Figure 6.4, Figure 6.5, Figure 6.6, Figure6.8

Segbehoe, Lawrence Sethor

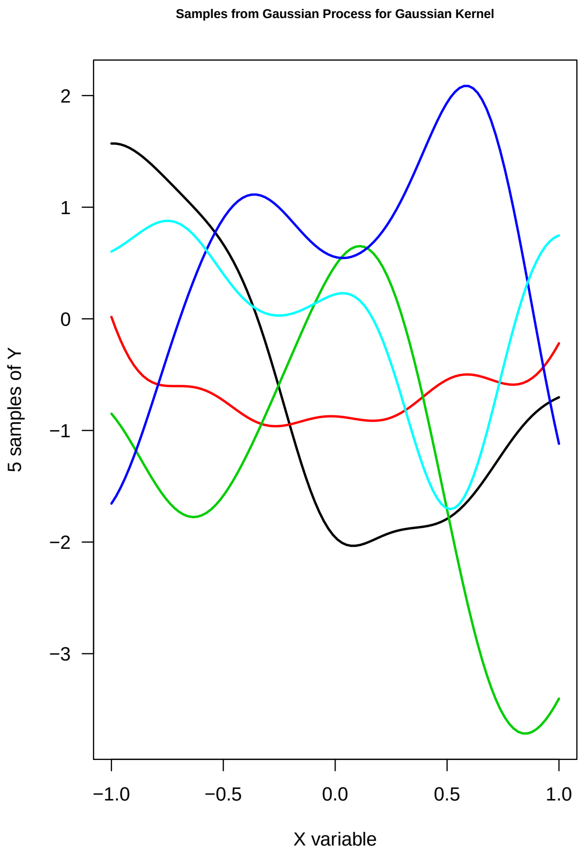
October 20, 2018

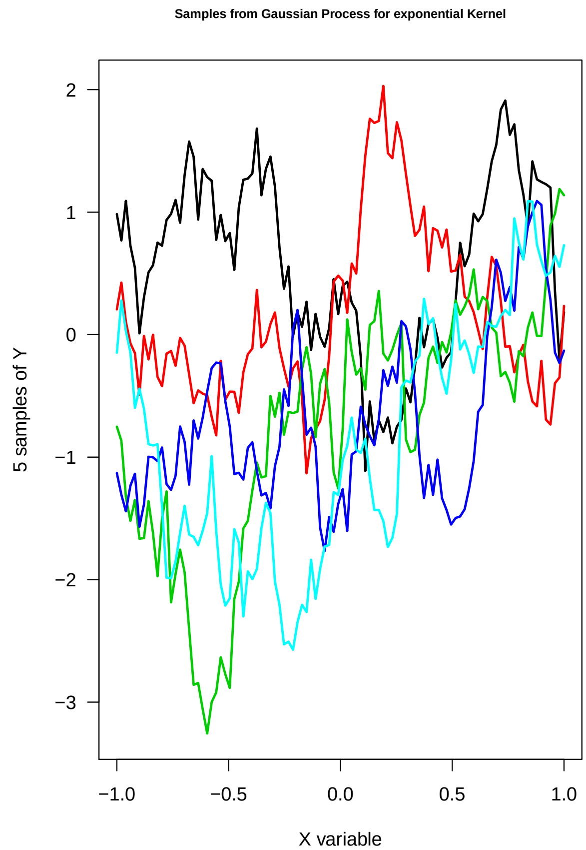
Gaussian Process

Plotting the Gaussian process graphs of 6.5

```
###############################################################################
####################  Gaussian Process Regression (GPR)  #####################
###############################################################################


## Function for Figure Six(6)
Gaussian.Process.Figure6.4 = function(X = seq(-1, 1, length.out = 100)){

  library(mvtnorm)
## Creating x values
X = c(X)

mu = rep(0, length(X))

set.seed(125)
K.gram.Gaussian = matrix(NA, nrow = length(mu), ncol = length(mu))
SIG = 0.1
for (i in 1:length(mu)) {
  for (j in 1: length(mu)) {
    ## Gaussian Kernel
    K.gram.Gaussian[i,j] = exp(-(1/(2*SIG))*(norm(as.matrix(X[i] - X[j]),
                                                  type = "F"))^2)
  }

}


K.gram.Exponential = matrix(NA, nrow = length(mu), ncol = length(mu))
for (i in 1:length(mu)) {
  for (j in 1: length(mu)) {
    ## Exponential Kernel
    K.gram.Exponential[i,j] = exp(-2*abs(X[i] - X[j]))
  }

}

# is.positive.definite(K.gram[1:5,1:5])
# K.gram[1:5,1:5]


N.sample = 5
Sample.Exponential.kernel = matrix(NA, nrow = length(mu), ncol = N.sample)
for (i in 1:N.sample) {
```

```
  Sample.Exponential.kernel[,i] = rmvnorm(1, mean = mu, sigma = K.gram.Exponential)
}

N.sample = 5
Sample.Gaussian.Kernel = matrix(NA, nrow = length(mu), ncol = N.sample)
for (i in 1:N.sample) {
  Sample.Gaussian.Kernel[,i] = rmvnorm(1, mean = mu, sigma = K.gram.Gaussian )
}

# tail(Sample.Gaussian.Kernel)
# tail(X)

par(mfrow = c(1,2))
plot(X, Sample.Gaussian.Kernel[,1], type="n",
     ylim=c(min(Sample.Gaussian.Kernel), max(Sample.Gaussian.Kernel)),
     ylab="5 samples of Y", las = 1, xlab = "X variable",
     main = "Samples from Gaussian Process for Gaussian Kernel", cex.main = 0.7)
for (i in 1:N.sample) {
 lines(X, Sample.Gaussian.Kernel[, i], col = i, lwd = 2)

}


plot(X, Sample.Exponential.kernel[,1], type="n",
     ylim=c(min(Sample.Exponential.kernel), max(Sample.Exponential.kernel)),
     ylab="5 samples of Y", las = 1, xlab = "X variable",
     main = "Samples from Gaussian Process for exponential Kernel", cex.main = 0.7)
for (i in 1:N.sample) {
 lines(X, Sample.Exponential.kernel[, i], col = i, lwd = 2)

}

par(mfrow = c(1,2))

}

Gaussian.Process.Figure6.4()
```

