Principal Component Analysis (PCA)

Learn How to Manage Data Dimensionality

Without Losing Information

Contents

The Need For Data Reduction

Explosion in information collection techniques and data availability has resulted in data scientists facing a unique requirement – Need for reducing data.

But how can more information be a problem?

Not all data fields are useful.

Variables can be highly correlated or at times redundant Most data analysis algorithms work on columns(variables).

Large number of variables may result in slowing down of the process.

- Does this mean simply removing data? Definitely not.
- Key rule is - Lose data but not information !

What Is High Dimensional Data ?

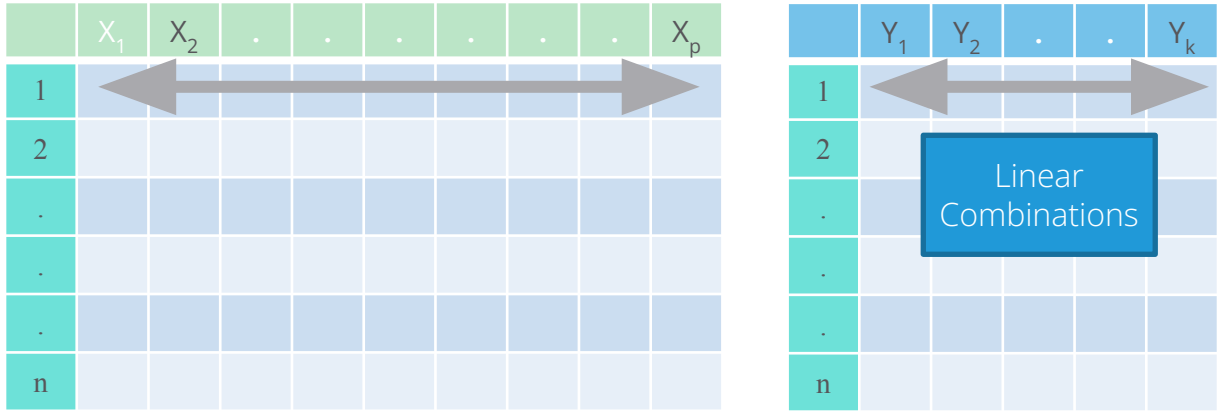
Simply put, high dimensional data can either be data for several observations with several variables or data with several layers of information.

- **Majority of the real world analytics** - in the fields of medical sciences, ecology, biology, finance, etc. - deals with datasets having a few dozens to hundreds and at times thousands of variables. Also the number of cases can go up several million depending upon the industry.

- Consider data maintained by a bank. For each customer, there can be numerous data which can be categorized into transaction data, customer demographics and customer call centre data.
- Further, there can be millions of such customers.

Data Reduction

- Summary of data with p variables by a smaller set of (k) derived variables.
- These k derived variables are **linear combinations of original p variables.**



- In short, **n * p** matrix is **reduced to n * k** matrix.

Variation and Correlation-Key Data Features

- Variation in the data is nothing but the information that the data gives.
- Variables in the data can be highly correlated
- Example: Employee satisfaction survey asks the following questions:

  - Do you think you are rewarded regularly for your work?
  - Do you think the system monitoring your performance is flawed?
  - Do you get timely encouragement from your supervisors regarding your work?

- The above three questions measure the robustness of employee work review. Having three questions to represent a common area can be considered redundant.

- Reduction is justified when there is such Correlation present in the data.

Principal Component Analysis (PCA)

- PCA is probably the most widely-used and well-known of the “Standard” multivariate methods.

  - PCA uses correlation structure of original p variables and derives p linear combinations which are uncorrelated.
  - Each Principal Component provides unique information about the data.
  - Although ‘p' principal components are derived, first 'k' principal components are expected to explain most of the variability in the data.

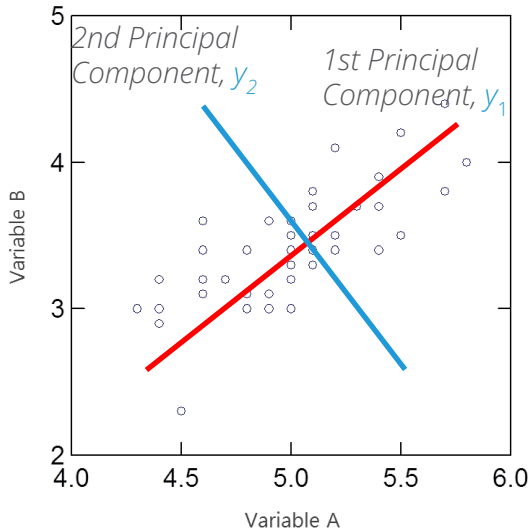
- The method was invented by **Pearson** (1901) and **Hotelling** (1933).
- The method was first applied in ecology by Goodall (1954) under the name “Factor Analysis”. (“Principal Factor Analysis” is a synonym of PCA).

PCA-General Concept

Consider 2 variables A & B (Assume that A & B are features of an object)

A & B have approximately same variance and are highly correlated

Fig. 01



- Now suppose we pass a vector through the scatter points such that it is a good linear fit to the data points. This could be considered a new feature of the object which is a linear combination of A & B.

- We then plot a second line which is perpendicular to the vector, such that both lines pass through the centroid of the data. This becomes our second linear combination.

PCA – General Concept

From p original variables: $x_1, x_2, \ldots, x_p$, derive p new variables $y_1, y_2, \ldots, y_p$

$$
\begin{aligned}
y_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1p}x_p \\
y_2 &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2p}x_p \\
&\vdots \\
y_p &= a_{p1}x_1 + a_{p2}x_2 + \ldots + a_{pp}x_p
\end{aligned}
$$

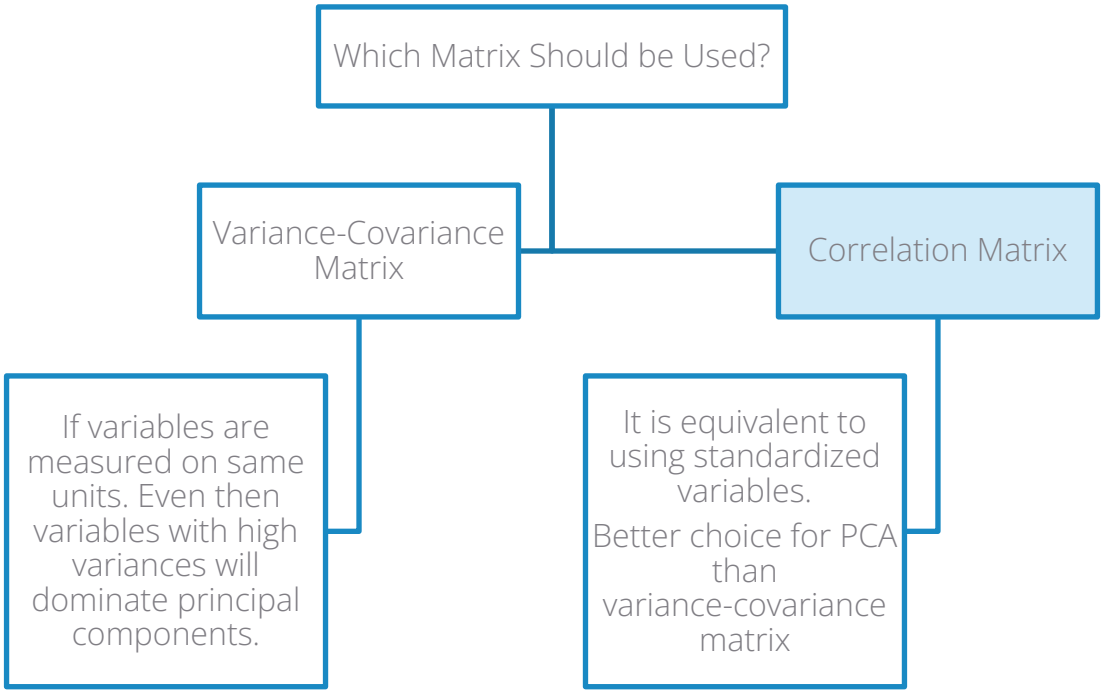
y_k's are Principal Components

Although p principal components are derived, data reduction is achieved with first k principal components.

Properties of Principal Components :

- y_k's are uncorrelated (Orthogonal).
- y_1 explains as much as possible of original variance in data set.
- y_2 explains as much as possible of remaining variance. And so on.

PCA – Choice of Matrix Analysis



* Correlation matrix of original variables = Variance-covariance matrix of standardized variables.

Principal Components – Definition

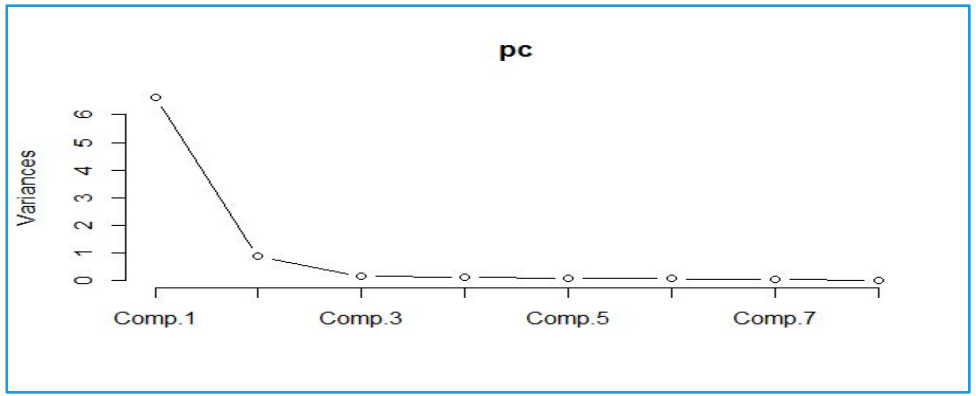
Component	Definition
First Component	A linear combination $a_{11}x_1 + a_{12}x_2 + ... + a_{1p}x_p$ which has maximum variance among all possible linear combinations, subject to $a_{11}^2 + a_{12}^2 + ... + a_{1p}^2 = 1$.
Second Component	A linear combination $a_{21}x_1 + a_{22}x_2 + ... + a_{2p}x_p$ which has maximum variance among all remaining linear combinations, subject to $a_{21}^2 + a_{22}^2 + ... + a_{2p}^2 = 1$ and zero correlation with first principal component.
Third Component *And so on..*	Similarly third Principal Component is obtained which will be uncorrelated with first and second principal components.

How Many Principal Components to Retain

P principal components are derived but data reduction is achieved with first k PC's.

Retain principal components associated with Eigen Value more than '1'.
Variance of PCs is equal to associated Eigen Value. This is known as "**Kaiser Criterion**".

Look at a **Scree plot** and check for "**Elbow**" to determine the correct number of PCs to use.
A scree plot shows how much variation each PC captures from the data. The y axis is eigenvalues, which essentially stand for the variation.
Example of **Scree Plot** below shows that, **first principal component** is **sufficient** to explain variation in the data.



