Probability Theory and Applications (MA208)

Department of MACS, NITK, Surathkal

Assignment Submission

Topic : Solve 5 problems on applications of probability methods and statistics using R programming Language

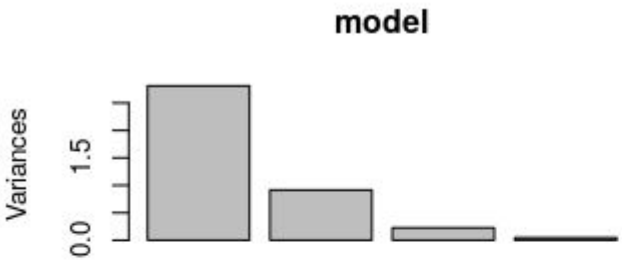
Academic Year 2018-19 6th Semester

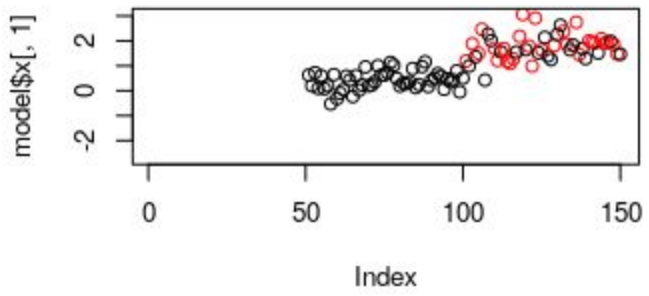
Instructor : Mr. Kedarnath Senapati

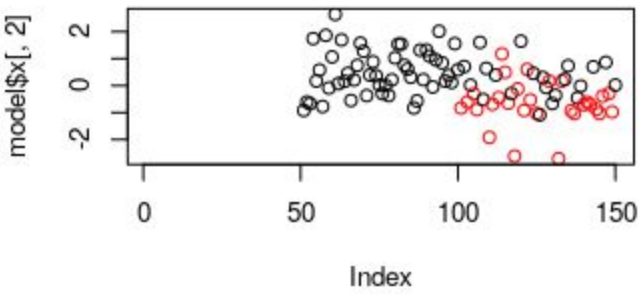
Submitted By : Vanshika Gupta (16CV245)
B. Tech 3rd year, NITK Surathkal

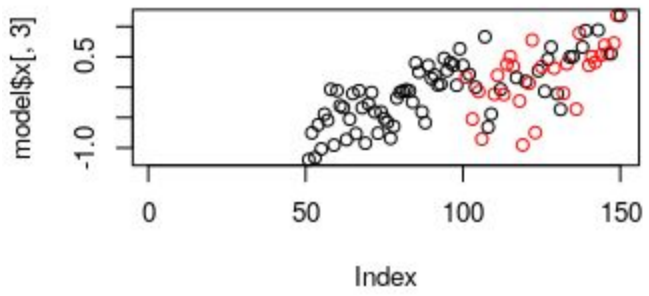
Problem 1 : Principal Component Analysis (PCA)is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set. It has various applications in machine learning as well as image analysis. Write the implementation using R using the IRIS dataset.

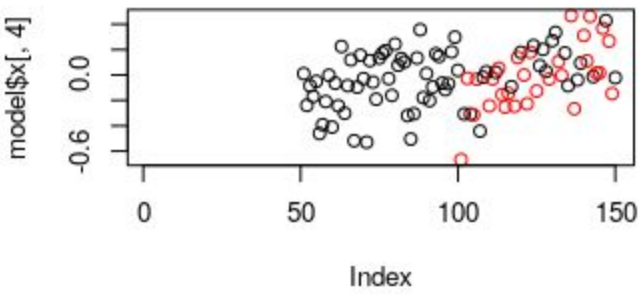
```
#implementation of the principal component analysis using the IRIS dataset
input = read.csv("/home/dell/Desktop/R_probability/Iris.csv")
names(input)
str(input)

#using the prcomp function
model = prcomp(input[,1:4], scale=TRUE)
model$sdev
model$rotation
model$center
model$scale
par(mfrow=c(2,2))
plot(model$x[,1], col=input[,5])
plot(model$x[,2], col=input[,5])
plot(model$x[,3], col=input[,5])
plot(model$x[,4], col=input[,5])
model$sdev^2 / sum(model$sdev^2)
plot(model)

## Normalize the input feature. (without using prcomp fucntion)
input$sepal_len1 = (input$SepalLengthCm - mean(input$SepalLengthCm) )/sd(input$SepalLengthCm)
input$sepal_wid1 = (input$SepalWidthCm - mean(input$SepalWidthCm))/sd(input$SepalWidthCm)
input$petal_len1 = (input$PetalLengthCm - mean(input$PetalLengthCm))/sd(input$PetalLengthCm)
input$petal_wid1 = (input$PetalWidthCm - mean(input$PetalWidthCm))/sd(input$PetalWidthCm)

##Get the covarience matrix and eigen vector.
matrix_form = matrix(c(input$sepal_len1, input$sepal_wid1, input$petal_len1,
input$petal_wid1), ncol=4)
m = cov(matrix_form)
eigenV = eigen(m)
eigenV$vectors
```