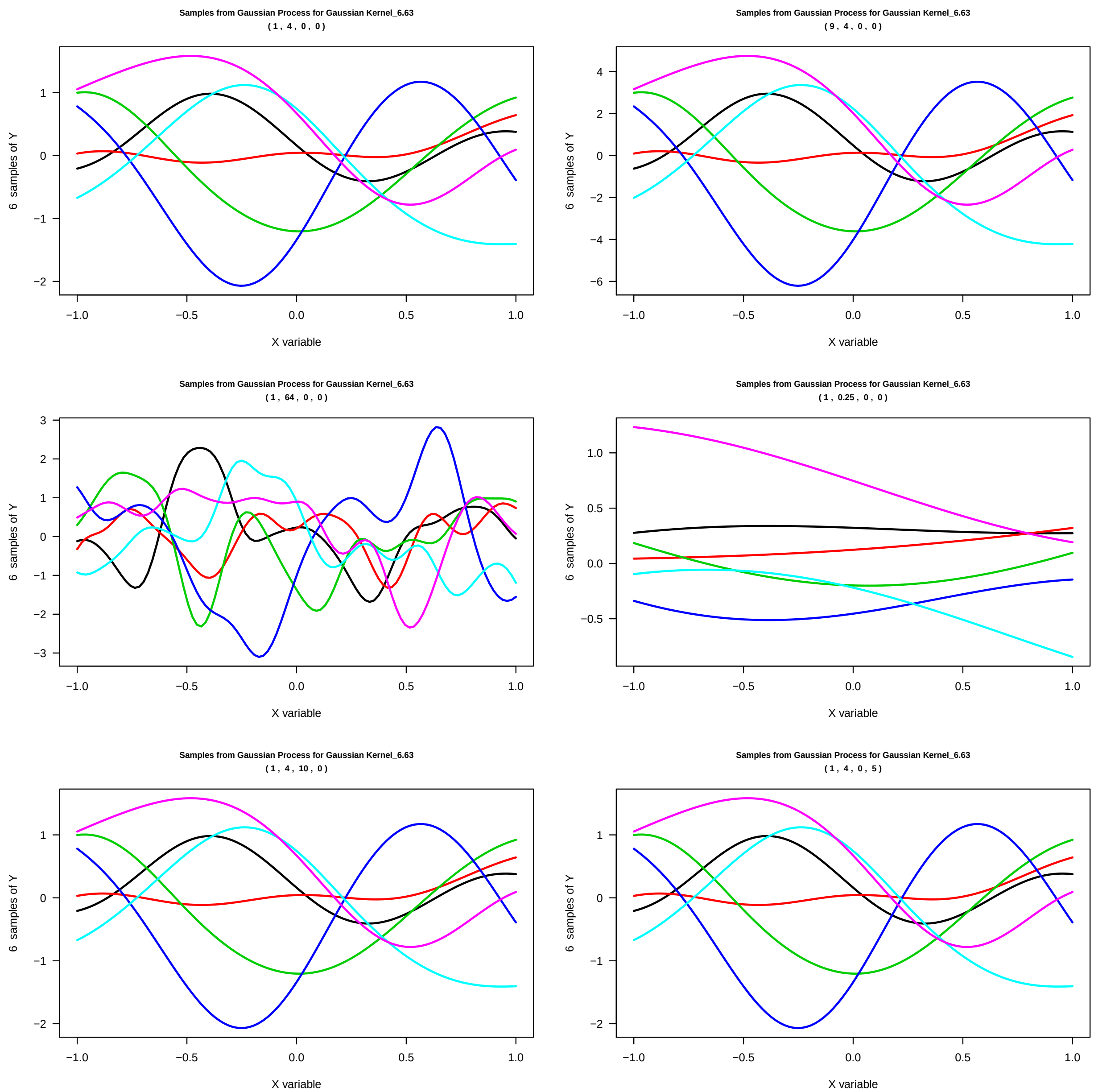




Plotting the Gaussian process graphs of 6.5

```
## Function for Figure Six(6)
Gaussian.Process.Figure6.5 = function(X = seq(-1, 1, length.out = 100),
                                      teta = c(1,36,0,0), N.sample = 5){

## Creating x values
X = c(X)

mu = rep(0, length(X))

set.seed(5)

# extraction thetas from the vector
  teta = as.vector(teta)
  theta_0 = teta[1]
  theta_1 = teta[2]
  theta_2 = teta[3]
  theta_3 = teta[4]

  # container for the gram matrix
  GM <- matrix(NA, ncol = length(mu), nrow = length(mu))

  # a for loop for creating the  Gram Matrix
```

```r
  for (i in 1:length(mu)){
      for(j in 1:length(mu)){
        # Gram Matrix for Kernel in Equation 6.63
        GM[i,j] = (theta_0)*exp((-(theta_1/2)*((norm(as.matrix(X[i]-X[j]),
                                                  type = "F"))^2)))
        + (theta_2) + (theta_3)*(as.matrix(X[i])%*%t(as.matrix(X[j])))
      }
  }




  Norm.dist.sample = matrix(NA, nrow = length(mu), ncol = N.sample)
