DEFENCE INSTITUTE OF ADVANCED TECHNOLOGY

DEEMED UNIVERSITY

शास्त्रेण शस्त्रं प्रकरोति रक्षाम्

Machine Learning for Cyber Security (CS-602)

L#03

Introduction – Distances

By

Dr Sunita Dhavale

Syllabus

- Data Analytics Foundations: R programming, Python Basics -Expressions and Variables, String Operations, Lists and Tuples, Sets, Dictionaries Conditions and Branching, Loops, Functions, Objects and Classes, Reading/Writing files, Hand ling data with Pandas, Scikit Library, Numpy Library, Matplotlib, scikit programming for data analysis, setting up lab environment, study of standard datasets. Introduction to Machine Learning- Applications of Machine Learning, Supervised, unsupervised classification and regression analysis
- Python libraries suitable for Machine Learning Feature Extraction. Data pre-processing, feature analysis etc., Dimensionality Reduction & Feature Selection Methods, Linear Discriminant Analysis and Principal Component Analysis, tackle data class imbalance problem

Syllabus

- Supervised and regression analysis, Regression, Linear Regression, Non-linear Regression, Model evaluation methods, Classification, K-Nearest Neighbor, Naïve Bayes, Decision Trees, Logistic Regression, Support Vector Machines, Artificial Neural Networks, Model Evaluation. Ensemble Learning, Convolutional Neural Networks, Spectral Embedding, Manifold detection and Anomaly Detection
- Unsupervised classification K-Means Clustering, Hierarchical Clustering, Density-Based Clustering, Recommender Systems-Content-based recommender systems, Collaborative Filtering, machine learning techniques for standard dataset, ML applications, Case studies on Cyber Security problems that can be solved using Machine learning like Malware Analysis, Intrusion Detection, Spam detection, Phishing detection, Financial Fraud detection, Denial of Service Detection.

Text/Reference Books

1. Building Machine Learning Systems with Python – Willi Richert, Luis Pedro Coelho
2. Alessandro Parisi, Hands-On Artificial Intelligence for Cybersecurity: Implement smart AI systems for preventing cyber attacks and detecting threats and network anomalies Publication date :Aug 2, 2019, Packt, ISBN-13, 9781789804027
3. Machine Learning: An Algorithmic Perspective – Stephen Marsland
4. Sunita Vikrant Dhavale, "Advanced Image-based Spam Detection and Filtering Techniques", IGI Global, 2017
5. Soma Halder , Sinan Ozdemir, Hands-On Machine Learning for Cybersecurity: Safeguard your system by making your machines intelligent using the Python ecosystem, By Publication date : Dec 31, 2018, Packt, ISBN-13 :9781788992282

1. Stuart Russell, Peter Norvig (2009), "Artificial Intelligence – A Modern Approach", Pearson Elaine Rich & Kevin Knight (1999), "Artificial Intelligence", TMH, 2nd Edition

2. NP Padhy (2010), "Artificial Intelligence & Intelligent System", Oxford

3. ZM Zurada (1992), "Introduction to Artificial Neural Systems", West Publishing Company

4. Research paper for study (if any) – White papers on multimedia from IEEE/ACM/Elsevier/Spinger/ Nvidia sources.

Lab assignments

1	**Python Programming part-1**
2	Python Programming part-2
3	Study and Implement Linear Regression Algorithm for any standard dataset like in cyber security domain
4	Study and Implement KMeans Algorithm for any standard dataset in cyber security domain
5	Study and Implement KNN for any standard dataset in cyber security domain
6	Study and Implement ANN for any standard dataset in cyber security domain
7	Study and Implement PCA for any standard dataset in cyber security domain
8	Case Study: Use of ML along with Fuzzy Logic/GA to solve real world Problem in cyber security domain
9	Mini assignment: Apply ML along with PSO/ACO to solve any real world problem in cyber security domain
10	ML Practice Test – 1 Quiz

Defence Institute of Advanced Technology

School of Computer Engineering & Mathematical Sciences

SEMESTER-I TIME TABLE (AUTUMN 2024)$

PROGRAMMES: (I) CS [M.TECH IN CYBER SECURITY] (II) AI [M.TECH CSE (ARTIFICIAL INTELLIGENCE)] BATCH: 2024-2026

Lecture / Day	L1 0900-1000	L2 1000-1100	L3 1100-1200	L4 1200-1300	Lunch Break 1300-1400	L4 1400-1500	L4 1500-1600	L4 1600-1700	L4 1700-1800
Monday	CE-602 (AI) CS-602 (CS)	CE-604 (AI) CS-603 (CS)	CE-601 (AI) CS-604 (CS)	CE-601 (AI)		LAB CE-601 (AI) LAB CS-602 (CS)		AM607	
Tuesday	CE-603 (AI) LAB CS-603 (CS)	CE-602 (AI) CS-602 (CS)	CE-601 (AI) CS-605 (CS)	CE-604 (AI) CS-604 (CS)		PGC 601		AM607	LAB CS-603 (CS)
Wednesday	CE-604 (AI) CS-605 (CS)	CE-603 (AI) CS-602 (CS)	CE-602 (AI) CS-603 (CS)	LAB CE-604 (AI) CS-604 (CS)		CE-605(AI) LAB CS-605 (CS)	LAB CS-605 (CS)	AM607	LAB CE-604 (AI)
Thursday	CE-604 (AI) CS-603 (CS)	CS-605 (CS)	LAB CE-602 (AI) CS-601 (CS)	CE-603 (AI) CS-601 (CS)		PGC 601		AM607	
Friday	LAB CE-603 (AI) LAB CS-601 (CS)		LAB CE-602 (AI) CS-601 (CS)	LAB CS-604 (CS)		CE-605(AI) LAB CS-604 (CS)	CE-605(AI)	LAB CE-605(AI)	

COURSE CODE & COURSE NAME		FACULTY
Programme: CS [M.Tech in Cyber Security] Classroom: Arjun	Programme: AI [M.Tech CSE (Artificial Intelligence)] Classroom: Kaveri	
CS-601 Data Security & Privacy	CE-601 Responsible Artificial Intelligence;	MJN: Dr. Manisha J. Nene
CS-602 ML for Cyber Security	CE-604 Practical Machine Learning;	SVD: Dr. Sunita V. Dhavale
CS-605 Network and Cloud Security	CE-602 Intelligent Algorithms	CRS: Prof. CRS Kumar
CS-604 Advanced System Security	---------	DVV: Dr. Deepti V. Vidyarthi
CS-603 Applied Cryptography	---------	AM: Dr. Arun Mishra
---------	CE-603 Deep Neural Network;	US: Dr. Upasna Singh
---------	CE-605 Mathematics for ML;	Unit-2: Dr Upasna, Unit 4: Dr Sunita, Unit3:MJN, Unit 1: Faculty To be Nominated
AM-607 Mathematics for Engineers	AM-607 Mathematics for Engineers	OO/DS/DP: Dr Odellu O., Dr Dasari S., Dr. Debasis P.
PGC-601 Research Methodology	PGC-601 Research Methodology	Common Subject for All

$ TENTATIVE T.T. SUBJECT TO CHANGE

Program Coordinator,
M.Tech (CS & AI), Batch 2024-26

Director, SoCE&MS

Distance Metrics in Machine Learning

- Types of Distance Metrics in Machine Learning
  - Euclidean Distance
  - Manhattan Distance
  - Minkowski Distance
  - Hamming Distance
  - Cosine Distance
  - Mahalanobis Distance