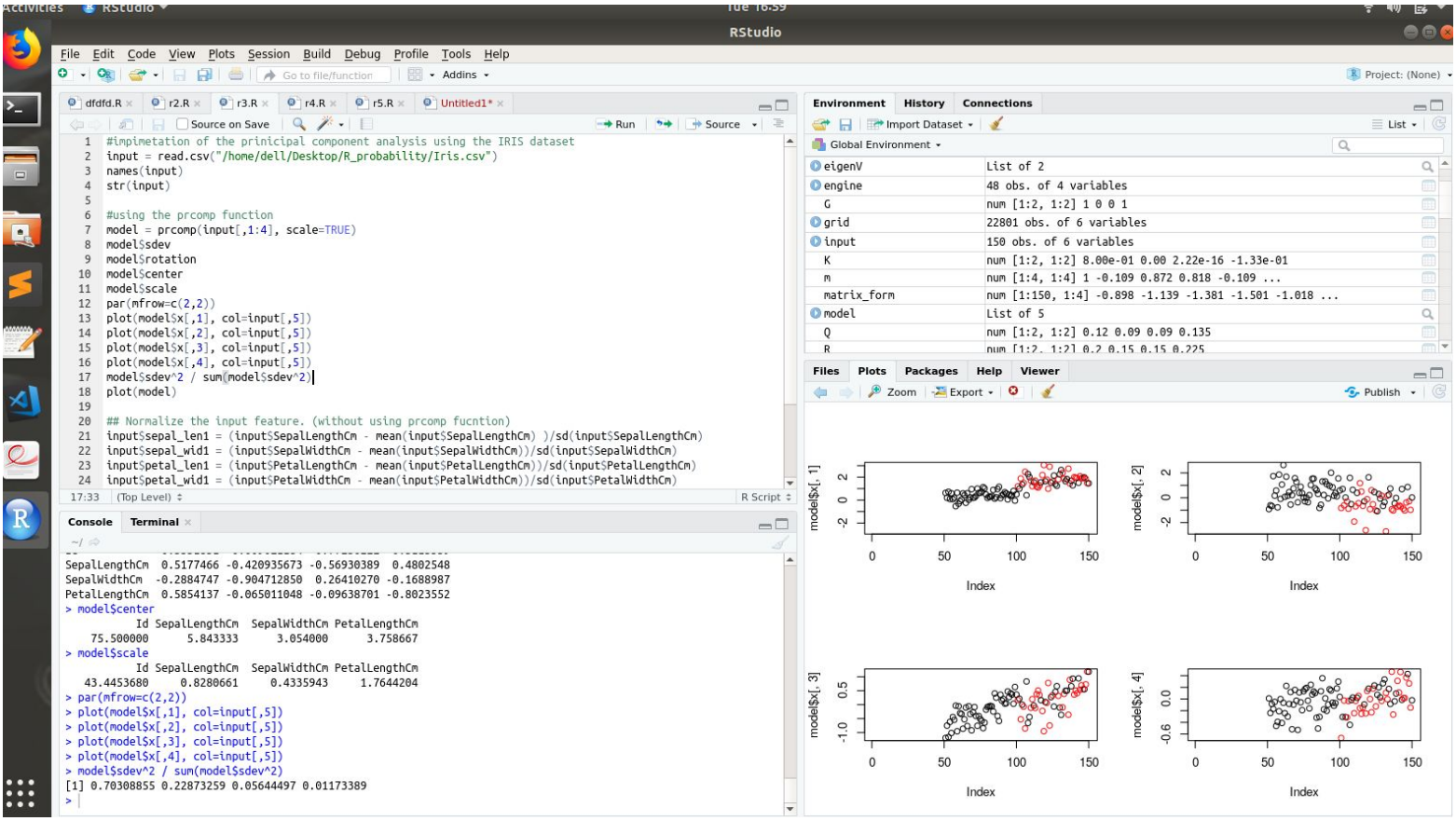






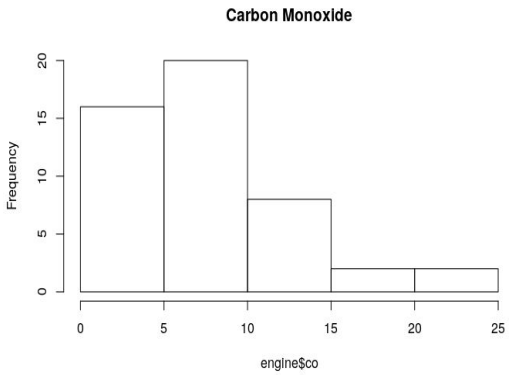


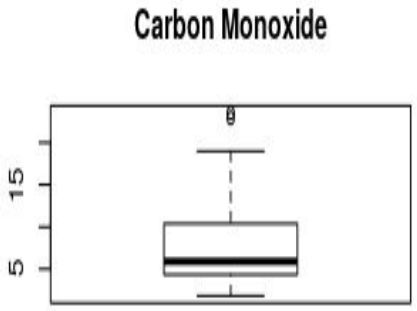
Stimulation was done using RStudio as seen in the screenshot:

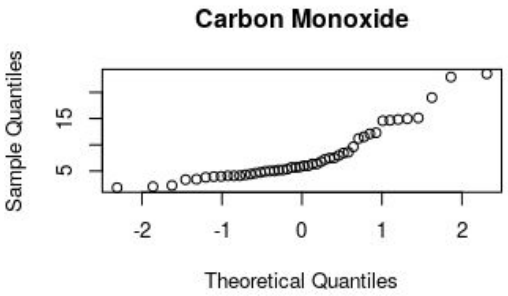


Problem 2 : Given data regarding the carbon monoxide emissions of a place. Compute the statistics related to it. Calculate the mean, and fix a distribution to suit the distribution of the data.

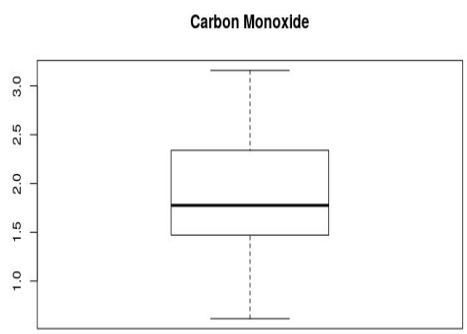
```
engine <- read.csv("/home/dell/Desktop/R_probability/table_7_3.csv",sep=",",head=TRUE)
summary(engine)
qqnorm(engine$co,main="Carbon Monoxide")
qqline(engine$co)
boxplot(engine$co,main="Carbon Monoxide")
hist(engine$co,main="Carbon Monoxide")
qqnorm(engine$co,main="Carbon Monoxide")
qqline(engine$co)
lengine <- log(engine$co)
boxplot(lengine,main="Carbon Monoxide")
hist(lengine,main="Carbon Monoxide")
qqnorm(lengine,main="QQ Plot for the Log of the Carbon Monoxide")
qqline(lengine)
m <- mean(lengine)
s <- sd(lengine)
n <- length(lengine)
se <- s/sqrt(n)
error <- se*qt(0.975,df=n-1)
left <- m-error
right <- m + error
lNull <- log(5.4) - error
rNull <- log(5.4) + error
```