  for (i in 1:N.sample) {
    Norm.dist.sample[,i] = rmvnorm(1, mean = mu, sigma = GM )
  }



  plot(X, Norm.dist.sample[,1], type="n",
       ylim=c(min(Norm.dist.sample), max(Norm.dist.sample)),
       ylab= paste(N.sample, " samples of Y"), las = 1, xlab = "X variable",
       main = c(paste("Samples from Gaussian Process for Gaussian Kernel_6.63"),
                  paste("(", teta[1],", ", teta[2],", ",teta[3], ", ", teta[4],
                        ")")),
       cex.main = 0.7 )
  for (i in 1:N.sample) {
   lines(X, Norm.dist.sample[, i], col = i, lwd = 2)
  }

}

par(mfrow = c(3,2))
Gaussian.Process.Figure6.5(teta = c(1,4,0,0), N.sample = 6 )
Gaussian.Process.Figure6.5(teta = c(9,4,0,0), N.sample = 6)
Gaussian.Process.Figure6.5(teta = c(1,64,0,0), N.sample = 6)
Gaussian.Process.Figure6.5(teta = c(1,0.25,0,0), N.sample = 6)
Gaussian.Process.Figure6.5(teta = c(1,4,10,0), N.sample = 6)
Gaussian.Process.Figure6.5(teta = c(1,4,0,5), N.sample = 6)
```




```r
par(mfrow = c(1,1))
```