Euclidean Distance

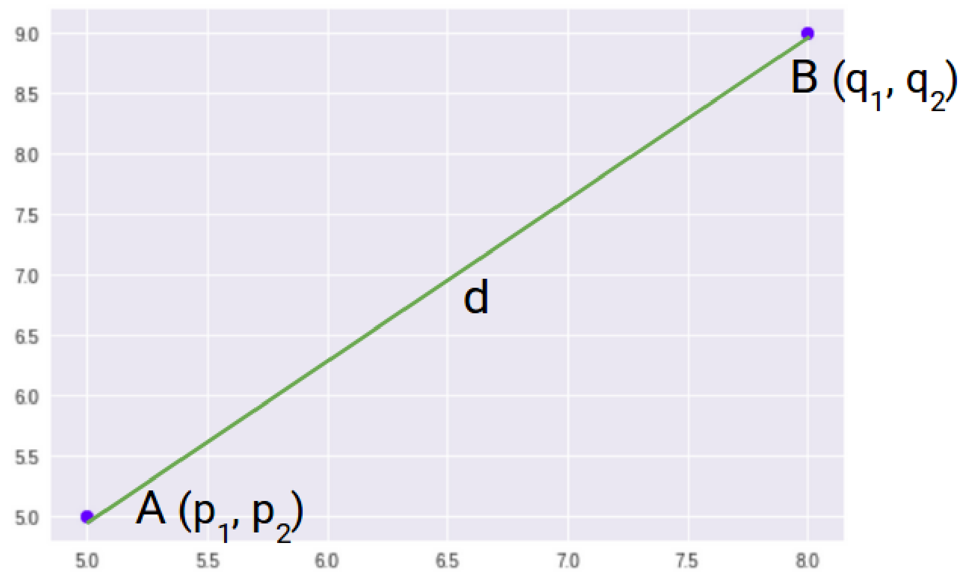
- Euclidean Distance represents the shortest distance between two points.
- Most machine learning algorithms including K-Means use this distance metric to measure the similarity between observations
- Used when dealing with continuous or numerical variables

Euclidean Distance

generalize this for an n-dimensional space- if n = number of dimensions, pi, qi = data points

$$d = ((p_1 - q_1)^2 + (p_2 - q_2)^2)^{1/2}$$

$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$



```
Euclidean Distance b/w (1, 2, 3) and (4, 5, 6) is:
```

R Tool

- # Function to calculate Euclidean distance
- # Sum function calculates the sum of the
- # squares of absolute difference between
- # corresponding elements of vect1 and vect2
- CalculateEuclideanDistance <- function(vect1, vect2) sqrt(sum((vect1 - vect2)^2))
- # Initializing two vectors having equal length
- vect1 <- c(2, 4, 4, 7)
- vect2 <- c(1, 2, 2, 10)
- print("Euclidean distance between vect1 and vect2 is: ")
- # Calling CalculateEuclideanDistance function
- CalculateEuclideanDistance(vect1, vect2)

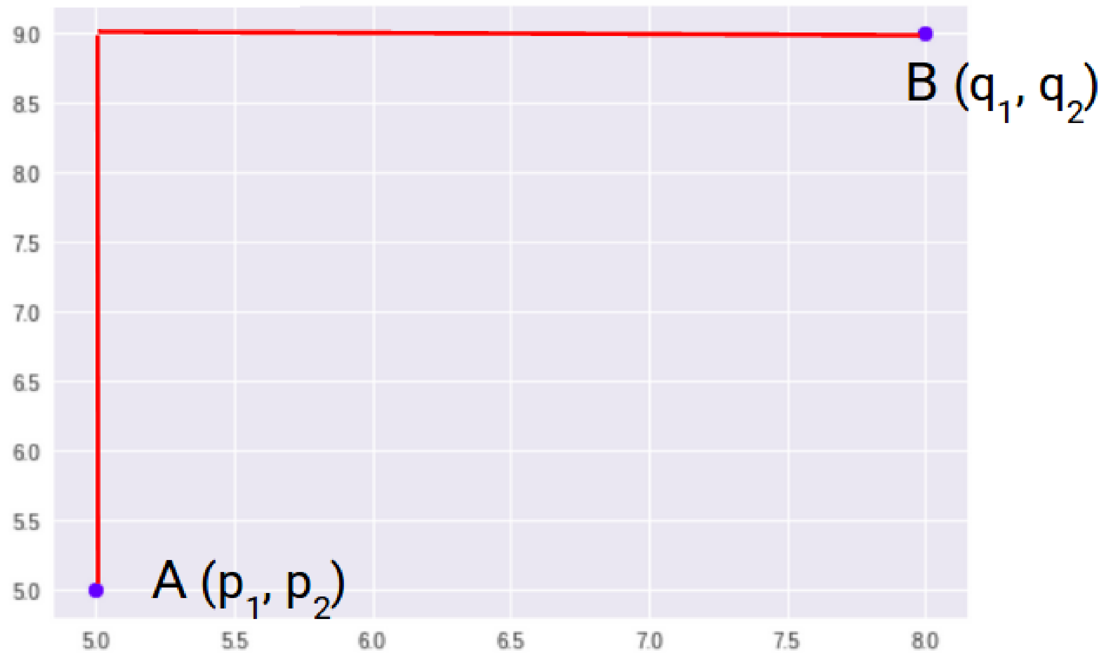
Manhattan Distance

- Manhattan/ *CityBlock* Distance is the sum of absolute differences between points across all the dimensions
- Used when dealing with continuous or numerical variables

Manhattan Distance

$$d = |p_1 - q_1| + |p_2 - q_2|$$

$$D_m = \sum_{i=1}^{n} |p_i - q_i|$$



```
Manhattan Distance b/w (1, 2, 3) and (4, 5, 6) is:
```

R Tool

- manhattanDistance <- function(vect1, vect2){
-     dist <- abs(vect1 - vect2)
-     dist <- sum(dist)
-     return(dist)
- }
- # Initializing a vector
- vect1 <- c(3, 6, 8, 9)
- # Initializing another vector
- vect2 <- c(1, 7, 8, 10)
- print("Manhattan distance between vect1 and vect2 is: ")
- # Call the function to calculate Manhattan
- # distance between vectors
- manhattanDistance(vect1, vect2)

Minkowski Distance

- Minkowski Distance is the generalized form of Euclidean and Manhattan Distance.
- p represents the order of the norm.
- When the order(p) is 1, it will represent Manhattan Distance
- when the order is 2, it will represent Euclidean Distance.
- Used when dealing with continuous or numerical variables

$$D = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

Hamming Distance

- Hamming Distance measures the similarity between two strings of the same length.
- The Hamming Distance between two strings of the same length is the number of positions at which the corresponding characters are different.
- larger the Hamming Distance between two strings, more dissimilar will be those strings
- Used when categorical variables
- only works when we have strings of the same length.
- Calculate Hamming Distance between two strings: “**euclidean**” and “**manhattan**”