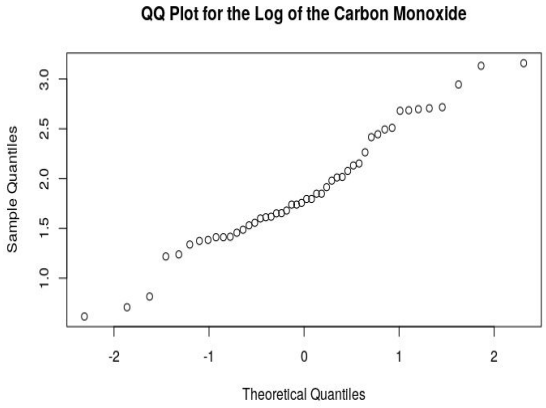


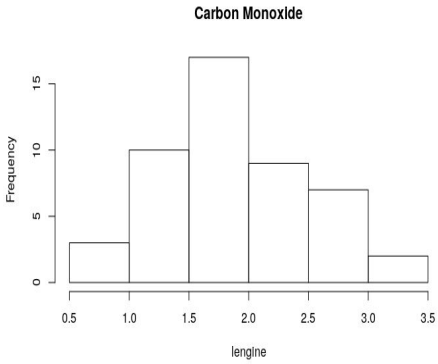




These are the distributions when logarithm is not take.