Quick Recap

Data Reduction and PCA	• In the age of big data, data reduction is necessary for analysis • Principal Component Analysis is most popular data reduction method.
PCA – General Approach	• Principal Components are linear combinations of original variables • These components are uncorrelated • Retention of components is based on Eigenvalues.

Principal Component Analysis II

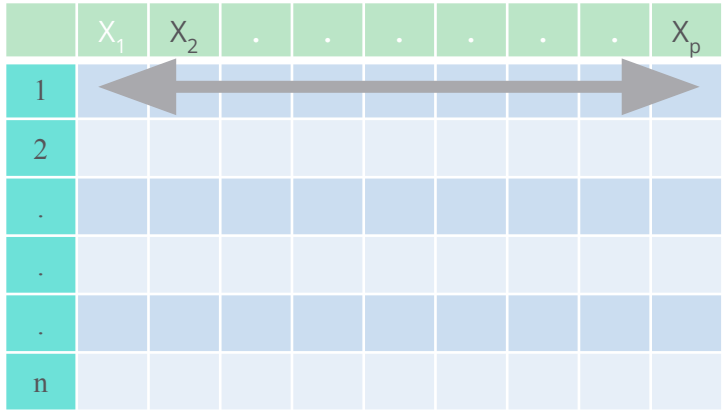
How to Manage Data Dimensionality

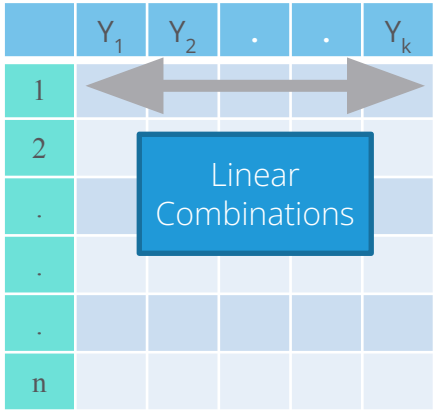
Without Losing Information

Contents

Data Reduction

- Summarization of data with p variables by a smaller set of (k) derived variables.
- These k derived variables are linear combinations of original p variables.





- In short, **n * p** matrix is **reduced to n * k** matrix.

Case Study – Athletics Records

Background

- Data on national athletics records for various countries is available.

Objective

- To achieve data reduction and obtain score for each country which can be used to rank countries based on athletics records.

Available Information

- Data Source: Applied Multivariate Statistical Analysis by Richard A. Johnson , Dean W. Wichern
- Sample size is 55 countries athletics.
- Records for 8 different athletics events – 100 meters to Marathon

Data Snapshot

Athleticsdata

Variables

Observations

Country	100m_s	200m_s	400m_s	800m_min	1500m_min	5000m_min	10000m_min	Marathon_min
Argentina	10.39	20.81	46.84	1.81	3.7	14.04	29.36	137.72
Australia	10.31	20.06	44.84	1.74	3.57	13.28	27.66	128.3

Column	Description	Type	Measurement	Possible Values
Country	Country Name	Categorical	-	-
100m_s	Time for 100 meter running	Continuous	Seconds	Positive Values
200m_s	Time for 200 meter running	Continuous	Seconds	Positive Values
400m_s	Time for 400 meter running	Continuous	Seconds	Positive Values
800m_min	Time for 800 meter running	Continuous	Minutes	Positive Values
1500m_min	Time for 1500 meter running	Continuous	Minutes	Positive Values
5000m_min	Time for 5000 meter running	Continuous	Minutes	Positive Values
10000m_min	Time for 10000 meter running	Continuous	Minutes	Positive Values
Marathon_min	Time for Marathon running	Continuous	Minutes	Positive Values

PCA in R

```
#Import the data
data<-read.csv("Athleticsdata.csv", header=TRUE)

athletics<-subset(data,select=c(-Country))

pc<-princomp(formula=~.,data=athletics,cor=T)

summary(pc)
```

- **subset()** is used to remove the variable “Country” from the data.
- **princomp()** from base R performs PCA on the given numeric data matrix.
- **formula=** contains the numeric variables.
  ~. ensures all numeric variables are taken.
- **cor=T** indicates that calculations should be done using the Correlation Matrix. It is equivalent to standardization.

PCA in R

```
# Output:
```

```
Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4     Comp.5      Comp.6     Comp.7      Comp.8
Standard deviation     2.5740680 0.9355011 0.39820722 0.3521954 0.28286280 0.260301726 0.21484785 0.149909664
Proportion of Variance 0.8282283 0.1093953 0.01982112 0.0155052 0.01000142 0.008469624 0.00576995 0.002809113
Cumulative Proportion  0.8282283 0.9376236 0.95744470 0.9729499 0.98295131 0.991420937 0.99719089 1.000000000
```

Interpretation:

- The summary function on object pc gives **std. deviation, proportion of variance and cumulative proportion**.

- First Principal Component explains 83% of the variation. Note that 8 PC's are derived using 8 variables but first PC explains most of the variation.

PCA in R –Matrix of Loadings

```
# Component Loadings
pc$loadings
```

- **loadings** are coefficients in linear combinations
- The first column under Comp.1 gives coefficients for first principal component

Output:

```
> pc$loadings

Loadings:
             Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
X100m_s       0.318  0.565  0.326  0.129  0.267  0.590  0.154  0.113
X200m_s       0.337  0.462  0.369 -0.257 -0.157 -0.648 -0.128 -0.102
X400m_s       0.356  0.249 -0.561  0.650 -0.221 -0.158
X800m_min     0.369        -0.531 -0.482  0.540        -0.237
X1500m_min    0.373 -0.140 -0.155 -0.407 -0.491  0.143  0.608  0.143
X5000m_min    0.364 -0.312  0.190        -0.250  0.155 -0.593  0.543
X10000m_min   0.367 -0.307  0.182        -0.128  0.232 -0.165 -0.796
Marathon_min  0.342 -0.440  0.260  0.300  0.493 -0.329  0.393  0.160

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.125  0.125  0.125  0.125  0.125  0.125  0.125  0.125
Cumulative Var  0.125  0.250  0.375  0.500  0.625  0.750  0.875  1.000
```

Interpretation:

- First Principal Component can be interpreted as ‘general athletics skill’ since all variables have similar loadings.

Deriving Scores Using PCA

```
# Adding PCA scores to original data as a new variable:
data$performance<-pc$score[,1]
head(data)
# Output:
```

```
> data$performance<-pc$score[,1]
> head(data)
    Country X100m_s X200m_s X400m_s X800m_min X1500m_min X5000m_min X10000m_min Marathon_min performance
1 Argentina   10.39   20.81   46.84      1.81       3.70      14.04       29.36       137.72   0.2656535
2 Australia   10.31   20.06   44.84      1.74       3.57      13.28       27.66       128.30  -2.4669681
3   Austria   10.44   20.81   46.82      1.79       3.60      13.26       27.72       135.90  -0.8134149
4   Belgium   10.34   20.68   45.04      1.73       3.60      13.22       27.45       129.95  -2.0582394
5   Bermuda   10.28   20.58   45.91      1.80       3.75      14.68       30.55       146.62   0.7471461
6    Brazil   10.22   20.43   45.21      1.73       3.66      13.62       28.62       133.13  -1.5710562
```

Interpretation:

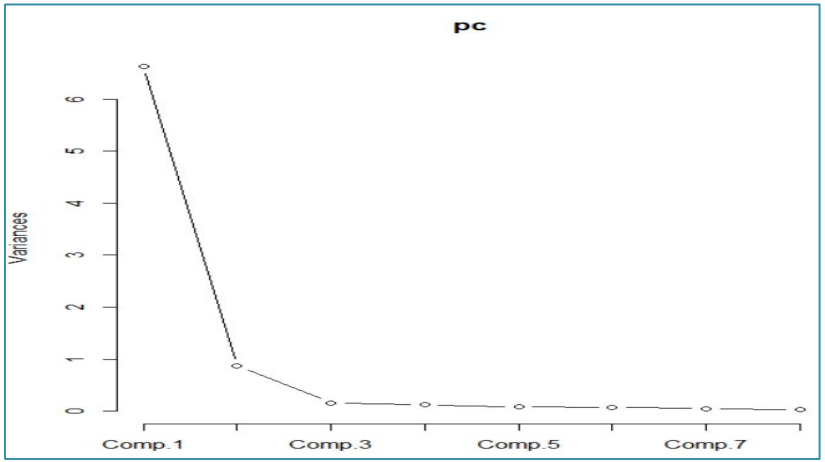
- New column ‘performance’ stores calculated scores using first Principal Component.
- Lower score implies lesser time and hence better athletics performance.

PCA in R - Scree Plot

```
# Scree Plot
plot(pc, type="lines")
```

plot() generates a scree plot

```
# Output:
```



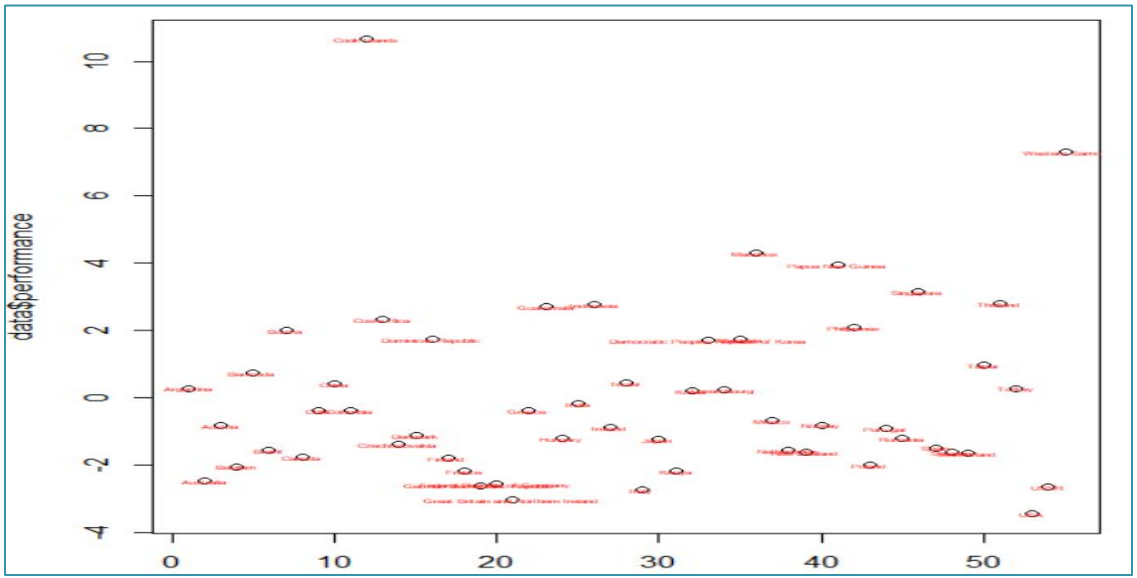
Interpretation:
First Principal Component is sufficient in explaining most of the variation.

```
# Plot of country wise performance
plot(data$performance)
text(data$performance,label=data$Country,col="red",cex=0.4)
```

text() is used to assign names to each points in the plot

PCA in R –Plot of “Performance”

```
# Output-plot of country wise performance plot:
```



Interpretation:

- Athletics from country Cook islands and Western Samoa are performing low since, their score are highest.(lower the score ,better is the performance).