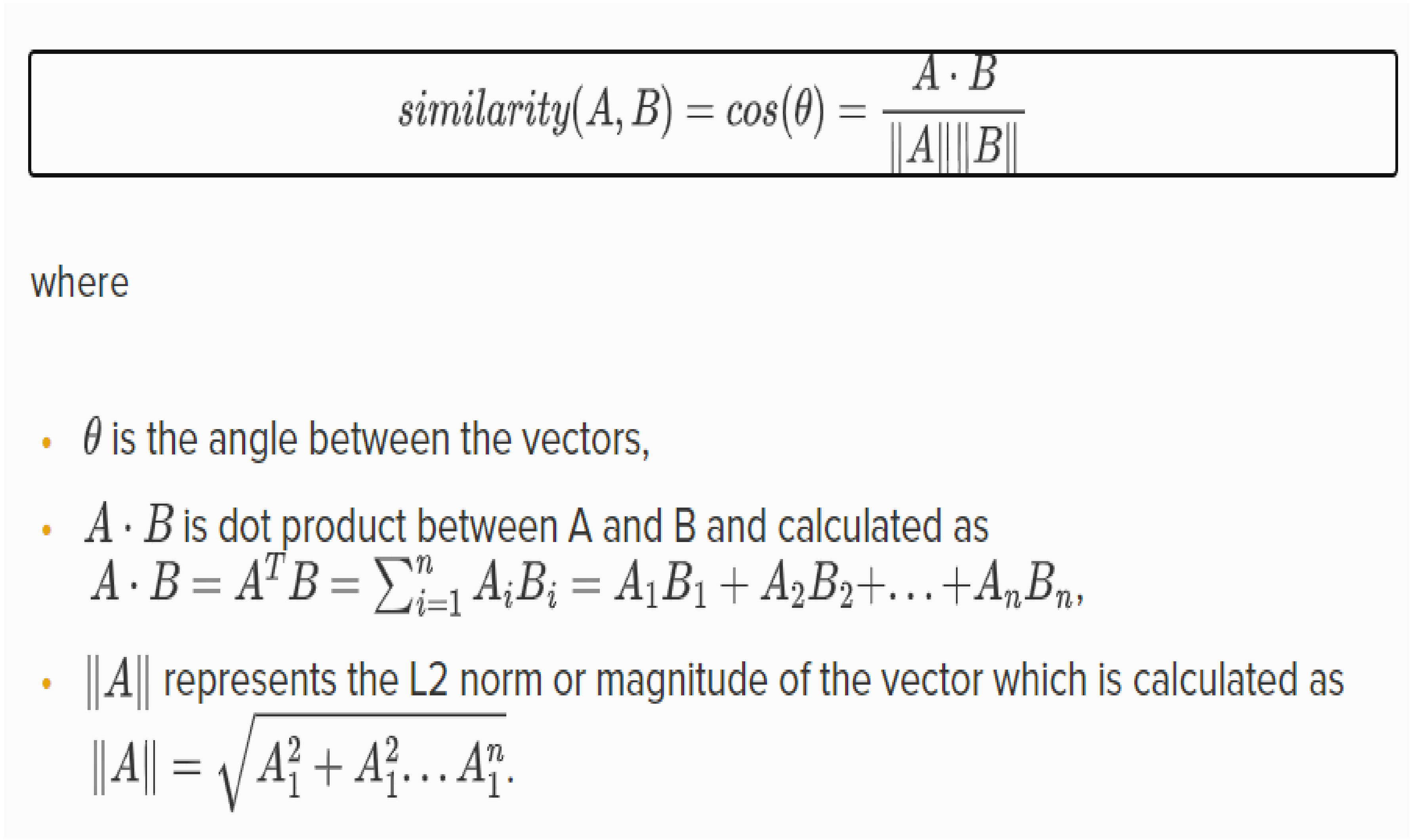
R Tool

- #function to compute Hamming distance
- # between numeric vectors
- # Initializing a vector of integers
- vect1 <- c(10, 2, 3, 7, 8, 12)
- # Initializing another vector of Qintegers
- vect2 <- c(10, 2, 1, 2, 0, 24)
- # Hamming distance
- HammingDistance = sum(vect1 != vect2)
- # Print Hamming distance
- print(paste("Hamming distance between vect1 and vect2 \
- is equal to", HammingDistance))

Cosine Distance

- to determine how similar the documents are irrespective of their size,
- smaller the angle, higher the similarity
- used as a metric in different machine learning algorithms like the KNN for determining the distance between the neighbors
- In recommendation systems, it is used to recommend movies with the same similarities and for textual data, it is used to find the similarity of texts in the document.
- The similarity can take values between -1 and +1. Smaller angles between vectors produce larger cosine values, indicating greater cosine similarity.

Cosine Distance

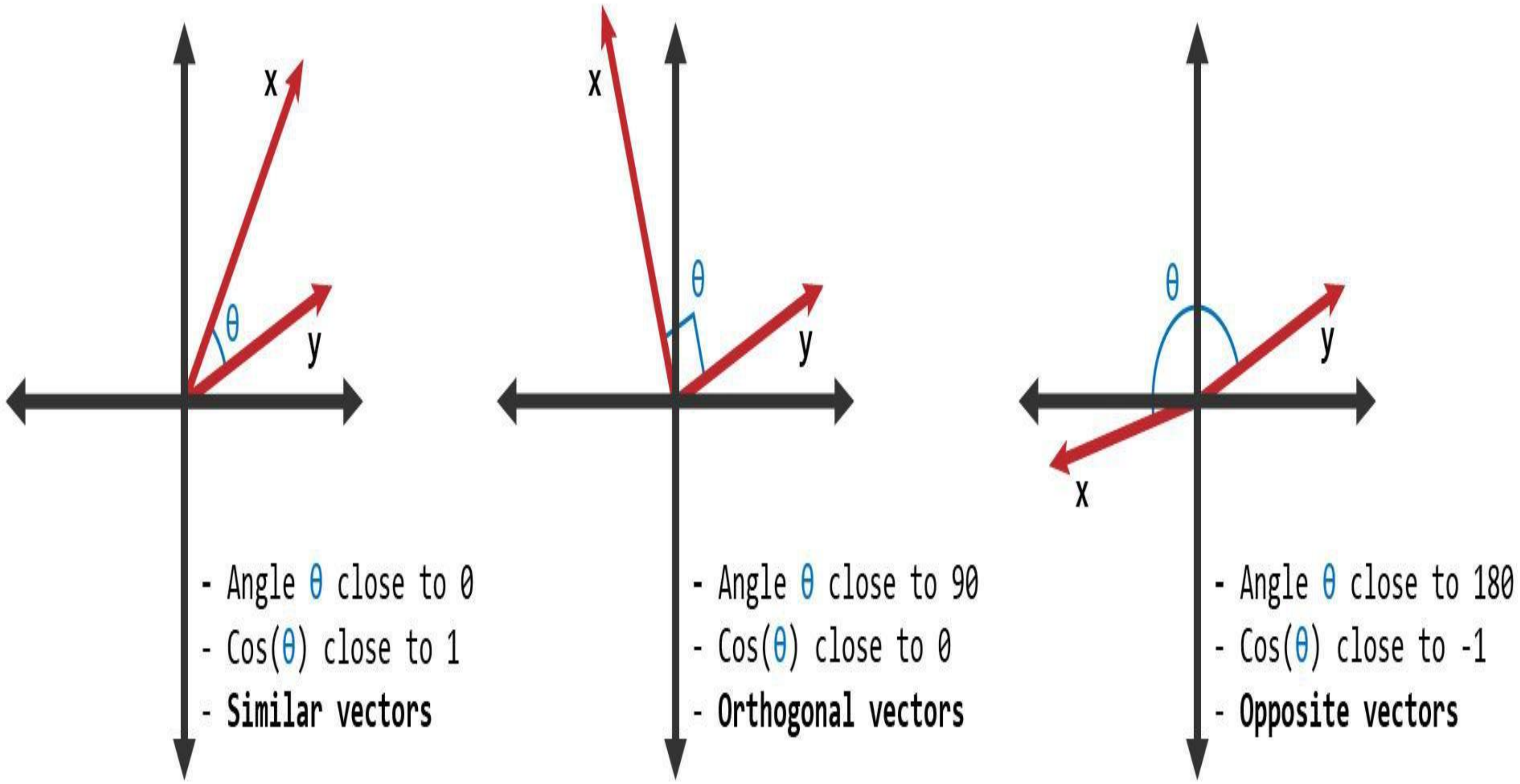


$$\boxed{similarity(A,B) = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}}$$