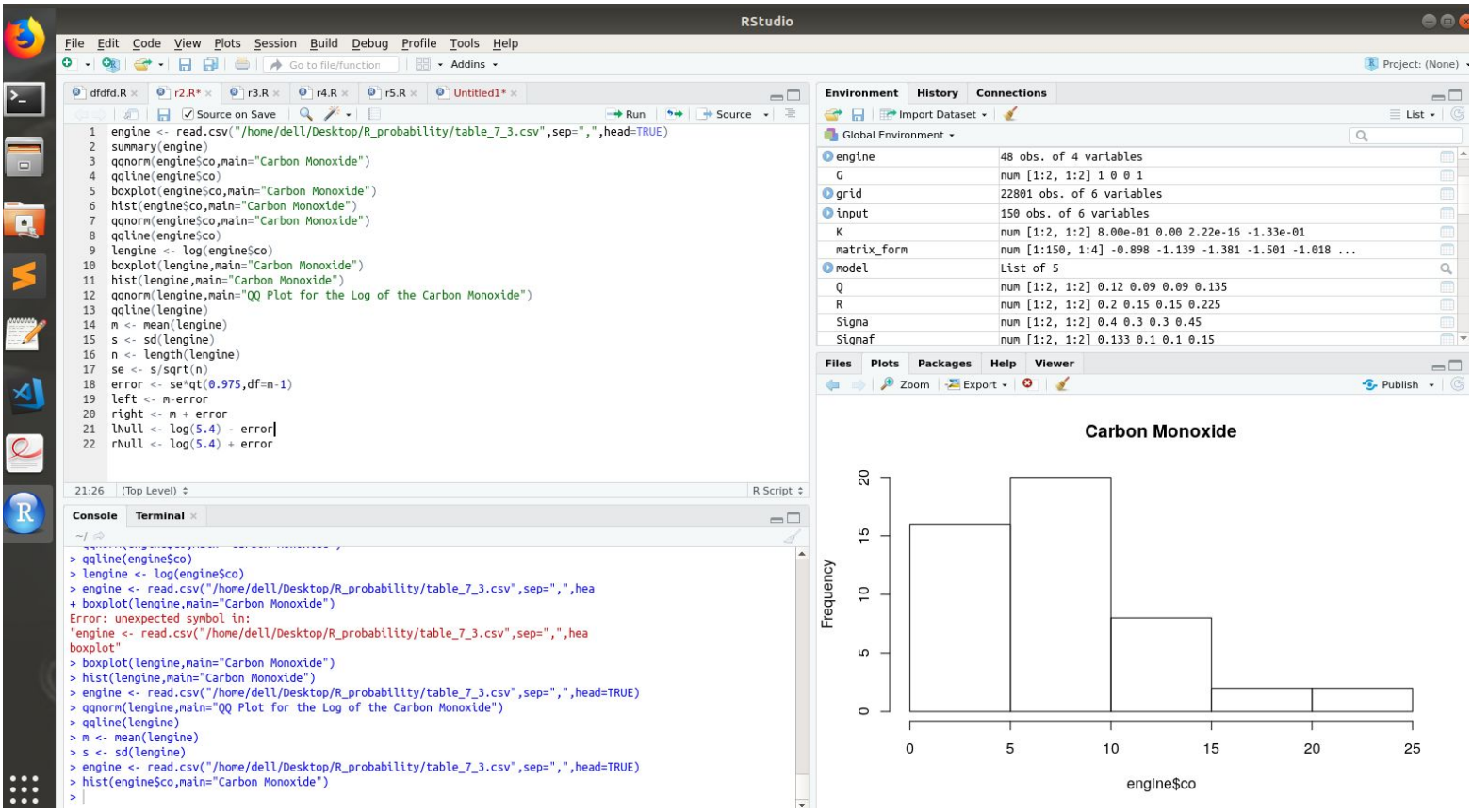






These are the distribution plots when the distribution has been transformed to logarithmic one. The plots gain a normalized and standard distribution.

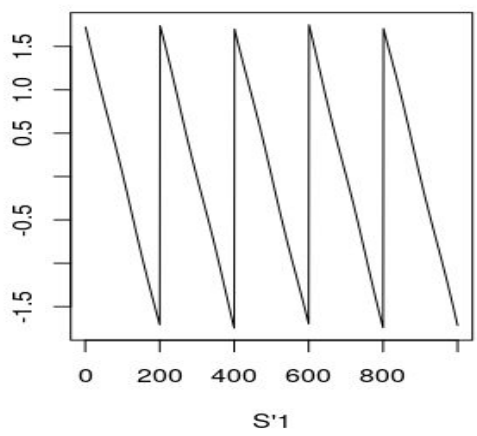
It can be seen in the above code that we have also computed the mean, variance, skewness and errors associated with the distribution.

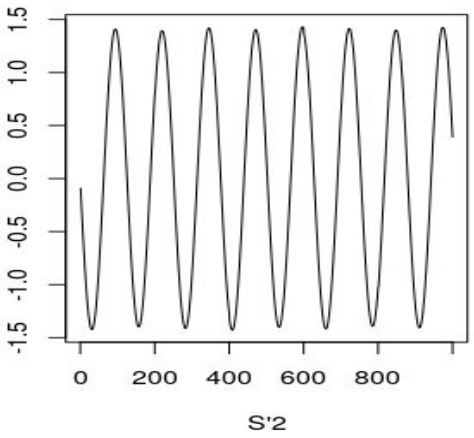


The above stimulation done on RStudio.

Problem 3: Implement the Independent Component Analysis Algorithm used for various applications in machine learning and image dimensionality reduction. Given a mixed signal, separate the signal into its constituent components.

```
library('fastICA')

# Source matrix
S <- cbind(sin((1:1000)/20), rep((((1:200)-100)/100), 5))
# Mixing matrix
A <- matrix(c(0.291, 0.6557, -0.5439, 0.5572), 2, 2)
# plot graphs
par(mfcol = c(1, 2))
plot(1:1000, S[,1], type = "l",xlab = "S1", ylab = "")
plot(1:1000, S[,2], type = "l", xlab = "S2", ylab = "")
# Mixed two signals
X <- S %*% A

par(mfcol = c(1, 2))
plot(1:1000, X[,1], type = "l",xlab = "X1", ylab = "")
plot(1:1000, X[,2], type = "l", xlab = "X2", ylab = "")
a <- fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
           method = "R", row.norm = FALSE, maxit = 200,
           tol = 0.0001, verbose = TRUE)

par(mfcol = c(1, 2))
plot(1:1000, a$S[,1], type = "l", xlab = "S'1", ylab = "")
plot(1:1000, a$S[,2], type = "l", xlab = "S'2", ylab = "")
```