Which are bottom 3 countries?

```
# head function gives countries with highest “performance”
# In our context, these are bottom 3 countries

top3<-head(data[order(-data$performance),],3)
top3

# Output :
```

	Country	X100m_s	X200m_s	X400m_s	X800m_min	X1500m_min	X5000m_min	X10000m_min	Marathon_min	performance
12	Cook Isands	12.18	23.20	52.94	2.02	4.24	16.70	35.38	164.70	10.653867
55	Western Samoa	10.82	21.86	49.00	2.02	4.24	16.28	34.71	161.83	7.297965
36	Mauritius	11.19	22.45	47.70	1.88	3.83	15.06	31.77	152.23	4.299192

Which are top 3 countries?

```
# tail function gives top 3 countries
bottom3<-tail(data[order(-data$performance),],3)
bottom3
```

```
# Output :
```

	Country	X100m_s	X200m_s	X400m_s	X800m_min	X1500m_min	X5000m_min	X10000m_min	Marathon_min	performance
29	Italy	10.01	19.72	45.26	1.73	3.60	13.23	27.52	131.08	-2.750446
21	Great Britain and Northern Ireland	10.11	20.21	44.93	1.70	3.51	13.01	27.51	129.13	-3.050287
53	USA	9.93	19.75	43.86	1.73	3.53	13.20	27.43	128.22	-3.460450

Interpretation:

- USA, Britain and Italy are the top three performing countries.
- Cook Islands, Western Samoa and Mauritius are the bottom three countries.

Principal Components Are Uncorrelated

```
# Correlation Matrix of principal components
round(cor(pc$scores))
```

round(cor()) calculates rounded correlations of the PCA scores.

```
# Output:
```

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7	Comp.8
Comp.1	1	0	0	0	0	0	0	0
Comp.2	0	1	0	0	0	0	0	0
Comp.3	0	0	1	0	0	0	0	0
Comp.4	0	0	0	1	0	0	0	0
Comp.5	0	0	0	0	1	0	0	0
Comp.6	0	0	0	0	0	1	0	0
Comp.7	0	0	0	0	0	0	1	0
Comp.8	0	0	0	0	0	0	0	1

Interpretation:

- Correlation matrix shows that, principal components are uncorrelated. Diagonal 1's are the correlation of component to itself.

Quick Recap

Data Reduction and PCA	• PCA reduces n * p matrix to n * k where k is smaller than p
PCA in R	• **`princomp()`** function in base R performs PCA. • Loadings from the summary output are used to derive new variables

Principal Component Regression

Contents

Multiple Linear Regression: Statistical Model

$$Y = b_0 + b_1X_1 + b_2X_2 + \ \ldots + b_pX_p + e$$

Where,

- Y : Dependent Variable
- $X_1, X_2, \ldots, X_p$: Independent Variables
- $b_0, b_1, \ldots, b_p$: Parameters of Model
- e : Random Error Component

Independent variables can either be **Continuous or Categorical**

Problem of Multicolinearity

Multicolinearity exists ▶ if there is a **strong linear relationship among independent variables.**

Consequences

- Highly Unstable Model Parameters
  As standard errors of their estimates are inflated
- Model Fails to predict accurately out of sample Data

Multicolinearity is detected using Variance Inflation Factor, VIF

$$\text{Tolerance} = 1 - R_i^2$$

$$\text{VIF} = 1/\text{Tolerance}$$

where R_i^2 (R Squared) is obtained using regression of Xi on other independent variables

Any VIF > 5, indicates presence of multicollinearity

Multicollinearity – Remedial Measures

The problem of Multicollinearity can be solved by different approaches:

- Drop one of the independent variables, which is explained by others
- Use Principal Component Regression in case of severe Multicollinearity
- Use Ridge Regression

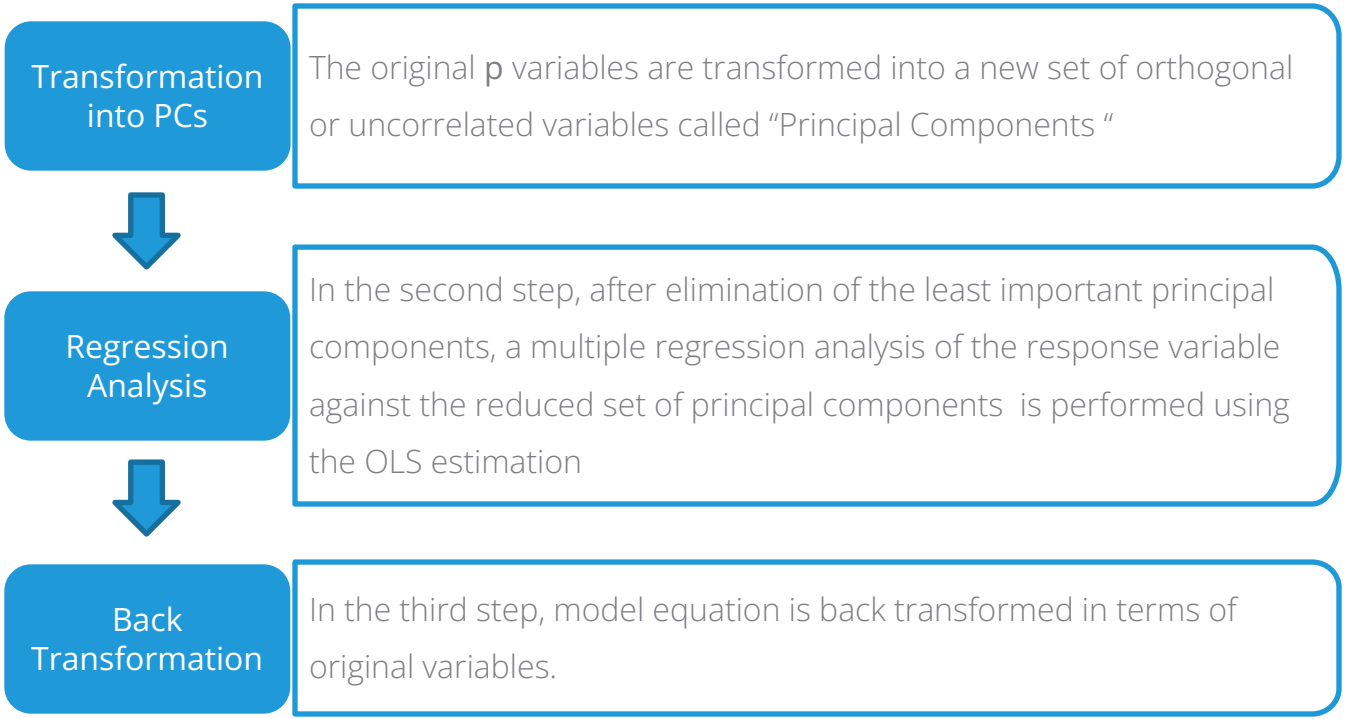
Principal Component Regression

In Principal Component Regression,

> First k principal components are used as independent variables instead of original X variables

- Each PC is a linear combination of all X variables
- Final model is expressed in terms of original independent variables for ease of interpretation

Principal Component Regression



PCR-Statistical Model

Model in terms of original X variables:

$$Y = b_0 + b_1x_1 + b_2x_2 + \cdots + b_px_p + e$$

Model in terms of Principal Components:

$$Y = a_0 + a_1PC_1 + a_2PC_2 + \cdots + a_kPC_k + e'$$

Case Study

Background

- A company periodically records data for sales and expenses. The company wishes to model the relationship between its sales and sales related expenses and obtain predictions

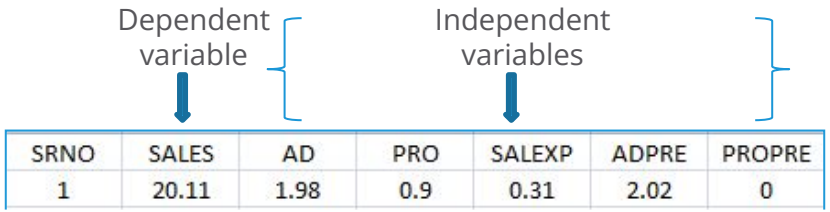
Objective

- To predict incremental sales based on planned sales related expenses

Available Information

- Data available for 143 micro business zones
- **Sales** is the Dependent Variable
- **Expenditure towards advertisements and promotions in the current and previous months** are Predictors

Data Snapshot



Observations

Columns	Description	Type	Measurement	Possible values
SRNO	Serial Number	-	-	Intergers
SALES	Incremental Sales	Numerical	Euros Million	positive value
AD	Current Advertising Expenses	Numerical	Euros Million	positive value
PRO	Current Promotional Expenses	Numerical	Euros Million	positive value
SALEXP	Misc. Sales Expenses	Numerical	Euros Million	positive value
ADPRE	Previous Period's Advertising Expenses	Numerical	Euros Million	positive values
PROPRE	Previous Period's Promotional Expenses	Numerical	Euros Million	Positive value

PCR in R

```
# Import csv file “pcrdata”
salesdata<-read.csv("pcrdata.csv",header=T)
# Fitting a Linear Model :
predsales<-lm(SALES~AD+PRO+SALEXP+ADPRE+PROPRE,data=salesdata)

summary(predsales)
```

- **lm()** fits a linear regression model.
- **summary()** generates model summary.

```
# Output of summary
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -10.8147     6.5314  -1.656  0.10005
AD            4.6762     1.4100   3.316  0.00117 **
PRO           7.7886     1.2628   6.168  7.3e-09 ***
SALEXP       22.4089     0.7704  29.089  < 2e-16 ***
ADPRE         3.1856     1.2442   2.560  0.01154 *
PROPRE        3.4970     1.3697   2.553  0.01177 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.201 on 137 degrees of freedom
Multiple R-squared:  0.9089,    Adjusted R-squared:  0.9055
F-statistic: 273.2 on 5 and 137 DF,  p-value: < 2.2e-16
```

Interpretation: Multiple R-Squared is 0.9089, showing model to be a good fit.

PCR in R

```
# Checking for Multicolinearity
install.packages("car")
library(car)

vif(predsales)
```

vif() in package **car** calculates VIFs.

```
# Output of VIF
```

AD	PRO	SALEXP	ADPRE	PROPRE
36.159771	31.846727	1.076284	24.781948	42.346468

Interpretation:

- VIF values are very high (>5, except for SALEEXP) indicating severe multicolinearity problem.

