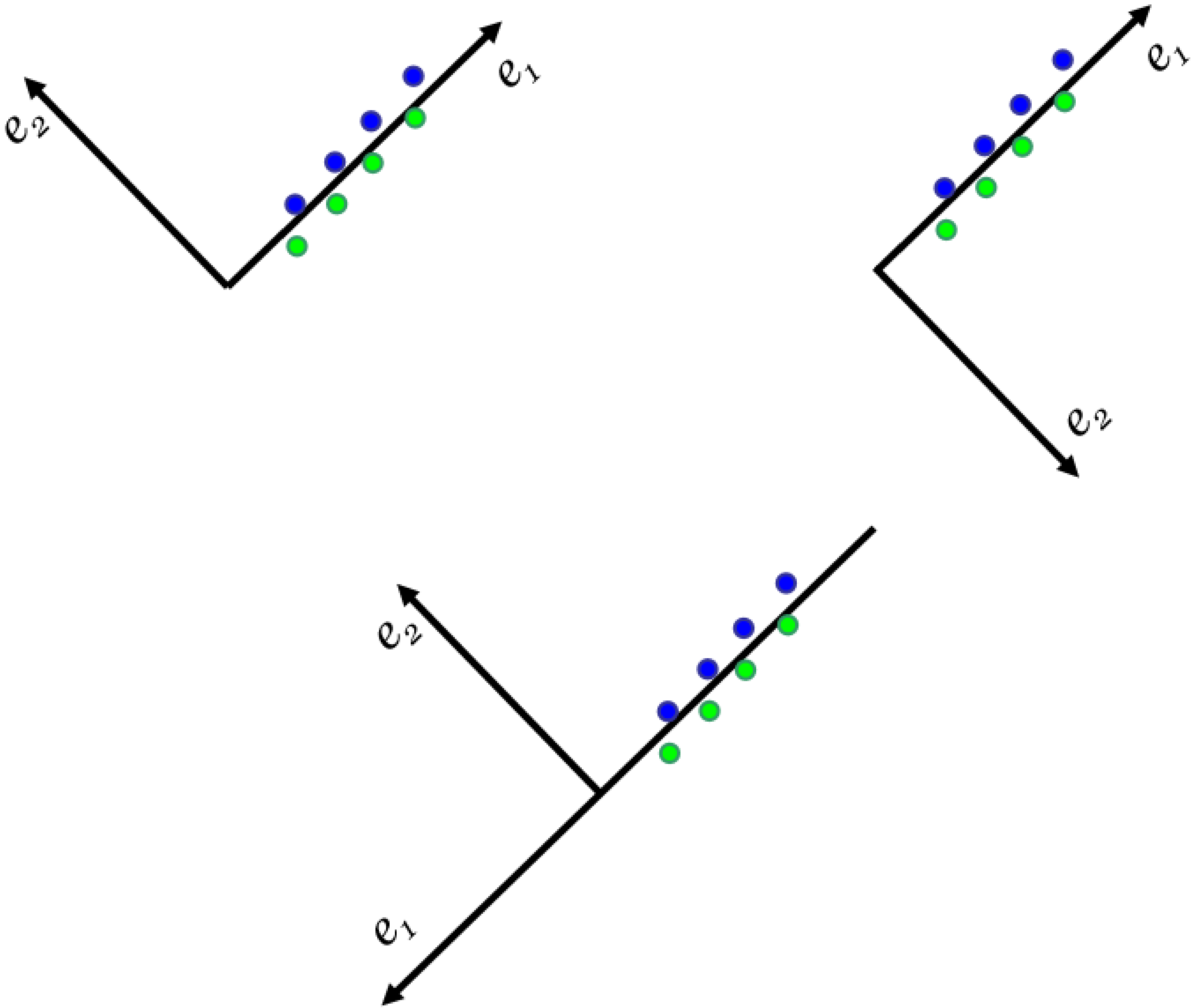
Importance of Principal Components (PC), or Eigenvectors, or Rotations

PC1 preserves more variance than PC2
PC2 preserves more variance than PC3
….

Proportion of Variance Preserved if only *k* PCs are used:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} = \frac{\sum_{i=1}^{k} \lambda_i}{\text{trace}\,(\Sigma)}$$

Eigenvectors are NOT unique



PCA: Covariance vs. Correlation Matrix

- **PCA results depend on the scales at which the variables are measured:**
  - PCA should only be used with the raw data if all variables have the same units of measure

- **PCA results depend on the variances of the variables: the ones with the highest sample variances will tend to be emphasized in the first few principal components:**
  - Use PCA with covariance matrix only if you wish to give variables with higher variances more weight in the analysis

- **If the variables either have different units of measurement (i.e., pounds, feet, gallons, etc), or if we wish each variable to receive equal weight in the analysis, then the variables should be standardized (Z-scores) before a principal components analysis is carried out.**

Matrix Decomposition

PCA: DIMENSION REDUCTION VIA FEATURE EXTRACTION

PC is a weighted linear sum of original features

Extracted Feature → PCA-based Feature Extraction:

$$\boxed{PC = w_1 * f_1 + w_2 * f_2 + \cdots + w_d * f_d}$$

The magnitude of each weight indicates how important the corresponding feature is → it could be used as a ***feature selection*** technique!

```
> options(digits=1)
> pca.model$rotation
               PC1   PC2   PC3  PC4
Sepal.Length  0.36 -0.66  0.58  0.3
Sepal.Width  -0.08 -0.73 -0.60 -0.3
Petal.Length  0.86  0.17 -0.08 -0.5
Petal.Width   0.36  0.08 -0.55  0.8
```

PC1 = 0.86 Petal.Length + 0.36 Petal.Width + 0.36 Sepal.Length – 0.08 Sepal.Width

PC2 = -0.73 Sepal.Width - 0.66 Sepal.Length + 0.17 Petal.Length + 0.08 Petal.Width

Classification of Dimension Reduction (DR) Methods

DR

Feature Selection

Find **subset of original variables**
Eg: Information gain, branch and bound

Feature Extraction

Find a combination of original dimensions
Eg: Multidimensional scaling, Principal component analysis

Motivation for Dimension Reduction

- **Not all the measured variables are important for understanding the underlying “interesting” phenomena – could complicate the process of data analysis**
- **Decrease the computational cost for other data mining tasks:**
  - **Proximity measure calculations: O(*d*) → O(*k*)**
- **Reduce the noise in the data**
- **Improve the accuracy of predictive models**
- **Reduce collinearity among variables/features**

What is Dimensionality Reduction?

Objective: To transform data from a high-dimensional space to a corresponding representation in some low-dimensional space, while "best" preserving the information.

Given dataset with *m* objects:

$$\boldsymbol{X} \in \mathbb{R}^{m \times n} \xrightarrow{\textbf{DR}} \mathrm{Y} \in \mathbb{R}^{m \times p}$$

m dimensions

p dimensions

where ***p* << *n***