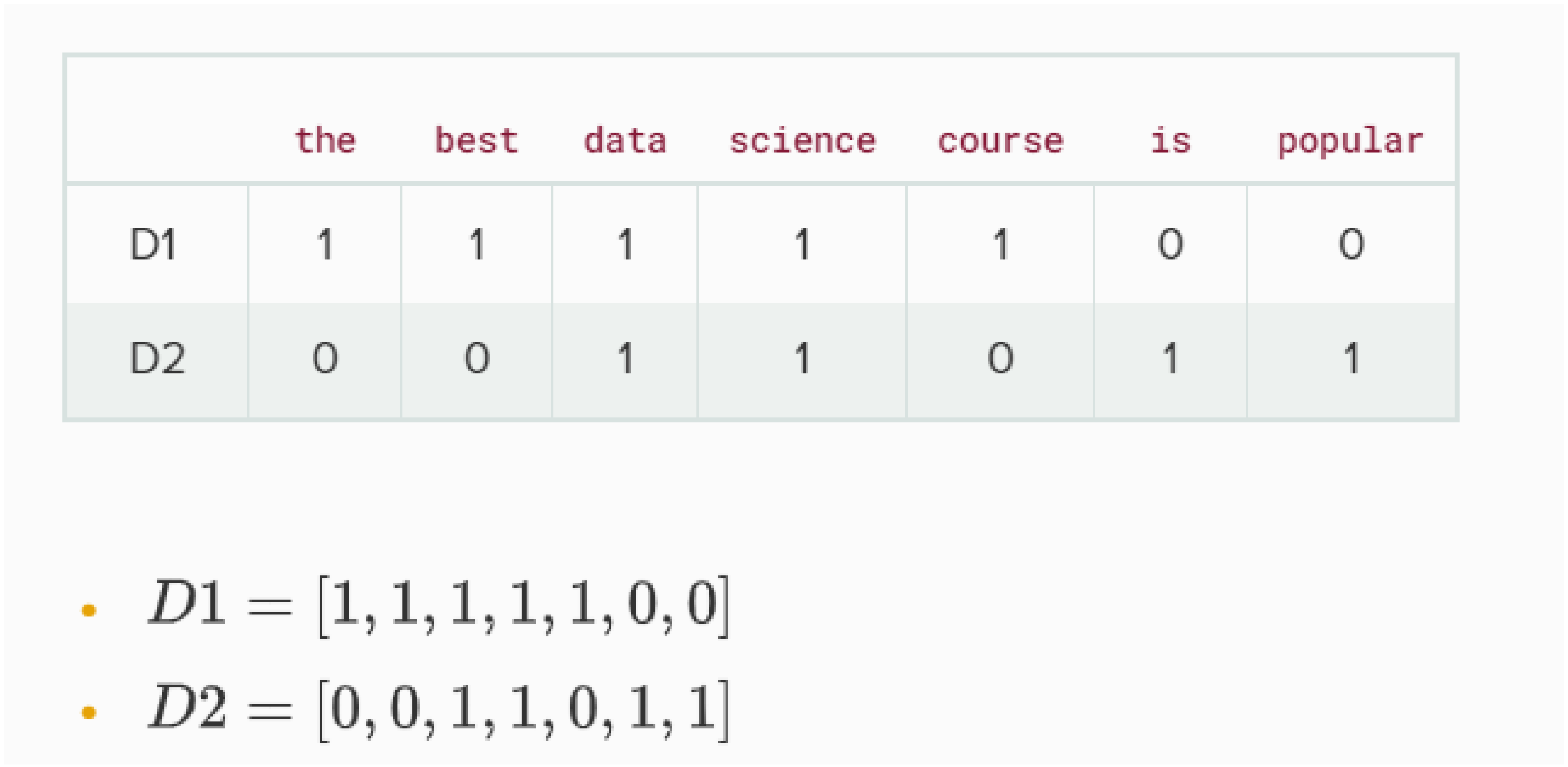
where

- θ is the angle between the vectors,
- $A \cdot B$ is dot product between A and B and calculated as $A \cdot B = A^T B = \sum_{i=1}^{n} A_i B_i = A_1 B_1 + A_2 B_2 + \ldots + A_n B_n$,
- $\|A\|$ represents the L2 norm or magnitude of the vector which is calculated as $\|A\| = \sqrt{A_1^2 + A_1^2 \ldots A_1^n}$.

Cosine Distance



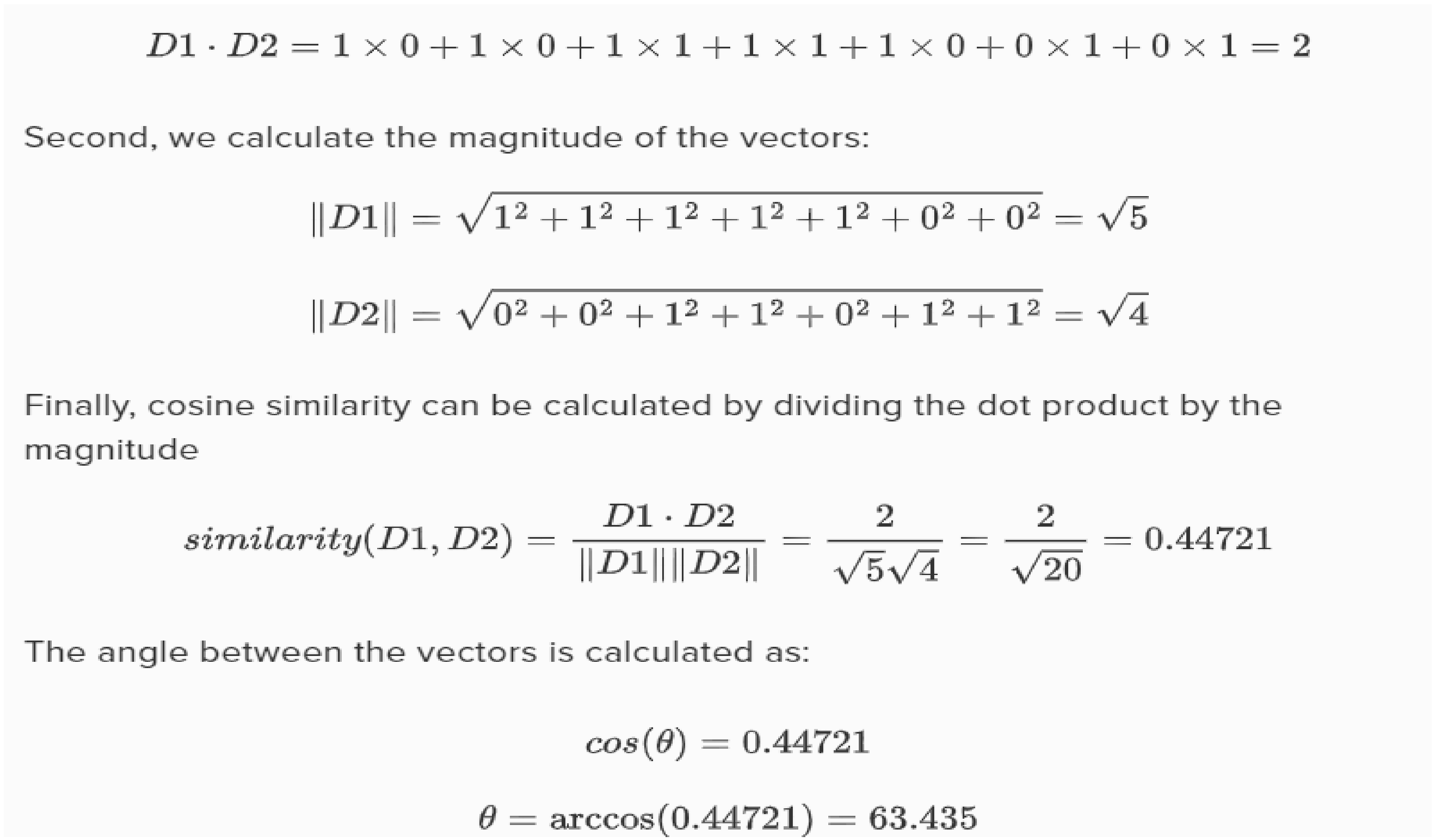
Cosine Distance



	the	best	data	science	course	is	popular
D1	1	1	1	1	1	0	0
D2	0	0	1	1	0	1	1

- $D1 = [1, 1, 1, 1, 1, 0, 0]$
- $D2 = [0, 0, 1, 1, 0, 1, 1]$

Cosine Distance



$$D1 \cdot D2 = 1 \times 0 + 1 \times 0 + 1 \times 1 + 1 \times 1 + 1 \times 0 + 0 \times 1 + 0 \times 1 = 2$$