PCR in R

```
# PCA in R
# Subsetting data for getting Principal components and performing PCA

salesdatapca<-subset(salesdata,select=c(-SRNO,-SALES))

pc<-princomp(formula=~.,data=salesdatapca, cor=T)
summary(pc)
```

- **princomp()**from base R performs PCA on the given numeric data matrix
- **formula=** contains the numeric variables. ~. ensures all variables are taken
- **cor=T** indicates that calculations should be done using the Correlation Matrix.
- **summary()** generates the summary of PCA

PCR in R

```
# Output
```

```
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4
Standard deviation     1.3015565 1.1318477 1.0705353 0.9334328
Proportion of Variance 0.3388099 0.2562159 0.2292091 0.1742593
Cumulative Proportion  0.3388099 0.5950257 0.8242349 0.9984942
                           Comp.5
Standard deviation     0.086769725
Proportion of Variance 0.001505797
Cumulative Proportion  1.000000000
```

Interpretation:

- The first three principal components explain 82% of the variation in the data. Therefore, we can use 3 PC's in Regression Model

PCR in R

```
# PCR in R
install.packages("pls")
library(pls)
pcmodel<-pcr(SALES~AD+PRO+SALEXP+ADPRE+PROPRE,ncomp=3,data=salesdata,sca
le=TRUE)

salesdata$pred_pcr<-predict(pcmodel,salesdata,ncomp=3)
head(salesdata)
```

Install and load package **pls** (Partial Least Squares).

pcr() in package **pls** performs Principal Component Regression
ncomp=3 is the number of components to be included in the model
scale=TRUE indicates X is scaled by dividing each variable by its standard deviation. This ensures data is standardised before running the PCR algorithm.

predict() is used to get predictions by PCR.

PCR in R

```
# Output
```

	SRNO	SALES	AD	PRO	SALEXP	ADPRE	PROPRE	pred_pcr
1	1	20.11	1.98	0.9	0.31	2.02	0.0	21.29053
2	2	15.10	1.94	0.0	0.30	1.99	1.0	18.16976
3	3	18.68	2.20	0.8	0.35	1.93	0.0	21.27149
4	4	16.05	2.00	0.0	0.35	2.20	0.8	17.62114
5	5	21.30	1.69	1.3	0.30	2.00	0.0	22.97930
6	6	17.85	1.74	0.3	0.32	1.69	1.3	20.57217

pred_pcr column gives predicted values of SALES using PCR.

Comparing Linear Regression Model and PCR model on Test data

```
# Importing Test Data
salesdata_test<-read.csv("pcrdata_test.csv",header=TRUE)

# Getting RMSE of linear regression model
salesdata_test$lmpredict<-predict(predsales,salesdata_test)
salesdata_test$lmres<-(salesdata_test$SALES-salesdata_test$lmpredict)
RMSE_lm<-sqrt(mean(salesdata_test$lmres)**2)

# Getting RMSE of PCR model
salesdata_test$pcrpredict<-predict(pcmodel,salesdata_test,ncomp=3)
salesdata_test$pcrres<-(salesdata_test$SALES-salesdata_test$pcrpredict
)
RMSE_pcr<-sqrt(mean(salesdata_test$pcrres)**2)
```

predict () will give the predicted value for the model.

Comparing Linear Regression Model and PCR model on Test data

```
# Viewing data after adding predicted & residual variables

head(salesdata_test)
```

```
# Output
  SRNO SALES   AD  PRO SALEXP ADPRE PROPRE lmpredict       lmres pcrpredict     pcrres
1    1 28.93 2.75 1.00   0.72  1.97   0.02  32.31368  -3.3836776   23.23291  5.6970943
2    2 25.96 1.73 1.06   0.89  2.77   0.02  34.36925  -8.4092464   22.26693  3.6930660
3    3 31.25 2.19 1.26   0.79  1.22   0.42  32.29821  -1.0482117   27.61578  3.6342207
4    4 25.05 1.82 1.45   0.83  2.23   0.15  35.21751 -10.1675083   25.21307 -0.1630736
5    5 27.32 2.38 1.01   0.74  1.01   0.07  28.22616  -0.9061594   27.05439  0.2656139
6    6 23.23 2.97 0.46   0.96  2.36   0.12  36.10681 -12.8768143   20.92296  2.3070370
```

```
RMSE_lm
[1] 7.949631
RMSE_pcr
[1] 0.1121959
```

Interpretation:

- **RMSE using PCR is less than RMSE using linear regression**, we may conclude that PCR model predicts SALES better than linear regression model when multicolinearity exists.

Quick Recap

Multiple Linear Regression and Multicollinearity	• Highly correlated predictor variables is a very frequent phenomenon in real world analytics.
Principal Component Regression	• PCR is a three way process where the variables are first transformed to principal components, regression is run by considering these components as regressors and finally, they are transformed back to their original forms.
PCR in R	• **`pcr()`** function in package **`pls`** performs PCR.

Factor Analysis (PCR)

Extracting Hidden Factors in Multivariate Data

Contents

Factor Analysis

- Factor Analysis is primarily used for,

Data Reduction	or	Structure Detection
The purpose of data reduction is to replace original variables with a smaller number of uncorrelated variables		The purpose of structure detection is **to examine the underlying (or latent) relationships between the variables**

- It is an interdependence technique: **No distinction between dependent and independent variables** (Also called as Unsupervised Method)

Statistical Model

- Each variable is expressed as a linear combination of factors
- The factors are some common factors plus a unique factor

The factor model is represented as:

$$X_i = l_{i1}F_1 + l_{i2}F_2 + l_{i3}F_3 + - - - - + l_{im}F_m + u_i + e_i$$

where,

l_{ij}	:	Loading of variable i on common factor j
F_j	:	Common factor j
u_i	:	Mean of variable i
m	:	Number of common factors
e_i	:	Specific factor

The model looks similar to linear regression but **factors are unobservable**

Statistical Model – Five Variables and Two Common Factors

- Each variable is expressed as a linear combination of **two** factors

  The factor model is represented as:

$$X_1 - u_1 = l_{11}F_1 + l_{12}F_2 + e_1$$
$$X_2 - u_2 = l_{21}F_1 + l_{22}F_2 + e_2$$
$$X_3 - u_3 = l_{31}F_1 + l_{32}F_2 + e_3$$
$$X_4 - u_4 = l_{41}F_1 + l_{42}F_2 + e_4$$
$$X_5 - u_5 = l_{51}F_1 + l_{52}F_2 + e_5$$

- F_1 and F_2 are common factors
- e's are specific factors
- Each variable loads on two common factors

Statistical Model – Five Variables and Two Common Factors

- The common factors themselves can be expressed as linear combinations of the observed variables.

$$F_i = W_{i1}X_1 + W_{i2}X_2 + W_{i3}X_3 + \ldots + W_{ik}X_k$$

where,

F_i : Estimated score of i^{th} factor
W_i : Weight or factor score coefficient
k : Number of variables

Common Factor and Unique Factor

Common Factor

- It is an abstraction, a **hypothetical construct that affects some or all observed variables**
- We want to estimate the common factors that contribute to the variance in our variables
- Example: Athletics performance can be thought of as combination of 'speed' and ‘stamina'

Unique Factor

- It is a **factor that contributes to the variance of only one variable**
- We want to exclude these unique factors from our solution
- The unique factors are unrelated to one another and unrelated to the common factors

Factor Analysis – Assumptions

The unobservable random variables F_j and e_i are assumed to satisfy following conditions:

1. F_j and e_i are independent

1. $E(F_j)=0$, $Cov(F_j, F_k)=0$ for $j \neq k$ and $V(F_j)=1$

1. $E(e_i)=0$ and $Cov(e_j, e_k)=0$ for $j \neq k$

When these assumptions are satisfied, the model is called Orthogonal Factor Model

Estimation of Factor Loadings

Typically 2 methods are used to estimate the factor loadings:

1. Principal Component Method
2. Maximum Likelihood Method

   (Assuming common Factors and specific factors follow Normal distributions)

- The number of factors to be included in the model is determined by considering the proportion of variance explained by the *m* factors model
- The portion of the variance of i^{th} variable contributed by m common factors is called as "**Communality**"
- The portion due to specific factor is called as "**Uniqueness**" or "**Specific Variance**"

Interpretation of Factor Loadings

A factor can be interpreted in terms of the variables that load high on it

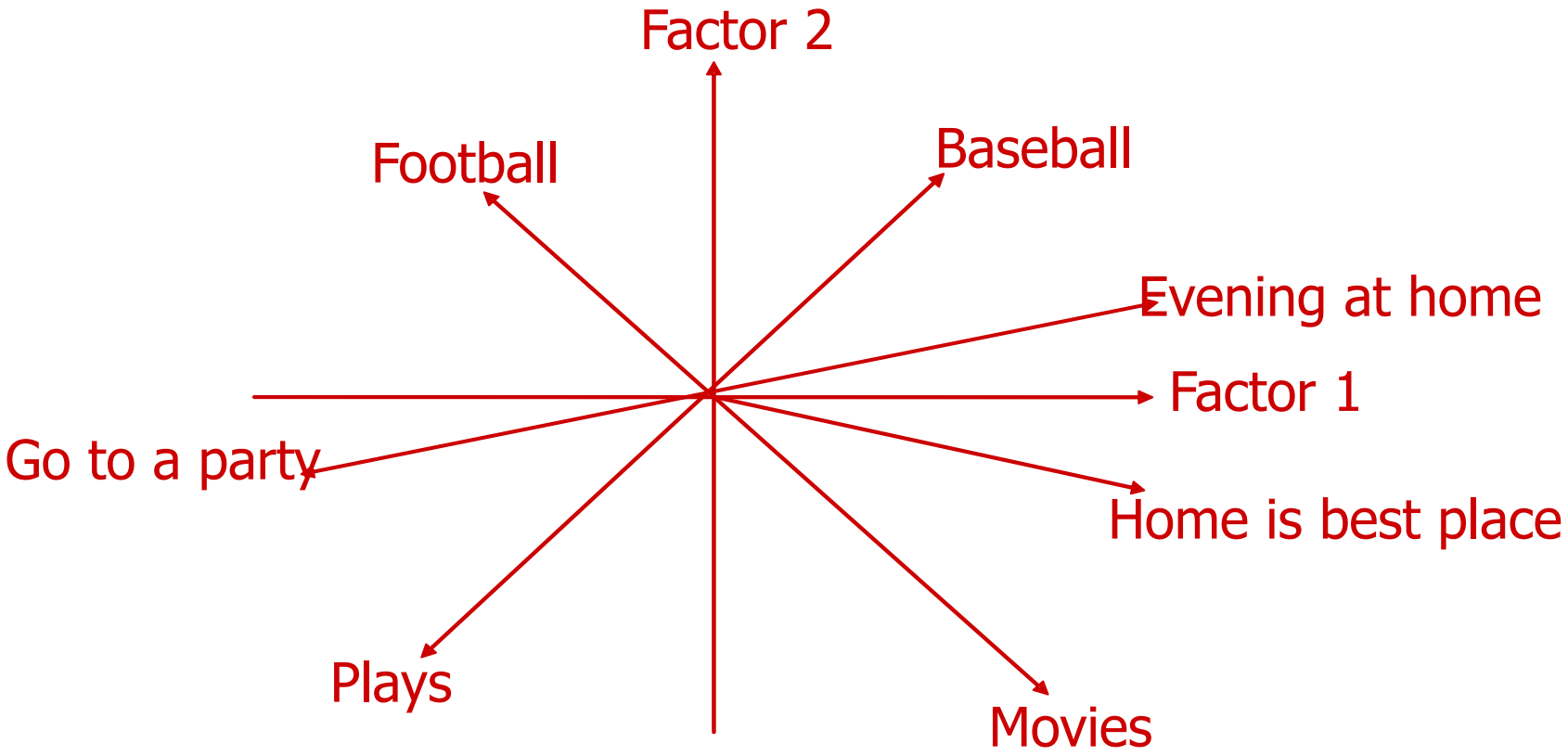
- **Factor loadings are correlations between the variables and the factors**. So, when factor loadings are high on a variable or a group of variable, it means that the factor represents the group of variables.