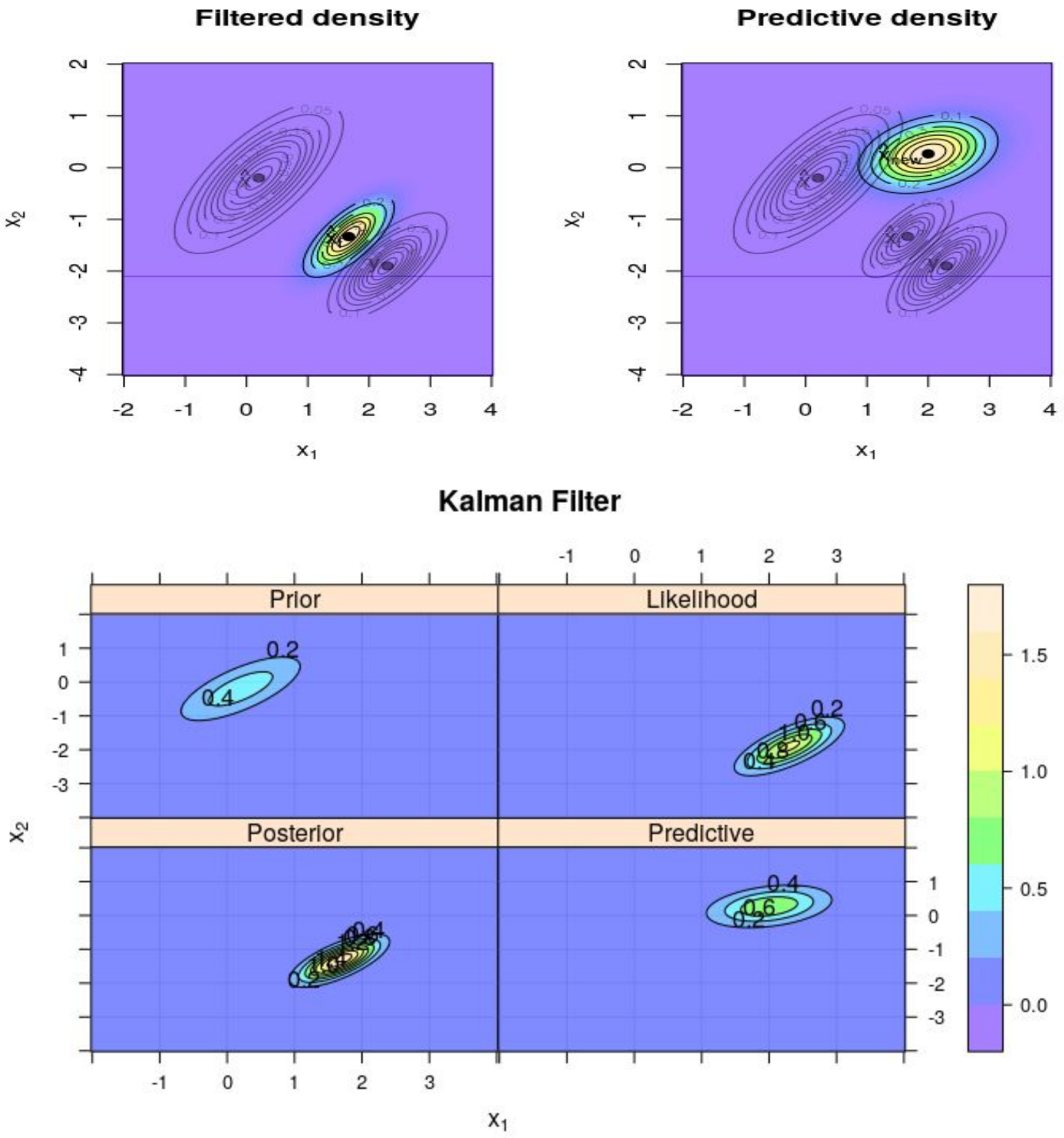


The signals as computed from the ICA algorithm.