Second, we calculate the magnitude of the vectors:

$$||D1|| = \sqrt{1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2} = \sqrt{5}$$

$$||D2|| = \sqrt{0^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 1^2} = \sqrt{4}$$

Finally, cosine similarity can be calculated by dividing the dot product by the magnitude

$$similarity(D1, D2) = \frac{D1 \cdot D2}{||D1||||D2||} = \frac{2}{\sqrt{5}\sqrt{4}} = \frac{2}{\sqrt{20}} = 0.44721$$

The angle between the vectors is calculated as:

$$cos(\theta) = 0.44721$$

$$\theta = \arccos(0.44721) = 63.435$$

R Tool

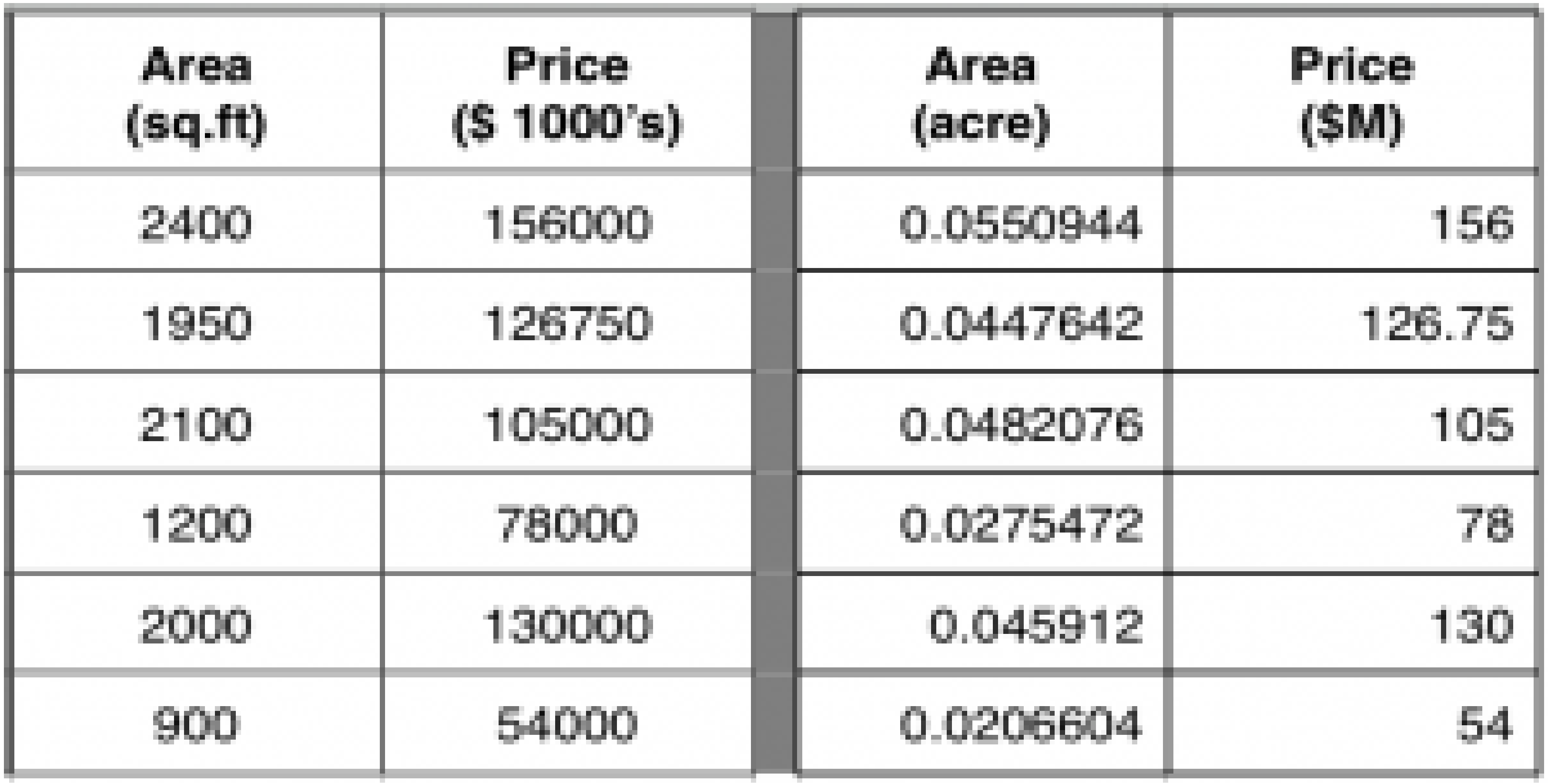
- x <- c(33, 33, 43, 55, 48, 37, 43, 24)
- y <- c(37, 38, 42, 46, 46, 59, 41, 50)
- library(lsa)
- cosine(x, y)
- [1,] 0.9624844
- #OR use
- cossim <- function(A,B) { (sum(A*B))/sqrt((sum(A^2))*(sum(B^2))) }

Mahalanobis Distance

- distance between a test point and the mean/distribution
- used to determine whether a sample is an outlier
- Idea: to calculate the covariance (or dispersion) matrix of each class to identify the relative distance between the two attributes from their centroid
- centroid = a base or central point that is the overall mean for multivariate data.

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$

‘area’ and ‘price’ of the same objects – Euclidean distance gives a different value.
Euclidean distance will work fine as long as the dimensions are equally weighted and are independent of each other.



Area (sq.ft)	Price ($ 1000’s)	Area (acre)	Price ($M)
2400	156000	0.0550944	156
1950	126750	0.0447642	126.75
2100	105000	0.0482076	105
1200	78000	0.0275472	78
2000	130000	0.045912	130
900	54000	0.0206604	54

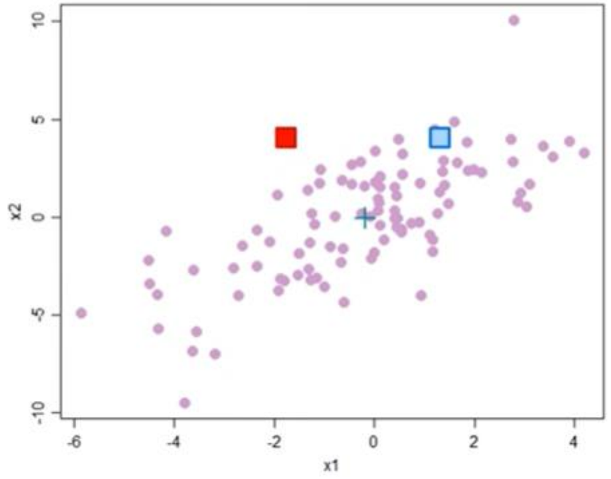
Mahalanobis Distance

- The two tables show the ‘area’ and ‘price’ of the same objects in different units of the variables change.
- Technically -> Since both tables represent the same entities, the distance between any two rows, point A and point B should be the same.
- But Euclidean distance gives a different value even though the distances are technically the same in physical space.
- This can technically be overcome by scaling the variables, by computing the z-score (ex: (x – mean) / std) or make it vary within a particular range like between 0 and 1.
- Now find Euclidean distance.