- Another useful aid in interpretation is to **plot the variables, using the factor loadings as coordinates**. Variables at the end of an axis are those that have high loadings on only that factor, and hence describe the factor.

Ideally we should like to see a pattern of loading such that each variable loads highly on a single factor and has small to moderate loading on the remaining factors.

Interpretation of Factor Loadings

Factors Underlying Selected Psychographics and Lifestyles



Rotation of Factors

- Factor rotation is simply changing the “viewing angle” of the factor space
- Through rotation the loading matrix is transformed into a simpler one that is easier to interpret
- After rotation:

Factors can be interpreted if	⇨	**Each variable has high loading on only one factor and moderate/low loadings on other factors.**

- The rotation is called Orthogonal Rotation if the axes are maintained at right angles

Rotation of Factors

Varimax Procedure:

- Axes maintained at right angles
- An orthogonal method of rotation that minimizes the number of variables with high loadings on a factor
- Orthogonal rotation results in uncorrelated factors

Factor Scores

The factor scores for the i^{th} factor may be estimated as follows:

$$F_i = W_{i1}X_1 + W_{i2}X_2 + W_{i3}X_3 + \ldots + W_{ik}X_k$$

- Intuitively, **factors are latent variables that underlie the scores in observed variables**. Factor scores would therefore represent the score of each person on the underlying latent variable.
- The **interpretation of each of these factors is based on the content of the original variables** so that each factor is interpreted as whatever the attributes with high loadings for this particular factor have in common.

Case Study – Athletics Records

Background

- Data for various countries' national athletics records is available.

Objective

- To discover hidden factors which can be interpreted as different skills required by athletes.

Available Information

- Data Source: Applied Multivariate Statistical Analysis by Richard A. Johnson , Dean W. Wichern
- Sample size is 55 countries athletics.
- Records for 8 different categories (activity – running) are available. Categories differ on the basis of length of tracks.

Data

Athleticsdata

Variables

Observations

Country	100m_s	200m_s	400m_s	800m_min	1500m_min	5000m_min	10000m_min	Marathon_min
Argentina	10.39	20.81	46.84	1.81	3.7	14.04	29.36	137.72
Australia	10.31	20.06	44.84	1.74	3.57	13.28	27.66	128.3

Column	Description	Type	Measurement	Possible Values
Country	Country Name	Categorical		
100m_s	Time for 100 meter running	Continuous	Seconds	Positive Values
200m_s	Time for 200 meter running	Continuous	Seconds	Positive Values
400m_s	Time for 400 meter running	Continuous	Seconds	Positive Values
800m_min	Time for 800 meter running	Continuous	Minutes	Positive Values
1500m_min	Time for 1500 meter running	Continuous	Minutes	Positive Values
5000m_min	Time for 5000 meter running	Continuous	Minutes	Positive Values
10000m_min	Time for 10000 meter running	Continuous	Minutes	Positive Values
Marathon_min	Time for Marathon running	Continuous	Minutes	Positive Values

Factor Analysis in R

```
# Import the data
data<-read.csv("Athleticsdata.csv", header=TRUE)
# Perform factor analysis
athletics<-subset(data,select=c(-Country))
fact<-factanal(athletics,2,rotation="varimax",scores="regression")
fact
```

- **subset()** is used to remove the column named “Country” from the data.
- **factanal()** from base R performs factor analysis on the given numeric data matrix.
- **rotation=“varimax”** performs varimax rotation of loading matrix
- **scores=**“regression” is used to specify method for factor scores

Factor Analysis in R

```
# Output
```

```
Call:
factanal(x = athletics2, factors = 2, scores = "regression",     rotation = "vari
max")

Uniquenesses:
     X100m_s      X200m_s      X400m_s     X800m_min    X1500m_min    X5000m_min
       0.079        0.077        0.151         0.135         0.082         0.034
 X10000m_min Marathon_min
       0.018        0.086

Loadings:
             Factor1 Factor2
X100m_s      0.287   0.916
X200m_s      0.376   0.885
X400m_s      0.537   0.749
X800m_min    0.686   0.628
X1500m_min   0.795   0.535
X5000m_min   0.898   0.400
X10000m_min  0.904   0.406
Marathon_min 0.913   0.284

                Factor1 Factor2
SS loadings       4.071   3.268
Proportion Var    0.509   0.408
Cumulative Var    0.509   0.917

Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 16.65 on 13 degrees of freedom.
The p-value is 0.216
> |
```

Interpretation:

- Two factors explain 92% of common variance.
- Factor 1 can be termed as ‘Stamina’ and factor 2 as ‘Speed’

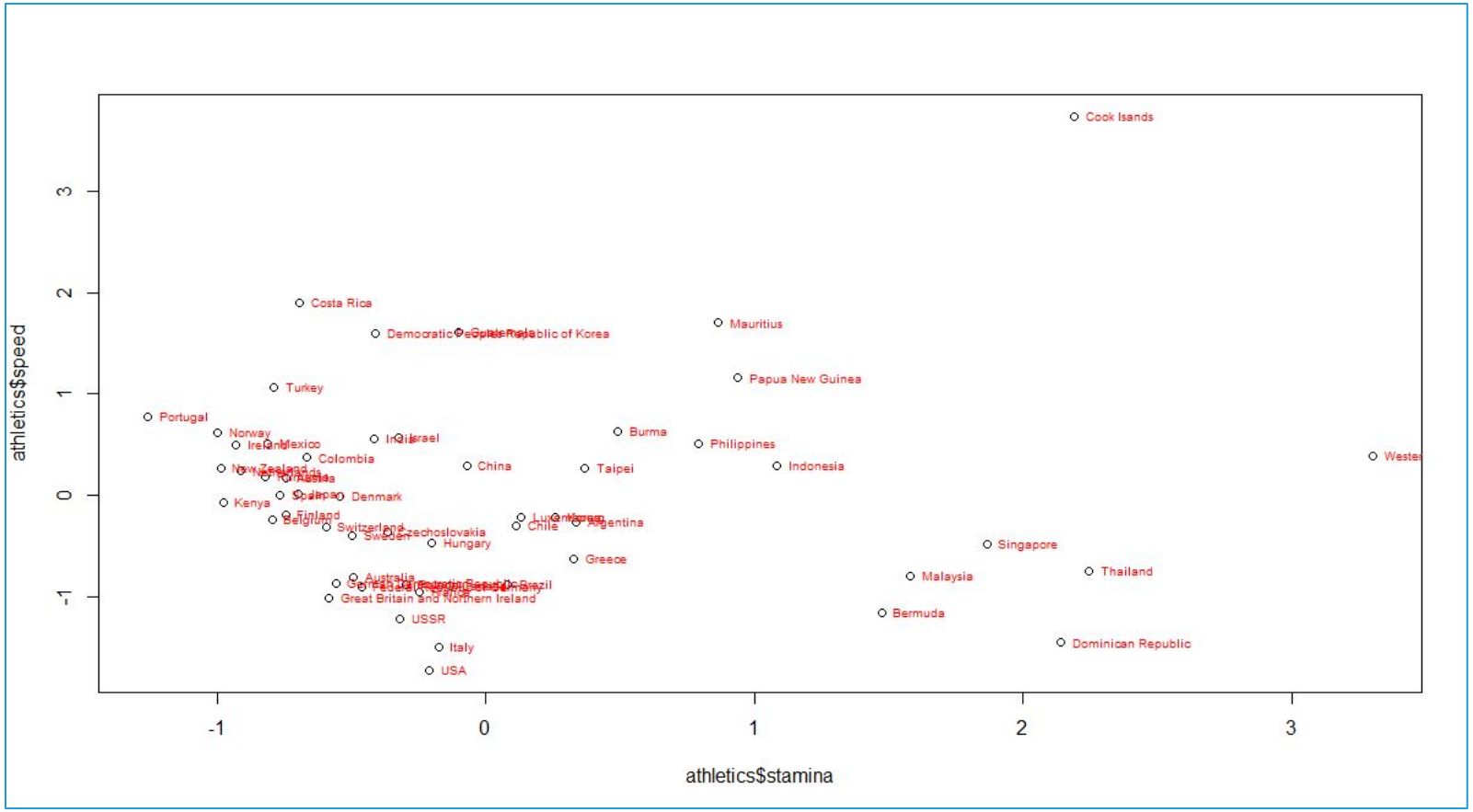
Factor Analysis in R – Factor Scores

```
# Adding Factor Scores to main data & plotting it
data$stamina<-fact$score[,1]
data$speed<-fact$score[,2]
plot(data$stamina,data$speed)
text(data$stamina,data$speed,data$Country,cex=0.6, pos=4,col="red")
```

- Factor scores are stored in the data frame ‘athletics’
- **text()** is used to assign names to each points in the scatter diagram.

Factor Analysis in R

```
# Output
```



Interpretation:
The points represent the scores on factor 1 and factor 2 for each country.

Quick Recap

In this session, we learnt about **Factor Analysis:**

Factor Analysis – Basics and Assumptions	• Each variable is expressed as a linear combination of factors. Some are common factors and one is unique.
	• The unobserved random variables
	  • Fj and ei are independent
	  • E(Fj)=0, Cov(Fj, Fk)=0 for j≠k and V(Fj)=1
	  • E(ei)=0 and Cov(e j, e k)=0 for j≠k

Estimation and Interpretation and Rotation of Factors	• Factors can be estimated by Principal Component Method or Maximum Likelihood Method • Factor loadings are correlations between the variables and the factors, high loadings on a group of variables suggests that the factor represents those variables • Rotation means changing the "viewing angle" of the factor space. The Varimax Procedure is the most common method of factor rotations
Factor Analysis in R	• **`factanal()`** function performs maximum likelihood factor analysis

Non-Hierarchical Clustering

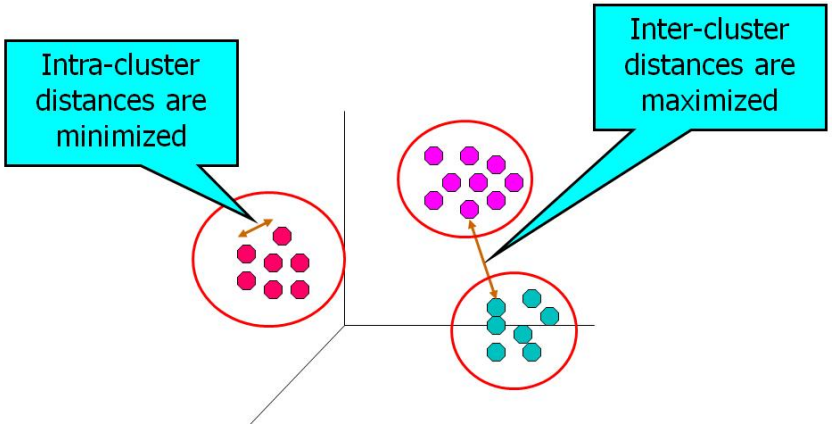
K Means Method

Contents

Cluster Analysis

Cluster analysis is a class of statistical techniques that can be used to classify objects or cases into groups called **Clusters.**

- A cluster is a group of relatively homogeneous cases or observations.
- The observations are dissimilar to objects outside the cluster, particularly objects in other clusters.