Problem 4 : Estimate the state of a dynamic systems by using Bayesian Filtering techniques. The popular Bayesian filters are: Kalman Filter. Show a visualisation of the Kalman filter using R.

```
library(mnormt)
mycols <- topo.colors(100,0.5)
xhat <- c(0.2, -0.2)
Sigma <- matrix(c(0.4, 0.3,
                  0.3, 0.45), ncol=2)
x1 <- seq(-2, 4,length=151)
x2 <- seq(-4, 2,length=151)
f <- function(x1,x2, mean=xhat, varcov=Sigma)
  dmnorm(cbind(x1,x2), mean,varcov)
z <- outer(x1,x2, f)
image(x1,x2,z, col=mycols, main="Prior density",
      xlab=expression('x'[1]), ylab=expression('x'[2]))
contour(x1,x2,z, add=TRUE)
points(0.2, -0.2, pch=19)
text(0.1, -0.2, labels = expression(hat(x)), adj = 1)


R <- 0.5 * Sigma
z2 <- outer(x1,x2, f, mean=c(2.3, -1.9), varcov=R)
image(x1, x2, z2, col=mycols, main="Sensor density")
contour(x1, x2, z2, add=TRUE)
points(2.3, -1.9, pch=19)
text(2.2, -1.9, labels = "y", adj = 1)
contour(x1, x2,z, add=TRUE)
points(0.2, -0.2, pch=19)
text(0.1, -0.2, labels = expression(hat(x)), adj = 1)

G = diag(2)
y <- c(2.4, -1.9)
xhatf <- xhat + Sigma %*% t(G) %*% solve(G %*% Sigma %*% t(G) + R) %*% (y - G %*% xhat)
Sigmaf <- Sigma - Sigma %*% t(G) %*% solve(G %*% Sigma %*% t(G) + R) %*% G %*% Sigma
z3 <- outer(x1, x2, f, mean=c(xhatf), varcov=Sigmaf)
image(x1, x2, z3, col=mycols,
      xlab=expression('x'[1]), ylab=expression('x'[2]),
      main="Filtered density")
contour(x1, x2, z3, add=TRUE)
points(xhatf[1], xhatf[2], pch=19)
text(xhatf[1]-0.1, xhatf[2],
      labels = expression(hat(x)[f]), adj = 1)
lb <- adjustcolor("black", alpha=0.5)
contour(x1, x2, z, add=TRUE, col=lb)
points(0.2, -0.2, pch=19, col=lb)
text(0.1, -0.2, labels = expression(hat(x)), adj = 1, col=lb)
contour(x1, x2, z2, add=TRUE, col=lb)