Mahalanobis Distance

- if the dimensions are correlated to one another the Euclidean distance between a point and the center of the points (distribution) can give little or misleading information about how close a point really is to the cluster.
- Euclidian Distance of both p1 and p2 w.r.t. mean is same, even though p1 (red) is outlier, as data has covariance.
- Covariance – variance matrix can be calculated for multivariate data.

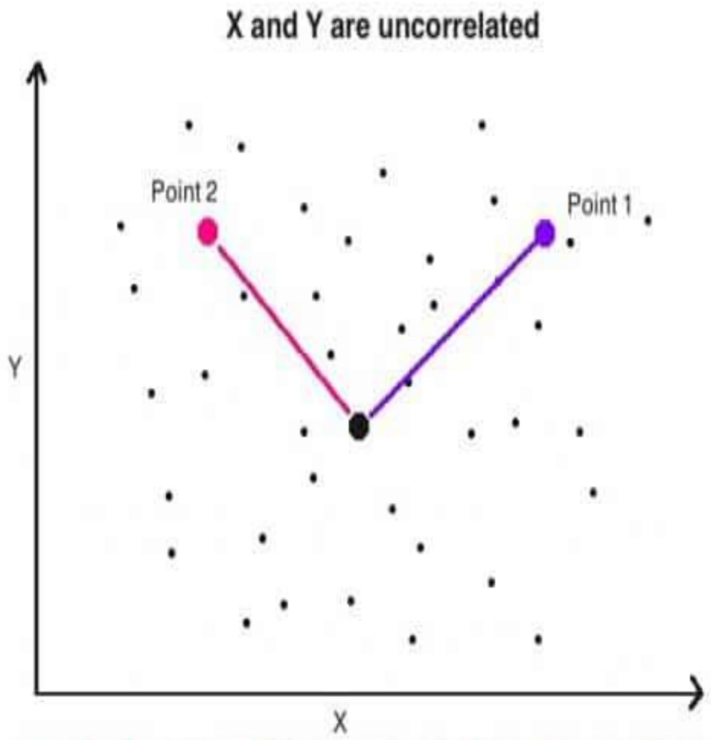
However, Euclidean distance has limitations in real datasets, which often have some degree of covariance



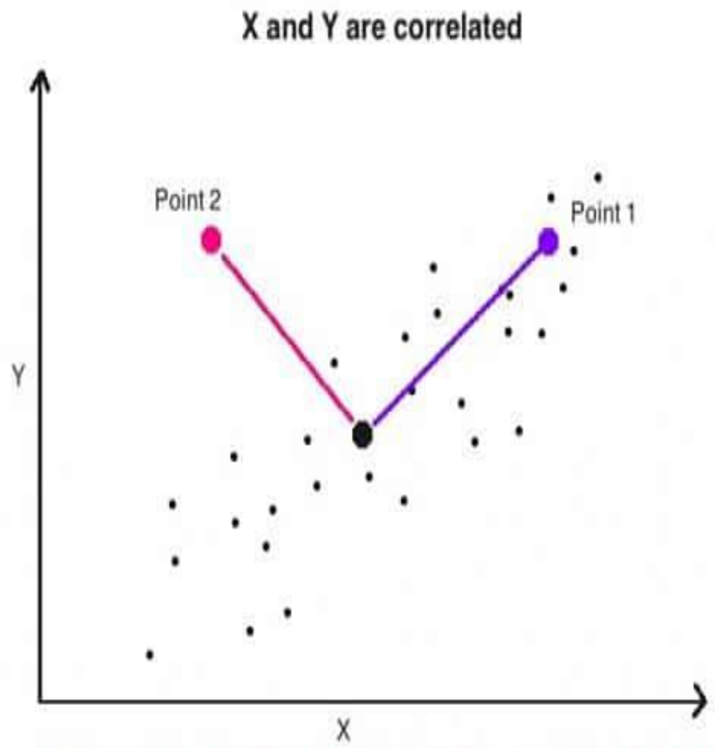
$$\begin{array}{c|ccc} & \boldsymbol{x1} & \boldsymbol{x2} & \boldsymbol{x3} \\ \hline \boldsymbol{x1} & cov(x1,x1) & cov(x1,x2) & cov(x1,x3) \\ \boldsymbol{x2} & cov(x2,x1) & cov(x2,x2) & cov(x2,x3) \\ \boldsymbol{x3} & cov(x3,x1) & cov(x3,x2) & cov(x3,x3) \end{array}$$

$$\begin{array}{c|ccc} & \boldsymbol{x1} & \boldsymbol{x2} & \boldsymbol{x3} \\ \hline \boldsymbol{x1} & var(x1) & cov(x1,x2) & cov(x1,x3) \\ \boldsymbol{x2} & cov(x2,x1) & var(x2) & cov(x2,x3) \\ \boldsymbol{x3} & cov(x3,x1) & cov(x3,x2) & var(x3) \end{array}$$

Mahalanobis Distance



When X and Y are uncorrelated, the Euclidean distance from the Centroid can be useful to infer if a point is member of the distribution. The farther it is, the less likely it is a member.



Both Point 1 and Point 2 have the same Euclidean distance from centroid. But only Point 1 is a member of the distribution. To detect Point 2 as outlier, dist(Point 2, centroid) should be much higher than dist(Point 1, Centroid). Mahalanobis distance can be used here instead.

Mahalanobis Distance

- It transforms the columns into uncorrelated variables
- Scale the columns to make their variance equal to 1
- Finally, it calculates the Euclidean distance.
- If the variables in your dataset are strongly correlated, then, the covariance will be high. Dividing by a large covariance will effectively reduce the distance.
- If the X's are not correlated, then the covariance is not high and the distance is not reduced much.

2x2 covariance matrix

$$S = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

$\frac{1}{m}\sum_{i=1}^{m}(x_1 - \mu_1)^2$	$\frac{1}{m}\sum_{i=1}^{m}(x_1 - \mu_1)(x_2 - \mu_2)$
$\frac{1}{m}\sum_{i=1}^{m}(x_2 - \mu_2)(x_1 - \mu_1)$	$\frac{1}{m}\sum_{i=1}^{m}(x_2 - \mu_2)^2$

Inverse of 2x2 covariance matrix

Assuming no correlation, our covariance matrix is:

$$S = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

The inverse of a 2x2 matrix can be found using the following:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \qquad A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Applying this to get the inverse of the covariance matrix:

$$S^{-1} = \frac{1}{\sigma_1^2 \sigma_2^2} \begin{bmatrix} \sigma_2^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix}$$