Plotting the Gaussian process graphs of 6.6

```r
# introduce a noise in Y
obs = 100

## Function for Figure Six(6)
Gaussian.Process.Figure6.6 = function(X = seq(-1, 1, length.out = 100),
                                      teta = c(1,36,0,0),
                                      N.sample = 5,
                                      setseed = 10){


## Creating x values
```

```
X = c(X)

mu = rep(0, length(X))

set.seed(setseed)

# extraction thetas from the vector
  teta = as.vector(teta)
  theta_0 = teta[1]
  theta_1 = teta[2]
  theta_2 = teta[3]
  theta_3 = teta[4]

  # container for the gram matrix
  GM <- matrix(NA, ncol = length(mu), nrow = length(mu))

  # a for loop for creating the  Gram Matrix
  for (i in 1:length(mu)){
      for(j in 1:length(mu)){
        # Gram Matrix for Kernel in Equation 6.63
        GM[i,j] = (theta_0)*exp((-(theta_1/2)*((norm(as.matrix(X[i]-X[j]),
                                                  type = "F"))^2)))
        + (theta_2) + (theta_3)*(as.matrix(X[i])%*%t(as.matrix(X[j])))
      }
  }

  ## Matrix C
  beta.inv = 0.3
  C.matrix = GM + beta.inv*diag(length(mu))


  Norm.dist.sample.Yn = matrix(NA, nrow = length(mu), ncol = N.sample)
  for (i in 1:N.sample) {
    Norm.dist.sample.Yn[,i] = rmvnorm(1, mean = mu, sigma = GM )
  }

  Norm.dist.sample.tn = matrix(NA, nrow = length(mu), ncol = N.sample)
  for (i in 1:N.sample) {
    Norm.dist.sample.tn[,i] = rmvnorm(1, mean = mu, sigma = C.matrix)
  }

  indices = sample(1:100, 10)

  Yn = Norm.dist.sample.Yn[  indices,]
  tn = Norm.dist.sample.tn[  indices,]
  Xnew = X[indices]

  return( list(
  plot(X, Norm.dist.sample.tn[,1], type="n",
       ylim=c(min(Norm.dist.sample.tn), max(Norm.dist.sample.tn)),
       ylab= paste(N.sample, " samples of Y"), las = 1, xlab = "X variable",
       main = c(paste("Samples from Gaussian Process for Gaussian Kernel_6.63"),
                  paste("(", teta[1],", ", teta[2],", ",teta[3], ", ", teta[4],
```

```r
                        ")")),
       cex.main = 0.7, sub  = paste("set.seed = ", setseed) ),
  for (i in 1:N.sample) {
   lines(X, Norm.dist.sample.Yn[, i], col = 4, lwd = 2)
  },
  points(Xnew, Yn , col = 2, lwd = 2, cex = 1.5),
  points(Xnew, tn, col = 3, lwd = 2, cex = 1.5),
  segments(x0 = Xnew, y0 = Yn, x1 = Xnew, y1 = tn, col = 6, lty =1, lwd = 2)
  ))
}
par(mfrow = c(2,2) )
Gaussian.Process.Figure6.6(N.sample = 1, setseed = 5)
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
```

```r
Gaussian.Process.Figure6.6(N.sample = 1, setseed = 8)
```