- Cluster Analysis is one of the unsupervised learning method.

K-Means Clustering

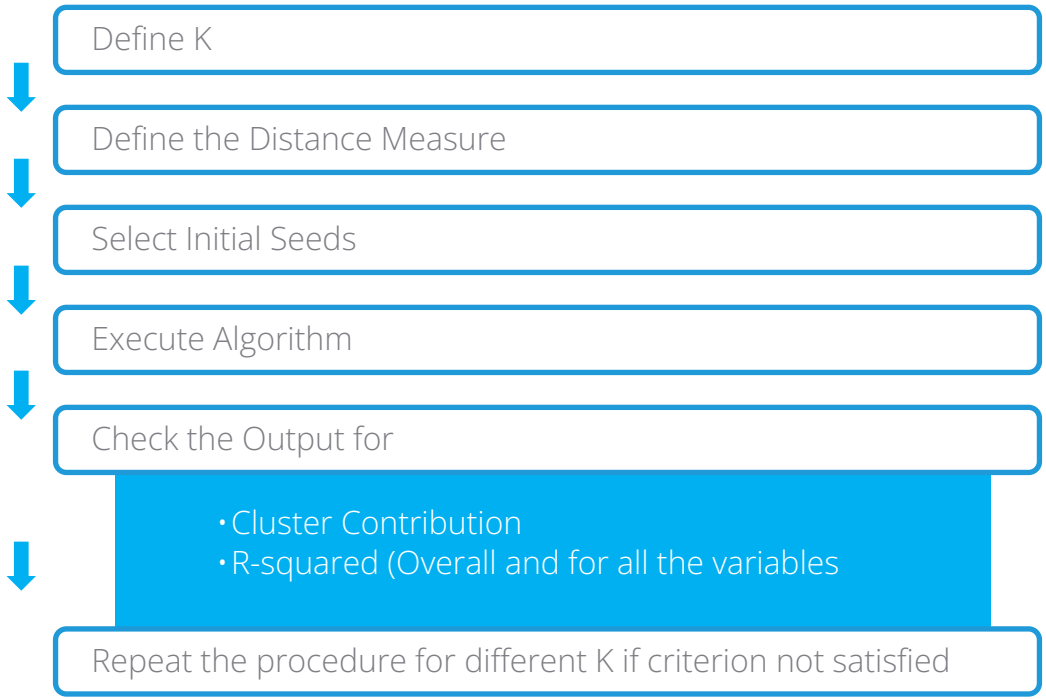
- K-Means Clustering is one of the most popular non-hierarchical clustering methods

- K –Means method is suitable for large data sets and widely used for customer segmentation in BFSI or retail domains

- The number of clusters (k) must be known a priori
  (Though in reality this may not be the case)

- Alternatively, cluster solutions can be observed for different k and evaluated to get the best possible cluster solution

K-Means Clustering – Steps



Distance Measures

- Clustering algorithms require a **mathematical measure to assess the similarity of a pair of observations or clusters**

Object	X1	X2			XP
1	a1	a2			ap
2	b1	b2			bp

- Manhattan Distance

The sum of the absolute differences in values of P variables

$$d(x , y) = | a1-b1 | + | a2-b2 | + ... + | ap-bp |$$

- Chebyshev Distance

The maximum absolute difference in values of P variables

$$d(x , y) = Max (| ai-bi |)$$

Distance Measures – Euclidean Distance

- **Squared Euclidean Distance: The** sum of squared differences between values of each variable

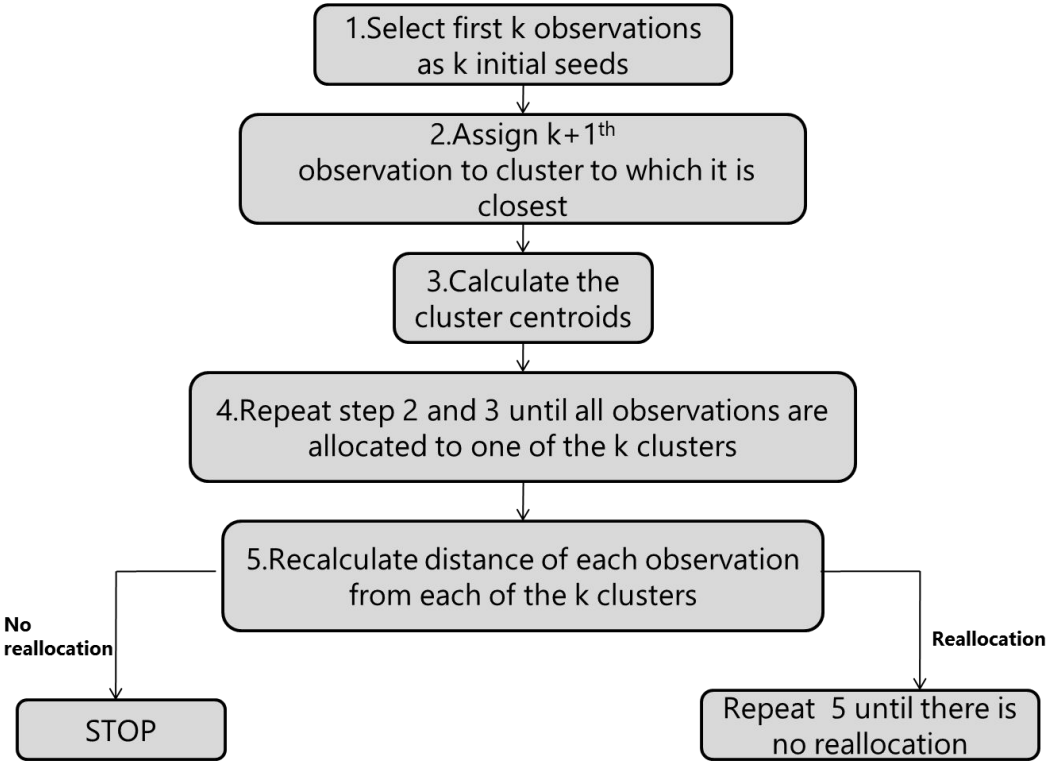
$$d(x , y) = (a1-b1)^2+(a2-b2)^2+....+(ap-bp)^2$$

- **The square root is defined as 'Euclidean Distance'**
- 'Euclidean Distance' is the most widely used distance measure in cluster analysis

Choice of Initial Seeds

- There are **different methods to decide initial seeds**, some of them are:
  - K-random observations
  - First K observations
  - Last K observations
  - Partition the data into k partitions randomly and then use the partition mean/ median as initial seeds

Algorithm


1.Select first k observations as k initial seeds
2.Assign k+1th observation to cluster to which it is closest
3.Calculate the cluster centroids
4.Repeat step 2 and 3 until all observations are allocated to one of the k clusters
5.Recalculate distance of each observation from each of the k clusters
No reallocation
Reallocation
STOP
Repeat 5 until there is no reallocation

Data Snapshot

Town Insurance

Variables

Observations

Town	x1	x2	x3
A	1.06	9.2	151
B	0.89	10.3	202
C	1.43	15.4	113
D	1.02	11.2	168
E	1.49	8.8	192
F	1.32	13.5	111
G	1.22	12.2	175
H	1.1	9.2	245

Columns	Description	Type	Measurement	Possible values
Town	Towns under study	character	-	-
x1	Loss Ratio	numeric	-	Positive values
x2	Premium Rates	numeric	-	Positive values
x3	Number of Policies	numeric	-	Positive values

Iteration 1

Data:

Town	x1	x2	x3
A	1.06	9.2	151
B	0.89	10.3	202
C	1.43	15.4	113
D	1.02	11.2	168
E	1.49	8.8	192
F	1.32	13.5	111
G	1.22	12.2	175
H	1.1	9.2	245

K=2

Step 1 : Initial Seeds

Initial Seeds

Town	x1	x2	x3
A	1.06	9.2	151
B	0.89	10.3	202

Iteration 1

Step 2: Find distance of Town C from A (Cluster1) and B(Cluster2)

Distance of C from A = $\sqrt{(1.43-1.06)^2+(15.4-9.2)^2+(113-151)^2}$ = **38.50**

Distance of C from B = √(1.43-0.89)2+(15.4-10.3)2+(113-202)2 = 89.15

Minimum Distance = **38.50**

Since distance between Town C and Town A is minimum, Town C will be combined with Town A

Updated cluster centroids (means) are:

Cluster	Members	X1	X2	X3
1	A,C	1.245	12.3	132
2	B	0.89	10.3	202

* Note : Values of x1,x2,x3 for 1st cluster are mean values of A & C

Iteration 1

Step 3: Find distance of Town D from Cluster1 and Cluster 2

Distance of D from Cluster 1	**36.018**
Distance of D from Cluster 2	**34.012**

Minimum Distance = 34.012

Here Town D will be combined with Town B (Cluster 2)

Updated cluster means are:

Cluster	Members	X1	X2	X3
1	A,C	1.245	12.3	132
2	B,D	0.955	10.75	185

Iteration 1

Step 4: Find distance of Town E from Cluster 1 and Cluster 2

Distance of E from Cluster 1	**60.102**
Distance of E from Cluster 2	**7.2862**

Minimum Distance = 7.2862

Here Town E will be combined with Cluster 2 (i.e. with Towns B & D)

Updated cluster means are:

Cluster	Members	X1	X2	X3
1	A,C	1.245	12.3	132
2	B,D,E	1.133333	10.1	187.3333

Iteration 1

Step 5: Find distance of Town F from Cluster 1 and Cluster 2

Distance of F from Cluster 1	**21.034**
Distance of F from Cluster 2	**76.409**

Minimum Distance = 21.034

Here Town F will be combined with Cluster 1 (i.e. with Towns A & C)

Updated cluster means are:

Cluster	Members	X1	X2	X3
1	A,C,F	1.27	12.7	125
2	B,D,E	1.133333	10.1	187.3333

Iteration 1

Step 6: Find distance of Town G from Cluster 1 and Cluster 2

Distance of G from Cluster 1	**50.003**
Distance of G from Cluster 2	**12.511**

Minimum Distance = 12.511

Here Town G will be combined with Cluster 2 (i.e. with Towns B,D & E)

Updated cluster means are:

