Drive- https://drive.google.com/drive/folders/1YXtgyqfvdhu55FcrcidE1iWOw7bT9IM-

Practical 1 - hypothesis

> x= c(6.2, 6.6, 7.1, 7.4, 7.6, 7.9, 8, 8.3, 8.4, 8.5, 8.6,8.8, 8.8, 9.1, 9.2, 9.4, 9.4, 9.7, 9.9, 10.2, 10.4,

10.8,11.3, 11.9) #Entering the data

> t.test(x-9, alternative="two.sided", conf.level=0.95) #Performing the t-test

Two-sample

> x=c(418,421,421,422,425,427,431,434,437,439,446,447,448,453,454,463,465) # Entering the

data

> y=c(429,430,430,431,36,437,440,441,445,446,447) # Entering the data

> t.test(x, y, alternative="two.sided", mu=0, var.equal=F, conf.level=0.95) #Performing the t-test

Practical 2 - time series

> data("AirPassengers") #checks if dataset “AirPassengers” is available

> class(AirPassengers) #checks type of dataset(here it is ts=timeseries)

> start(AirPassengers) #returns start of dataset

> end(AirPassengers) #returns end of dataset

> summary(AirPassengers) #to know min, max, median, 1st & 3rd quartiles

> frequency(AirPassengers)

> plot(AirPassengers)

> abline(reg=lm(AirPassengers~time(AirPassengers))) #for best-fit-line

> cycle(AirPassengers)

> boxplot(AirPassengers~cycle(AirPassengers))

Practical 3 - couchdb

> install.packages('sofa')

> library('sofa')

> x=Cushion$new()

> x$ping()

> db\_create(x, dbname='ty')

> db\_list(x)

> doc1 = '{"rollno":"01", "name":"masha", "grade":"a"}'

> doc\_create(x, doc1, dbname="ty", docid="a\_1")

> doc2 = '{"rollno":"02", "name":"SCOOBY", "grade":"A"}'

> doc\_create(x, doc2, dbname="ty", docid="a\_2")

> doc3 = '{"rollno":"03", "name":"spike", "grade":"b", "remark":"pass"}'

> doc\_create(x, doc3, dbname="ty", docid="a\_3")

> db\_changes(x, 'ty')

> db\_query(x, dbname='ty', selector=list('\_id'=list('$gt'=NULL)))$docs

> db\_query(x, dbname='ty', selector=list(grade='a'))$docs

> db\_query(x, dbname='ty', selector=list(remark='pass'))$docs

> db\_query(x, dbname='ty', selector=list(rollno=list('$gt'='02')),fields=c('name','grade'))$docs

> library('jsonlite') #load library "jsonlite"

> res = db\_query(x, dbname='ty', selector=list('\_id'=list('$gt'=NULL)),fields=c('name','rollno','grade','remark'),as='json')

> fromJSON(res)$docs #display JSON doc in dataframe

> doc\_delete(x, dbname='ty', docid='a\_2')

> doc\_get(x, dbname='ty', docid='a\_2')

> docNEW='{"diva":"delve", "happy":"hours", "chirpy":"ours"}'

> doc\_update(x, dbname='ty', doc=docNEW, docid='a\_3', rev='1-5c28df6228ece9abb505107e601caeee')

> docNEW='{"name":"beekeeper", "rollno":"09", "skills":"beginner"}'

> doc\_update(x, dbname='ty', doc=docNEW, docid='a\_1', rev='1-664cdfdc0b6473dc29db24517bb12265')

Practical 4 - linear regression

> lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients

> plot(iris$Petal.Length, iris$Petal.Width, pch=21,

bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",

ylab="Petal width")

> abline(lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients, col="black")

> lm(Petal.Width ~ Petal.Length, data=iris)$coefficients

> plot(iris$Petal.Length, iris$Petal.Width, pch=21,

bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",

ylab="Petal width")

> abline(lm(Petal.Width ~ Petal.Length, data=iris)$coefficients, col="black")

> summary(lm(Petal.Width ~ Petal.Length, data=iris))

> plot(iris$Sepal.Width, iris$Sepal.Length, pch=21,

bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Sepal Width",

ylab="Sepal Length")

> abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")

> summary(lm(Sepal.Length ~ Sepal.Width, data=iris))

> plot(iris$Sepal.Width, iris$Sepal.Length, pch=21,

bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",

ylab="Sepal length")

> abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")

> abline(lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="setosa"),])$coefficients,

col="red")

> abline(lm(Sepal.Length ~ Sepal.Width,

data=iris[which(iris$Species=="versicolor"),])$coefficients, col="green3")

> abline(lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="virginica"),])$coefficients,

col="blue")

> lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="setosa"),])$coefficients

> lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="versicolor"),])$coefficients

> lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="virginica"),])$coefficients

> lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients

> lm(Sepal.Length ~ Sepal.Width:Species + Species, data=iris)$coefficients

> summary(lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris))