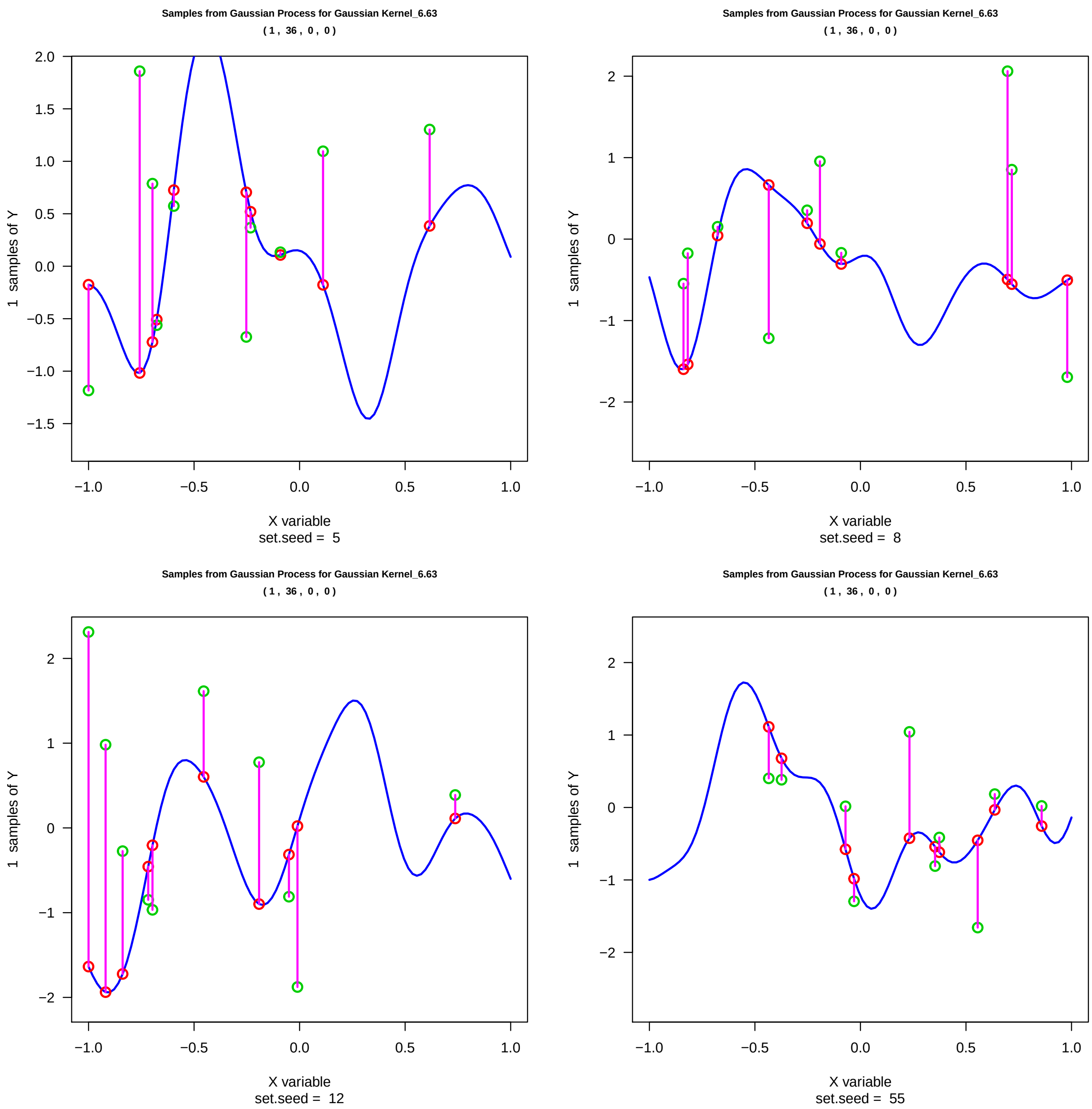
```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
```

```r
Gaussian.Process.Figure6.6(N.sample = 1, setseed = 12)
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
```

```
##
## [[4]]
## NULL
##
## [[5]]
## NULL
```

```r
Gaussian.Process.Figure6.6(N.sample = 1, setseed = 55)
```