Cluster	Members	X1	X2	X3
1	A,C,F	1.27	12.7	125
2	B,D,E,G	1.155	10.625	184.25

Iteration 1

Step 7: Find distance of Town H from Cluster 1 and Cluster 2

Distance of H from Cluster 1	**120.05**
Distance of H from Cluster 2	**60.767**

Minimum Distance = 60.767

Here Town H will be combined with Cluster 2 (i.e. with Towns B,D,E & G)

Updated cluster means are:

Cluster	Members	X1	X2	X3
1	A,C,F	1.27	12.7	125
2	B,D,E,G,H	1.144	10.34	196.4

Since all the towns are assigned to two clusters, to verify our clusters membership we go for the next iteration

Iteration 2

In iteration 2, initial seeds will be those two clusters which are obtained at the end of iteration 1

Step 1:

Initial seeds				
Cluster	**Members**	**X1**	**X2**	**X3**
1	A,C,F	1.27	12.7	125
2	B,D,E,G,H	1.144	10.34	196.4

Iteration 2

Step 2: Find the Distance of Town A from Cluster 1 (i.e from combined Towns A,C &F) and then Cluster 2(i.e from combined Towns B,D,E,G & H)

Distance of A from Cluster 1 = √(1.06-1.27)2+(9.2-12.7)2+(151-125)2 = 26.23536

Distance of A from Cluster 2 = √(1.06-1.44)2+(9.2-10.34)2+(151-196.4) = 45.41439

Minimum Distance = 26.23536

Since distance between Town A and Cluster 1 is minimum, Town A will be retained in Cluster 1

Iteration 2 – Summary

Initial seeds				
Cluster	**Members**	**X1**	**X2**	**X3**
1	A,C,F	1.27	12.7	125
2	B,D,E,G,H	1.144	10.34	196.4

No town is reassigned to different cluster

This is final cluster solution

Town	Distance from Cluster1	Distance from Cluster2	Cluster
A	26.24	45.41	1
B	77.04	5.61	2
C	12.30	83.55	1
D	43.03	28.41	2
E	67.11	4.67	2
F	14.02	85.46	1
G	50.00	21.48	2
H	120.05	48.61	2

Points For Good Cluster Solution

- **Standardize variables if scale differs widely**. Variables with high variance tend to influence cluster solution.
- Use **data reduction technique like factor analysis before cluster analysis if number of variables is high**
- **Run the algorithm for different choices of K and initial seeds**
- **Use dummy variables for nominal scaled variables**. K means algorithm is not suited for nominal scaled variables.
- **You may use hierarchical clustering for sample of the data to get preliminary information on K and distance values**

Statistics Associated with Cluster Solution

Cluster solution can be assessed using 'between clusters' variability and 'within clusters' variability.

Within Sum of Squares (WSS) is a measure to explain homogeneity within a cluster. WSS can be calculated for each cluster and then added to get Total WSS

Total WSS should be small

R-squared is computed as ratio of Between Clusters Variability to Total Variability.

R-squared should be large

Quick Recap

Cluster Analysis	• Statistical techniques that can be used to classify objects or cases into groups called Clusters • Cluster is a group of relatively homogeneous cases or observations
K-Means Clustering – Steps Involved	• Define k • Define Distance Measure • Define Initial Seeds • Assign cases to clusters based on distance measure • Repeat the Procedure Until no reassignment is required

Non-Hierarchical Clustering – II

K Means Method

Contents

Case Study

Background

- A FMCG company has recorded information of customers based on their buying behaviour for a period of 1 year and would like to implement strategies by segmenting these customers into tiers.

Objective

- To create segment of customers.

Available Information

- Sample size is 1158.
- Variables : Custid, nsv, n_brands , n_bills, growth, region

Data Snapshot

RETAILERS DATA

Variables

Custid	nsv	n_brands	n_bills	growth	region
1001	2119456	7	14	-1.79	Mumbai
1002	1460163	12	42	-1.73	Mumbai
1003	147976	4	6	2.81	Mumbai
1018	2213576	14	14	5.69	Delhi
1019	2433971	11	25	3.71	Delhi

Columns	Description	Type	Measurement	Possible values
Custid	Unique customer ID	numeric	-	-
nsv	Net Sales Value	numeric	Rs.	positive values
n_brands	Number of unique brands purchased	numeric	-	positive values
n_bills	Number of bills generated	numeric	-	positive values
growth	Growth in net sales value	numeric	-	Positive & negative values
region	City of Customer	character	Delhi, Kolkata, Mumbai, Nagpur	4

K-Means Method in R

```
# Importing Data
custsales<-read.csv("RETAILERS DATA.csv",header=T)
custsales_cl<-subset(custsales,select=c(-Custid,-region))

# Scale (standardize) all variables.(subtract mean and divide by
standard deviation)

custsales_cl<-scale(custsales_cl)
CL<-kmeans(custsales_cl,4)
CL
```

kmeans() perform k-means clustering on a data matrix cutsales_cl for 4 clusters.

```
# Output
K-means clustering with 4 clusters of sizes 229, 314, 210, 405

Cluster means:
         nsv    n_brands    n_bills      growth
1  1.1863778 -0.02444231  0.3044816 -0.62581250
2 -0.5014544  0.09508729 -0.3772226  0.05665368
3  1.0589762  1.50534917  1.6219927  1.62282815
4 -0.8311329 -0.84045295 -0.7207329 -0.53153606

Within cluster sum of squares by cluster:
[1] 314.9123 145.5306 732.8205 166.3279
 (between_SS / total_SS =  70.6 %)
```

Interpretation :

- Cluster 3 looks platinum customers group.

K-Means Method in R
Append Segment Variable

```
# Adding New column “segment” :

custsales$segment <- CL$cluster
head(custsales)
```

```
# Output
```

	Custid	nsv	n_brands	n_bills	growth	region	segment
1	1001	2119456	7	14	-1.79	Mumbai	1
2	1002	1460163	12	42	-1.73	Mumbai	1
3	1003	147976	4	6	2.81	Mumbai	4
4	1004	1350474	13	30	-0.99	Delhi	1
5	1005	1414461	15	29	13.56	Delhi	3
6	1006	2299185	21	49	11.07	Delhi	3

K-Means Method in R : Summarize Clusters Using Original Variables

```
# Aggregating data based on segments
aggregate(cbind(nsv,n_brands,n_bills,growth)~segment,data=custsales,
FUN=mean)
```

```
# Output
```

	segment	nsv	n_brands	n_bills	growth
1	1	1985624.2	11.532751	24.532751	1.836419
2	2	524186.9	12.525478	12.070064	5.004777
3	3	1875311.4	24.238095	48.619048	12.275762
4	4	238729.4	4.755556	5.790123	2.274099

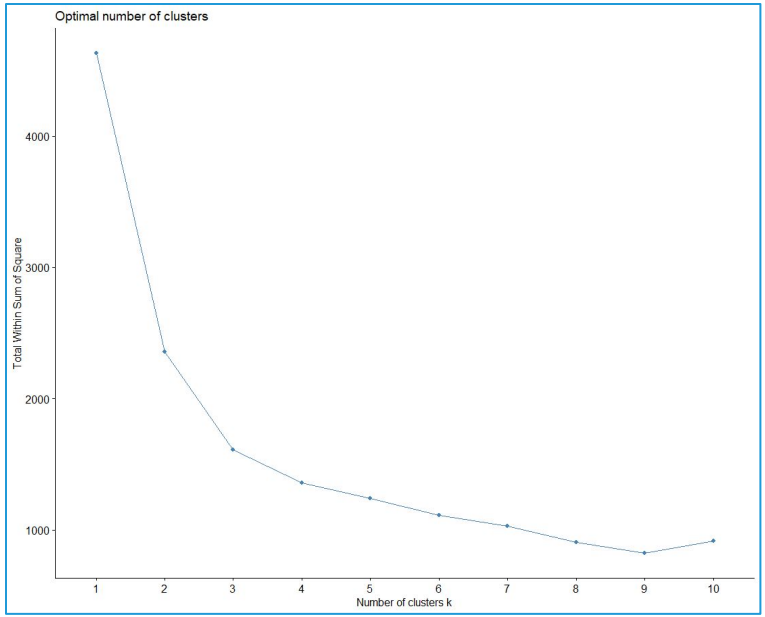
Interpretation :

- Cluster 3 is group of ‘Platinum’ clusters.
- Cluster 4 is a group of ‘non-performers’

K-Means Method in R
Elbow Method

- **fviz_nbclust()** determines & visualize the optimal number of clusters using different methods, here we use within cluster sums of squares(wss)

```
# Install & load package “factoextra”
install.packages("factoextra")
library(factoextra)
fviz_nbclust(custsales_cl, kmeans,method = "wss")
# Output
```



Interpretation :

- The location of a bend in the plot is generally considered as an indicator of the appropriate number of clusters.
- Here K= 3 or 4 is a good solution.
- The method is termed as Elbow Method.

kmeansruns() in "fpc" Package
Finding Best K

- Package: fpc: Flexible Procedures for Clustering

- Performs K-means method for different values of 'K' and provides best value of K.

```
library(fpc)
CL1<-kmeansruns(custsales_cl,krange=2:10)
CL1$bestk
```

- **kmeansruns(data,krange)**
- **data** A numeric matrix of data, or an object that can be coerced to such a matrix
- **krange=** integer vector. Numbers of clusters which are to be compared

* Use this as only indicative

K-Means Clustering in R

kmeans() output includes :

cluster	A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
centers	A matrix of cluster centres.
totss	The total sum of squares.
withinss	Vector of within-cluster sum of squares, one component per cluster.
tot.withinss	Total within-cluster sum of squares, i.e. sum (withinss).
betweenss	The between-cluster sum of squares, i.e. totss-tot.withinss.
size	The number of points in each cluster.
iter	The number of (outer) iterations.
ifault	integer: indicator of a possible algorithm problem – for experts.

K Means Algorithms

- The algorithm of Hartigan and Wong (1979) is used by default.

- Note that some authors use k-means to refer to a specific algorithm rather than the general method: most commonly the algorithm given by

MacQueen (1967) or sometimes that given by Lloyd (1957) and Forgy (1965).

K-Median Clustering in R

```
# Install and load package “flexclust”
# K-Median Clustering
install.packages("flexclust")
library(flexclust)

kmedian<-kcca(custsales_cl,3,family=kccaFamily("kmedian"))
kmedian
```

- **kcca()** performs k-centroid clustering on data matrix. The first two arguments are **data object** and **number of clusters** to be formed.
- **family=kccaFamily()** specifies object of class **kccaFamily**. Other options are **"kmeans", "angle", "jaccard",** or **"ejaccard"**.

```
# Output
kcca object of family ‘kmedians’

call:
kcca(x = custsales_cl, k = 3, family = kccaFamily("kmedian"))

cluster sizes:

  1   2   3
243 217 698
```

K-Median Clustering in R

```
# Adding New column “segment” :
custsales$seg_median <- kmedian@cluster
head(custsales)
```

```
# Output
```

	Custid	nsv	n_brands	n_bills	growth	region	seg_median
1	1001	2119456	7	14	-1.79	Mumbai	1
2	1002	1460163	12	42	-1.73	Mumbai	1
3	1003	147976	4	6	2.81	Mumbai	3
4	1004	1350474	13	30	-0.99	Delhi	1
5	1005	1414461	15	29	13.56	Delhi	2
6	1006	2299185	21	49	11.07	Delhi	2

Quick Recap

K-Means Clustering in R	• **`kmeans()`** function in base R performs K-Means Clustering • **`kmeansruns()`** from package **`fpc`** can also be used for finding number of clusters.
K-Median Clustering in R	• **`kcca()`** function from package **`flexclust`** performs k-centroid clustering on data matrix. • **`kccaFamily()`** specifies object of class **`kccaFamily`**.