Mahalanobis Distance

$$d_M(x, y) = \sqrt{(x-y)^T S^{-1} (x-y)}$$

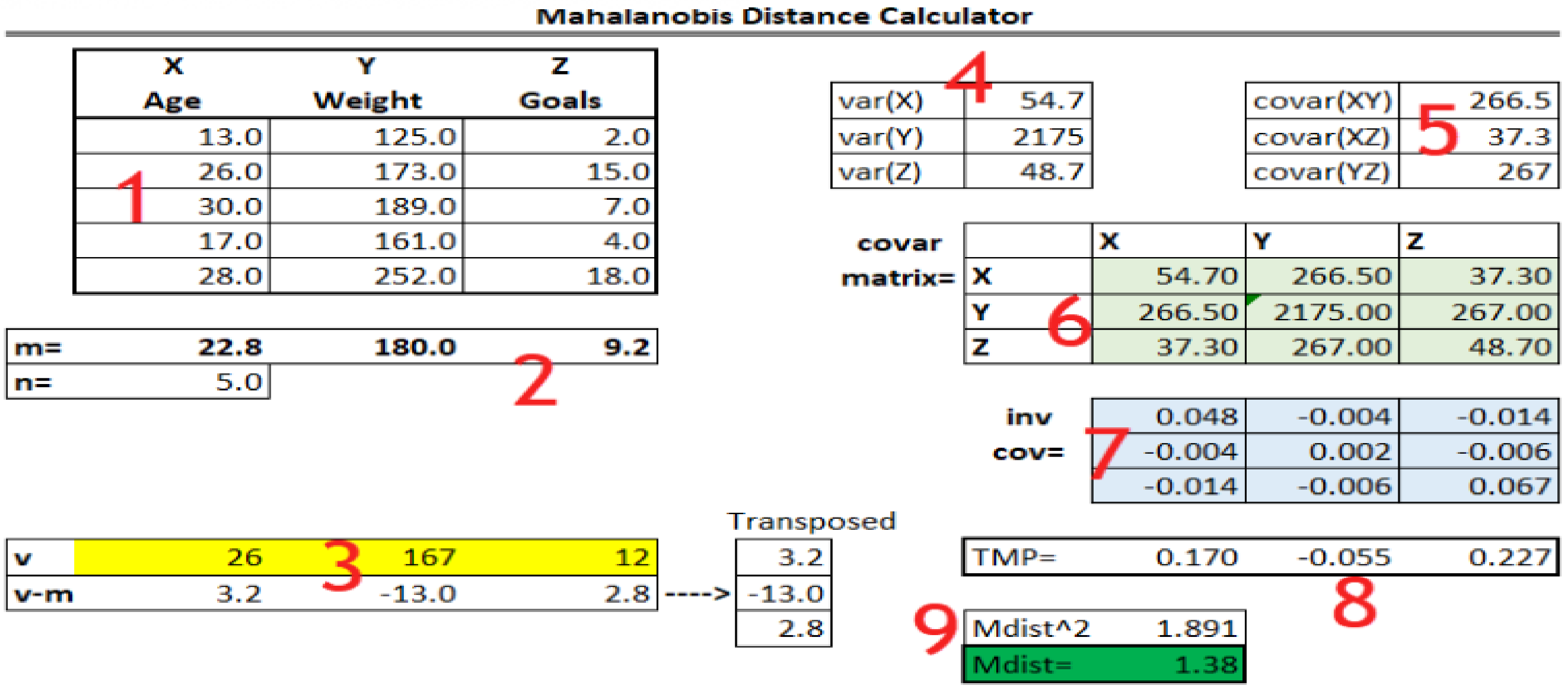
$$= \sqrt{\begin{bmatrix} x_1 - y_1 & x_2 - y_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \end{bmatrix}}$$

$$= \sqrt{\begin{bmatrix} \frac{x_1 - y_1}{\sigma_1^2} & \frac{x_2 - y_2}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \end{bmatrix}}$$

$$= \sqrt{\frac{(x_1 - y_1)^2}{\sigma_1^2} + \frac{(x_2 - y_2)^2}{\sigma_2^2}}$$

Mahalanobis Distance Between v and given distribution

$$D^2 = (x - m)^T \cdot C^{-1} \cdot (x - m)$$



Mahalanobis Distance Calculator

X Age	Y Weight	Z Goals
13.0	125.0	2.0
26.0	173.0	15.0
30.0	189.0	7.0
17.0	161.0	4.0
28.0	252.0	18.0

m=	22.8	180.0	9.2
n=	5.0		

var(X)	54.7
var(Y)	2175
var(Z)	48.7

covar(XY)	266.5
covar(XZ)	37.3
covar(YZ)	267

covar matrix=

	X	Y	Z
X	54.70	266.50	37.30
Y	266.50	2175.00	267.00
Z	37.30	267.00	48.70

inv cov=

0.048	-0.004	-0.014
-0.004	0.002	-0.006
-0.014	-0.006	0.067

v	26	167	12
v-m	3.2	-13.0	2.8

----> Transposed

3.2
-13.0
2.8

TMP=	0.170	-0.055	0.227

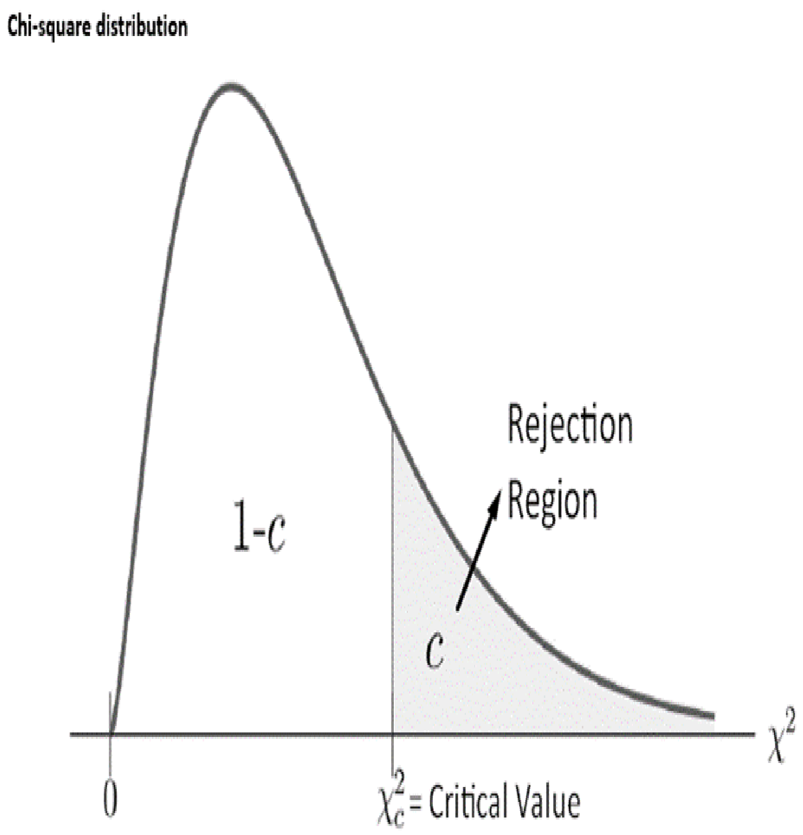
Mdist^2	1.891
Mdist=	1.38

8) TMP = multiply the v-m vector with the inverse covariance matrix.
9) Mdist = multiply the TMP and transposed v-m vectors

Case Study: Mahalanobis Distance - finding outliers for multivariate data in R

- To find the outliers we need to find distance between each point and the center/mean value of every variable in multivariate data.
- E.g "airquality."

1. Find the center point of "Ozone" and "Temp."
2. Calculate the covariance matrix of "Ozone" and "Temp."
3. Find the Mahalanobis distance of each point to the center.
4. Find the cut-off value from chi-square distribution.
5. Select the distances that are less than cut-off. These are the values that aren't outliers.