points(2.3, -1.9, pch=19, col=lb)
text(2.2, -1.9,labels = "y", adj = 1, col=lb)

A <- matrix(c(1.2, 0,
```

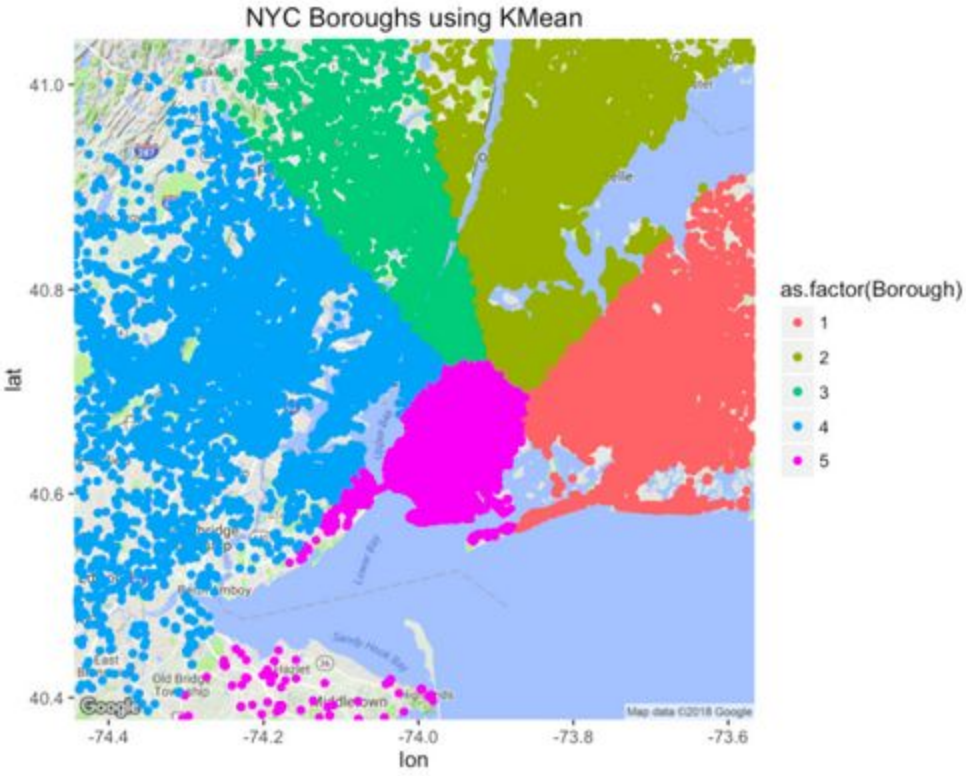
```
             0, -0.2), ncol=2)
Q <- 0.3 * Sigma
K <- A %*% Sigma %*% t(G) %*% solve(G%*% Sigma %*% t(G) + R)
xhatnew <- A %*% xhat + K %*% (y - G %*% xhat)
Sigmanew <- A %*% Sigma %*% t(A) - K %*% G %*% Sigma %*% t(A) + Q
z4 <- outer(x1,x2, f, mean=c(xhatnew), varcov=Sigmanew)
image(x1, x2, z4, col=mycols,
      xlab=expression('x'[1]), ylab=expression('x'[2]),
      main="Predictive density")
contour(x1, x2, z4, add=TRUE)
points(xhatnew[1], xhatnew[2], pch=19)
text(xhatnew[1]-0.1, xhatnew[2],
      labels = expression(hat(x)[new]), adj = 1)
contour(x1, x2, z3, add=TRUE, col=lb)
points(xhatf[1], xhatf[2], pch=19, col=lb)
text(xhatf[1]-0.1, xhatf[2], col=lb,
      labels = expression(hat(x)[f]), adj = 1)
contour(x1, x2, z, add=TRUE, col=lb)
points(0.2, -0.2, pch=19, col=lb)
text(0.1, -0.2, labels = expression(hat(x)), adj = 1, col=lb)
contour(x1, x2, z2, add=TRUE, col=lb)
points(2.3, -1.9, pch=19, col=lb)
text(2.2, -1.9,labels = "y", adj = 1, col=lb)


## Plot all stages with lattice
library(lattice)
grid <- expand.grid(x=x1,y=x2)
grid$Prior <- as.vector(z)
grid$Likelihood <- as.vector(z2)
grid$Posterior <- as.vector(z3)
grid$Predictive <- as.vector(z4)
contourplot(Prior + Likelihood + Posterior + Predictive ~ x*y,
             data=grid, col.regions=mycols, region=TRUE,
             as.table=TRUE,
             xlab=expression(x[1]),
             ylab=expression(x[2]),
             main="Kalman Filter",
             panel=function(x,y,...){
             panel.grid(h=-1, v=-1)
             panel.contourplot(x,y,...)
             })
```