> summary(step(lm(Sepal.Length ~ Sepal.Width \* Species, data=iris)))

Practical 5 - logistics regression

> library(datasets)

> ir\_data=iris

> head(ir\_data)

> str(ir\_data)

> levels(ir\_data$Species)

> sum(is.na(ir\_data))

> ir\_data=ir\_data[1:100,]

> set.seed(100)

> samp=sample(1:100,80)

> ir\_test=ir\_data[samp,]

> ir\_ctrl=ir\_data[-samp,]

> install.packages("ggplot2")

> library(ggplot2)

> install.packages("GGally")

> library(GGally)

> install.packages("zeallot")

> install.packages("backports")

> ggpairs(ir\_test)

> y=ir\_test$Species; x=ir\_test$Sepal.Length

> glfit=glm(y~x, family = 'binomial')

> summary(glfit)

> newdata=data.frame(x=ir\_ctrl$Sepal.Length)

> predicted\_val=predict(glfit, newdata, type="response")

> prediction=data.frame(ir\_ctrl$Sepal.Length, ir\_ctrl$Species,predicted\_val)

> prediction

> qplot(prediction[,1], round(prediction[,3]), col=prediction[,2], xlab = 'Sepal Length', ylab=

'Prediction using Logistic Reg.')

Practical 6 - PCA

data\_iris <- iris[1:4]

Cov\_data <- cov(data\_iris )

# Find out the eigenvectors and eigenvalues using the covariance matrix

Eigen\_data <- eigen(Cov\_data)

# Using the inbuilt function

PCA\_data <- princomp(data\_iris ,cor="False")

# Let’s now compare the output variances

Eigen\_data$values

PCA\_data$sdev^2

PCA\_data$loadings[,1:4]

Eigen\_data$vectors

summary(PCA\_data)

biplot (PCA\_data)

screeplot(PCA\_data, type="lines")

#Select the first principal component for the second model

model2 = PCA\_data$loadings[,1]

#For the second model, we need to calculate scores by multiplying our loadings with the data

model2\_scores <- as.matrix(data\_iris) %\*% model2

#Loading libraries for naiveBayes model

library(class)

install.packages("e1071")

library(e1071)

#Fitting the first model over the entire data

mod1<-naiveBayes(iris[,1:4], iris[,5])

#Fitting the second model using the first principal component

mod2<-naiveBayes(model2\_scores, iris[,5])

# Accuracy for the first model

table(predict(mod1, iris[,1:4]), iris[,5])

# Accuracy for the second model

table(predict(mod2, model2\_scores), iris[,5])

Practical 7 - ANOVA

> data("warpbreaks")

> head(warpbreaks)

> summary(warpbreaks)

> model1<-aov(breaks~wool+tension,data=warpbreaks)

> summary(model1)

> plot(model1)

> model2<-aov(breaks~wool+tension+wool:tension,data=warpbreaks)

> summary(model2)

> plot(model2)

Hit <Return> to see next plot: 4

Practical 8 - clustering

> library(datasets)

> head(iris)

> library(ggplot2)

> ggplot(iris,aes(Petal.Length,Petal.Width,color=Species))+geom\_point()

> set.seed(15)

> iriscluster<-kmeans(iris[,3:4],3,nstart=15)

> iriscluster

> table(iriscluster$cluster,iris$Species)

> iriscluster$cluster<-as.factor(iriscluster$cluster)

> ggplot(iris,aes(Petal.Length,Petal.Width,color=iriscluster$cluster))+geom\_point()

> ggplot(iris,aes(Petal.Length,Petal.Width,color=iriscluster$cluster,shape=iriscluster$cluster))+geom\_point()

Practical 9 - decision tree

> iris

> str(iris)

> library(rpart)

> ctree1=rpart(Species~.,data=iris)

> install.packages("rpart.plot")

> library(rpart.plot)

> rpart.plot(ctree1,extra="auto")

Practical 10 - mongodb

> use inventory

>db.product.insert({"productno":"p1","description":"earphone","company":"oneplus","price":

8000});

>db.product.insert({"productno":"p2","description":"desktop","company":"apple","price":120

000});

>db.product.insert({"productno":"p3","description":"smartwatch","company":"samsung","pric

e":17000});

>db.product.insert({"productno":"p4","description":"speaker","company":"jbl","price":13000}

);

>db.product.insert({"productno":"p5","description":"smartphone","company":"samsung","price":14000});

> db.product.find();

> db.product.find().pretty();

> db.product.find({productno:"p3"});

> db.product.find({company:"samsung"});

> db.product.find().sort({"price":1}).pretty();

> db.product.find().sort({"price":-1}).pretty();

> db.product.find({"price":{$gt:15000}}).pretty();

> db.product.update({"productno": "p1"},{$set:{"price":10000}});

> db.product.find({"productno":"p1"}).pretty();

> db.dropDatabase()

remove

> db.books.remove({"isbnno": 10201}, 1);

limit

> db.books.find().limit(3);