Mahalanobis Distance - finding outliers for multivariate data in R

chi-square (X2) critical value -

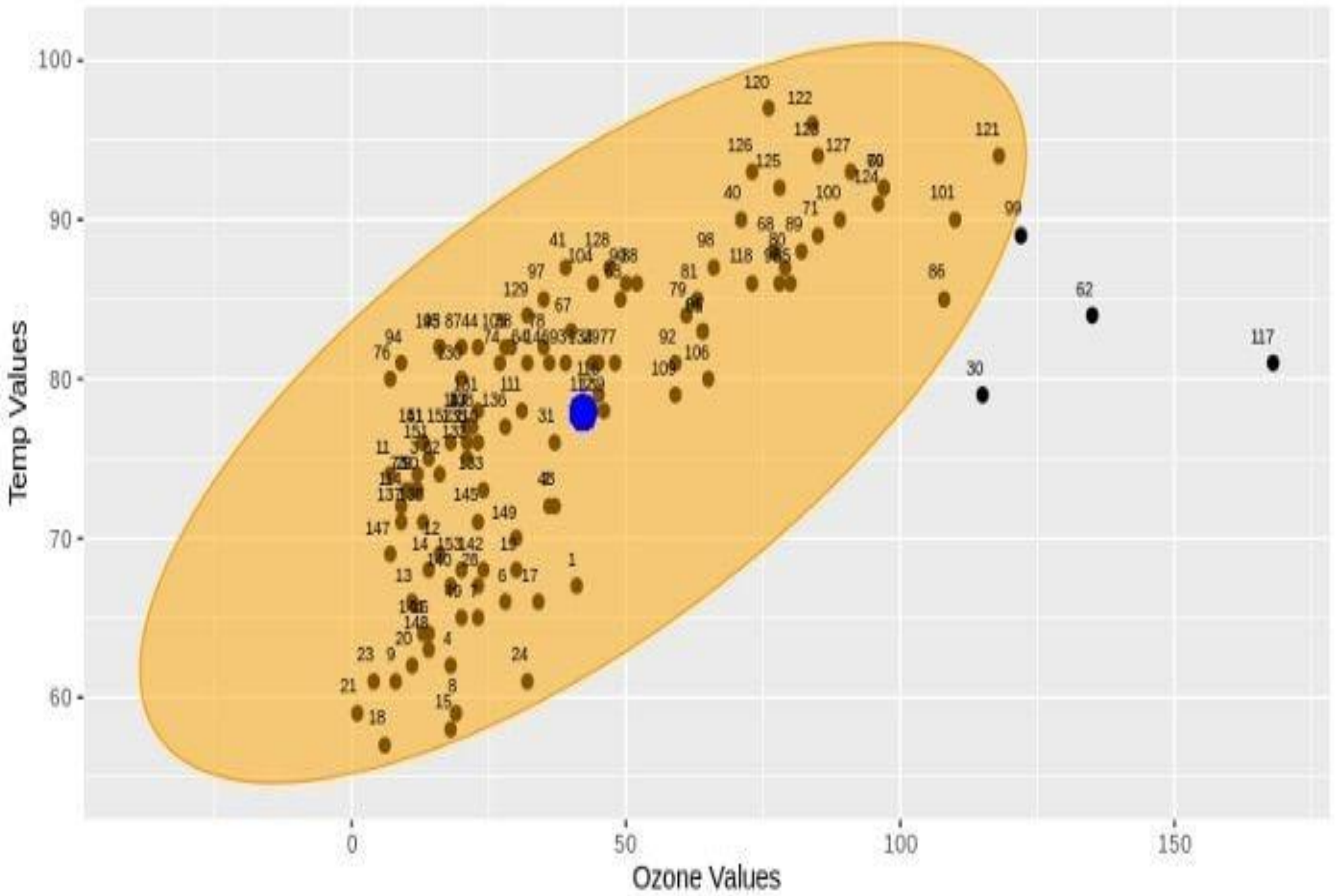
- statistical value used in chi-square test to determine whether differences between observed and expected data are significant
- $\chi^2 = \sum [(Oi - Ei)^2 / Ei]$, Oi denotes actual number of observations within a category, Ei symbolizes the expected frequency
- a threshold that helps in determining if relationship between categorical variables in a study is real or if it could likely occur randomly
- determined based on degree of freedom (df=number of categories or groups minus one) and significance level (alpha-0.05/0.1/0.01)
- Alpha-probability of rejecting null hypothesis (H0) when it is true
- Use a chi-square distribution table to find critical value of chi-square for the degrees of freedom and level of significance.
- Find intersection of the row corresponding to the DOF and column corresponding to chosen significance level (alpha).

Mahalanobis Distance - finding outliers for multivariate data in R

- # Select only Ozone and Temp variables
- air = airquality[c("Ozone" , "Temp")]
- # We need to remove NA from data set
- air = na.omit(air)
- # Finding the center point
- air.center = colMeans(air)
- # Finding the covariance matrix
- air.cov = cov(air)

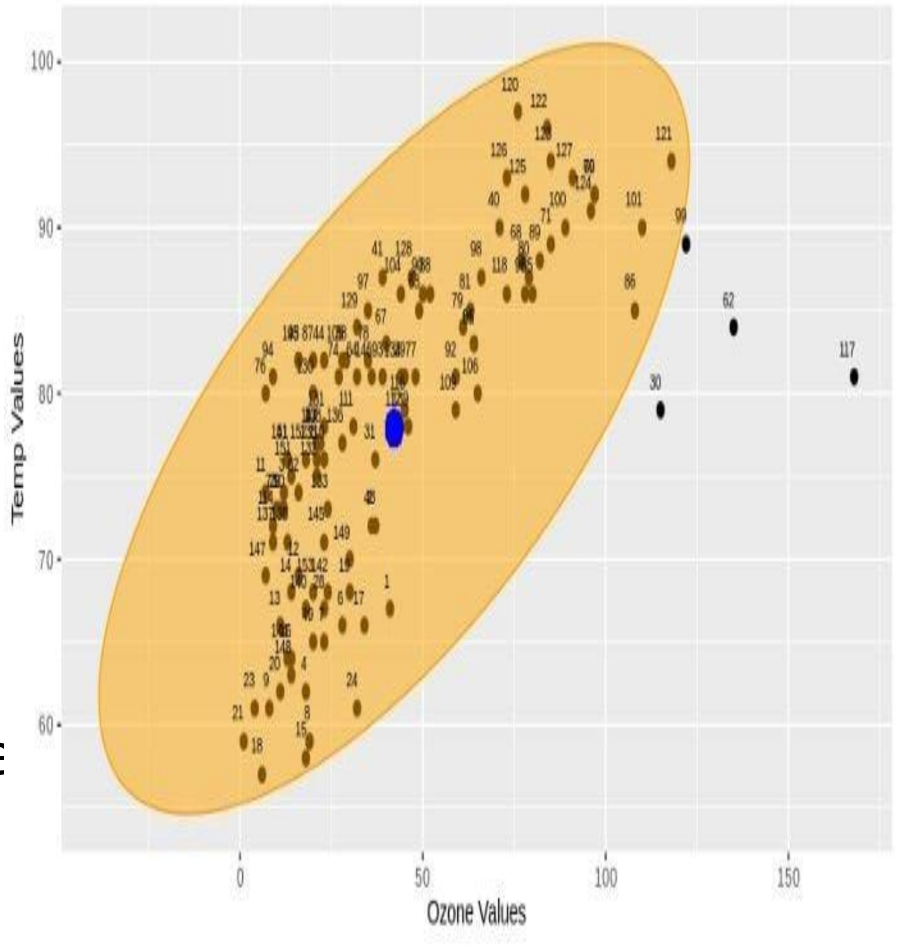
- # Call the package
- library(ggplot2)
- # Ellipse coordinates names should be same with air data set
- ellipse <- as.data.frame(ellipse)
- colnames(ellipse) <- colnames(air)
- # Create scatter Plot
- figure <- ggplot(air , aes(x = Ozone , y = Temp)) +
- geom_point(size = 2) +
- geom_polygon(data = ellipse , fill = "orange" , color = "orange" , alpha = 0.5)+
- geom_point(aes(air.center[1] , air.center[2]) , size = 5 , color = "blue") +
- geom_text(aes(label = row.names(air)) , hjust = 1 , vjust = -1.5 ,size = 2.5) +
- ylab("Temp Values") + xlab("Ozone Values")
- # Run and display plot
- figure

Plot Data



Mahalanobis Distance - finding outliers for multivariate data in R

- # Finding distances
- distances <- mahalanobis(x = air , center = air.center , cov = air.cov)
- # Cutoff value for ditances from Chi-Sqaure Dist.
- # with p = 0.95 df = 2 which in ncol(air)
- cutoff <- qchisq(p = 0.95 , df = ncol(air))
- ## Display observation whose distance greater than cutoff value
- air[distances > cutoff ,]
- ## Returns 4 Outliers: 30. 62. 99. 117. observations



Thank You

- **Submit all assignments.**