Hierarchical Clustering

Contents

Hierarchical Methods

- Hierarchical clustering creates clusters which have a pre-determined, top to bottom ordering
- Hierarchical clustering method is further divided into two approaches
  - Agglomerative
  - Divisive

Agglomerative clustering starts with each object in a separate cluster, clusters are formed by grouping objects into bigger and bigger clusters, the process is continued until all objects are members of single cluster

Divisive clustering starts with all objects grouped in a single cluster , clusters are divided until each observation is in a separate cluster

Distance Measures – Euclidean Distance

- Squared Euclidean Distance

Object	X1	X2			XP
1	a1	a2			ap
2	b1	b2			bp

The sum of squared differences between values of each variable

$$d(x , y) = (a1-b1)^2+(a2-b2)^2+....+(ap-bp)^2$$

- The square root is defined as 'Euclidean Distance'
- 'Euclidean Distance' is the most widely used distance measure in cluster analysis

Case Study

Background

- An insurance company would like to explore its small town business and create strategies for different groups of small towns

Objective

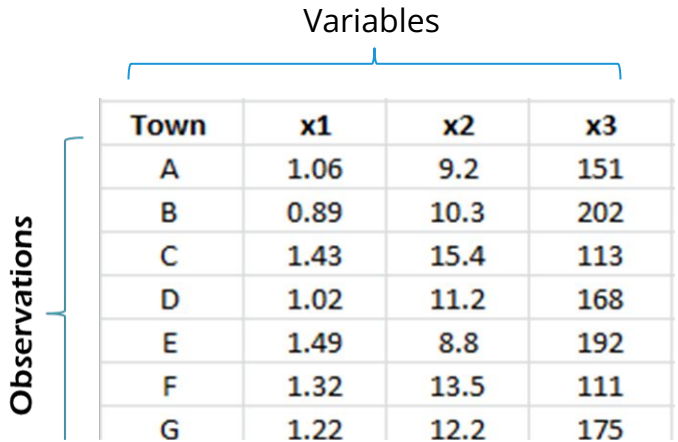
- To form clusters of towns

Available Information

- Data for 8 towns is recorded
- Information fields for the towns are
  - Loss Ratio
  - Premium Rates
  - No. of Policies from that town

Data Snapshot

Town Insurance



Town	x1	x2	x3
A	1.06	9.2	151
B	0.89	10.3	202
C	1.43	15.4	113
D	1.02	11.2	168
E	1.49	8.8	192
F	1.32	13.5	111
G	1.22	12.2	175

Columns	Description	Type	Measurement	Possible values
Town	Towns under study	character	-	-
x1	Loss Ratio	numeric	-	Positive values
x2	Premium Rates	numeric	-	Positive values
x3	Number of Policies	numeric	-	Positive values

Euclidean Distance Matrix – Iteration 1

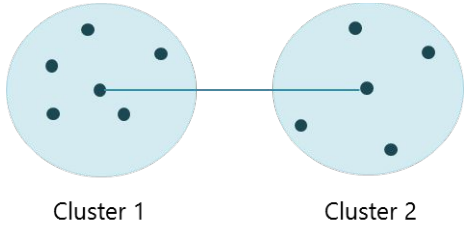
Distance between C and A = $\sqrt{(1.43-1.06)^2+(15.4-9.2)^2+(113-151)^2}$ = **38.50**

	A	B	C	D	E	F	G	H
A		51.01	38.50	17.12	41.00	40.23	24.19	94.00
B	51.01		89.15	34.01	10.13	91.06	27.07	43.01
C	38.50	89.15		55.16	79.28	2.76	62.08	132.15
D	17.12	34.01	55.16		24.12	57.05	7.07	77.03
E	41.00	10.13	79.28	24.12		81.14	17.34	53.00
F	40.23	91.06	2.76	57.05	81.14		64.01	134.07
G	24.19	27.07	62.08	7.07	17.34	64.01		70.06
H	94.00	43.01	132.15	77.03	53.00	134.07	70.06	

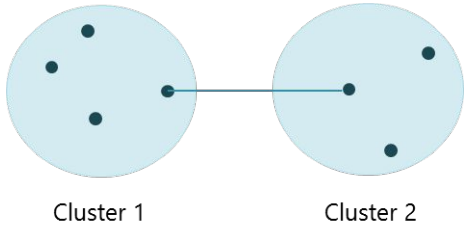
- **Distance between Firm 'C' and 'F' is minimum** hence ,Firms 'C' and 'F' are combined at the first iteration.
- The distance matrix is revised as per methods described in next 2 slides and subsequent merging is performed.

Agglomerative Clustering

- **Centroid method**: The distance between 2 clusters is measured as a Euclidean distance between centroids (average value) of each of the clusters

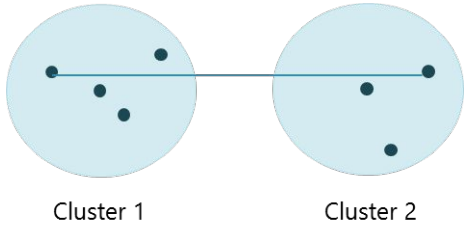


- **Single linkage**: The distance between 2 clusters is the distance between their two closest points. At any stage the clusters are merged by the single shortest distance

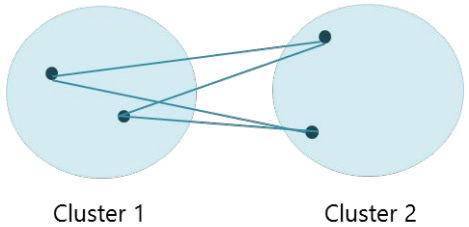


Agglomerative Clustering

- **Complete linkage**: The distance between 2 clusters is the distance between their two farthest points. At any stage the clusters are merged by the single farthest distance



- **Average linkage**: The distance between 2 clusters is defined as the average of the distances between all pairs of objects, where one member of the pair is from each of the cluster



Get an Edge!

More Linkage Methods

There are three more ways in which cluster linkages can be established

1. Median Linkage	Median distance between the observations in two clusters
2. McQuitty Linkage	Two clusters are to be joined, then the distance of the new cluster to any other cluster is the average of distance of the clusters to be joined to the other cluster
3. Ward Linkage	Sum of squared deviation from point to centroid. Objective is to minimize the within cluster sum of squares

Single Linkage Method

		Cluster Membership							
Iteration No	No of Cluster Formed	1	2	3	4	5	6	7	8
step1	8	A	B	C	D	E	F	G	H
1	7	A	B	C,F	D	E	G	H	
2	6	A	B	C,F	D,G	E	H		
3	5	A	B,E	C,F	D,G	H			
4	4	A	B,E,D,G	C,F	H				
5	3	A,B,E,D,G	C,F	H					
6	2	A,B,E,D,G, C,F	H						
7	1	A,B,E,D,G, C,F,H							

Hierarchical Clustering in R

```
#Importing and Readying the Data
townins<-read.csv("Town Insurance.csv",header=T)

townins2<-subset(townins,select=c(-Town))

#Calculating Distance Matrix
d<-dist(townins2, method="euclidean")

#Hierarchical Clustering

fit <- hclust(d, method="single")
```

dist() computes and returns distance matrix.

- **hclust()** runs hierarchical clustering.
- **d=** Distance Matrix
- **method=** specifies linkage method.

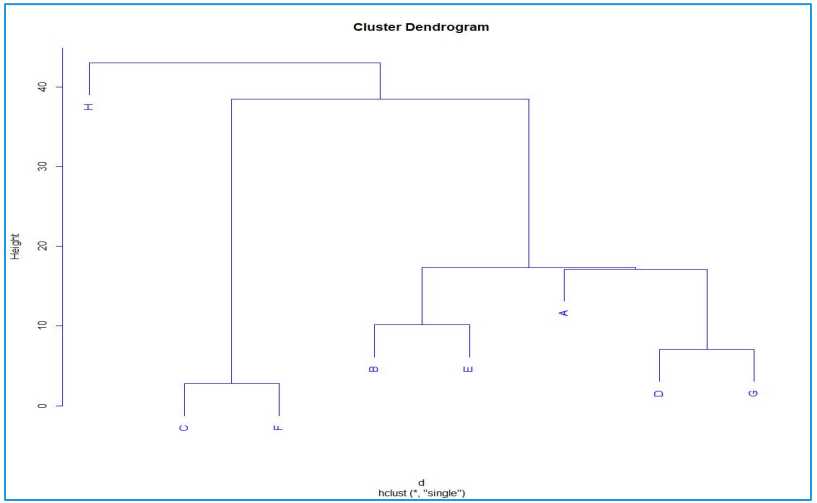
* method= can take the following arguments depending on which method is used for linkage. Available options are - “complete”, “average”, “centroid”, “median”, “mcquitty” and “ward.D2”

Hierarchical Clustering in R

```
#Display Dendrogram
plot(fit,label=townins$Town,col="blue")
```

Plotting **hclust()** output returns a dendrogram.

```
# Output
```



- Towns C and F are close and form one cluster
- Towns A,B,D,E and G form second cluster
- Town H looks separate entity

Hierarchical Clustering in R

```
# Append Segment Variables
townins$segment<-cutree(fit,k=3)

townins
```

```
# Output
```

	Town	x1	x2	x3	segment
1	A	1.06	9.2	151	1
2	B	0.89	10.3	202	1
3	C	1.43	15.4	113	2
4	D	1.02	11.2	168	1
5	E	1.49	8.8	192	1
6	F	1.32	13.5	111	2
7	G	1.22	12.2	175	1
8	H	1.10	9.2	245	3

- **cutree()** cuts a dendrogram tree into several groups by specifying the desired number of clusters k(s), or cut height(s).
- **fit** is the dendrogram tree
- **k=** number of clusters the tree should be cut into.

Quick Recap

Hierarchical Clustering	• Hierarchical Clustering can be of two types –
	  • **Agglomerative** – starts with each object in separate clusters and stepwise merging is carried out until a single cluster with all objects is formed
	  • **Divisive** – All objects grouped in a single cluster, clusters are divided until each observation is in a separate cluster

Hierarchical Clustering in R	• **`hclust()`** function in package **`stats`** performs Hierarchical (Agglomerative) clustering. • **`cutree()`** cuts a dendrogram tree into several groups by specifying the desired number of clusters k(s), or cut height(s).