This is a visualisation of the code of the Kalman filter used in signal processing using RStudio .

Problem : Given a dataset of the Uber sales. Compute the clustering using kmeans and segregate the data using this classification technique using R.

```
# Load the .csv files
apr14 <-
read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/master/uber-trip-data
/uber-raw-data-apr14.csv")
may14 <-
read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/master/uber-trip-data
/uber-raw-data-may14.csv")
jun14 <-
read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/master/uber-trip-data
/uber-raw-data-jun14.csv")
jul14 <-
read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/master/uber-trip-data
/uber-raw-data-jul14.csv")
aug14 <-
read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/master/uber-trip-data
/uber-raw-data-aug14.csv")
sep14 <-
read.csv("https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/master/uber-trip-data
/uber-raw-data-sep14.csv")
library(dplyr)
data14 <- bind_rows(apr14, may14, jun14, jul14, aug14, sep14)
# VIM library for using 'aggr'
library(VIM)

# 'aggr' plots the amount of missing/imputed values in each column
aggr(data14)
library(lubridate)

# Separate or mutate the Date/Time columns
data14$Date.Time <- mdy_hms(data14$Date.Time)
data14$Year <- factor(year(data14$Date.Time))
data14$Month <- factor(month(data14$Date.Time))
data14$Day <- factor(day(data14$Date.Time))
data14$Weekday <- factor(wday(data14$Date.Time))
data14$Hour <- factor(hour(data14$Date.Time))
data14$Minute <- factor(minute(data14$Date.Time))
data14$Second <- factor(second(data14$Date.Time))
set.seed(20)
clusters <- kmeans(data14[,2:3], 5)

# Save the cluster number in the dataset as column 'Borough'
data14$Borough <- as.factor(clusters$cluster)
str(clusters)
library(ggmap)

NYCMap <- get_map("New York", zoom = 10)
ggmap(NYCMap) + geom_point(aes(x = Lon[], y = Lat[], colour = as.factor(Borough)),data = data14) +
  ggtitle("NYC Boroughs using KMean")
```



This is the result produced after the k means clustering.

********** END
****