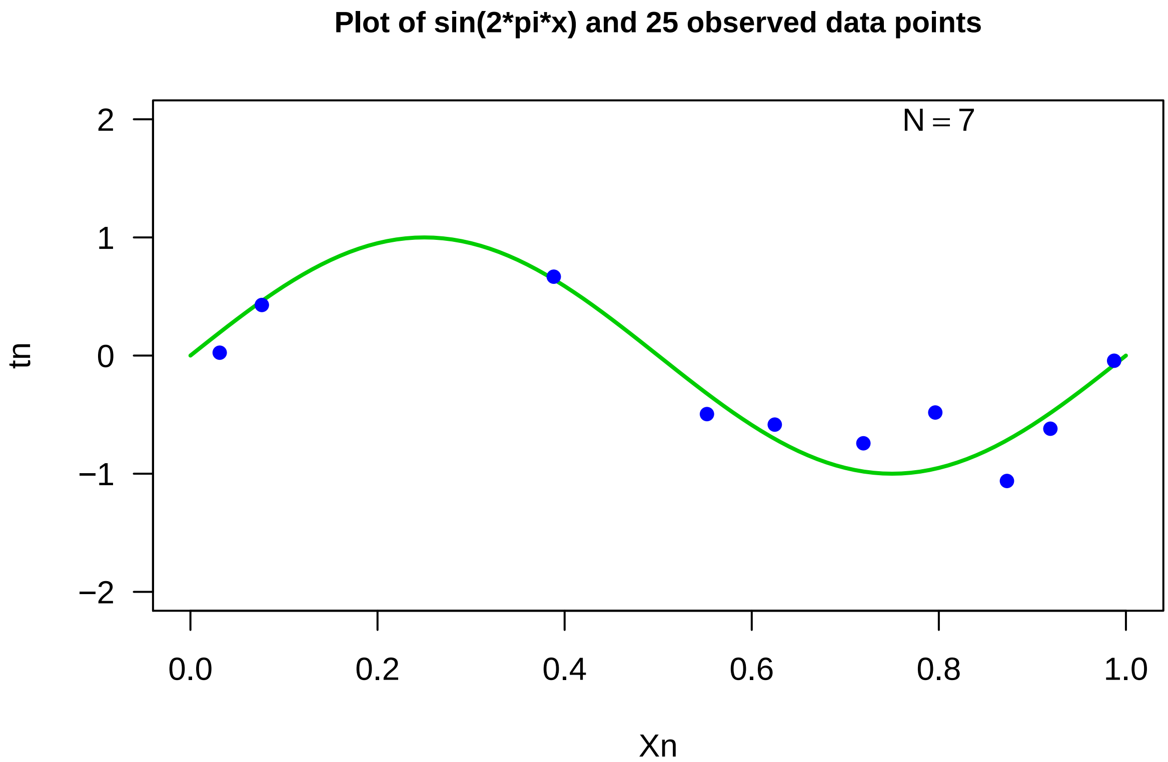

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
```

```
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
```

```r
par(mfrow = c(1,1))
```

Plotting the Gaussian process graphs of 6.8

```r
## Creating a function for sin(2*pi*x)
f <- function(x){sin(2*pi*x)}
## Creating input variable for the function
Xn <- seq(0,1, 0.01)
## making a dataframe for the two sets of data above

tn <- f(Xn)
data_sinx <- data.frame(Xn, tn = f(Xn))

## setting a seed to avoid changing random samples from rnorm()
set.seed(59)


X_training4 <- runif(10 ,min = 0, max = 1)

## Creating the target variable
## Varying the variance parameter in the rnorm function show how far
## the blue points are away from the green curve
sigma_squared <- 0.3
## Generating the target variables based on the training data set and
## Gaussian noise.

t_target4 = f(X_training4) + rnorm(10, 0, sigma_squared)

## making a dataframe form the observed (training and target) dataset


datframe4 = data.frame(X_training4, t_target4)

# dim(data_sinx)

# Plot

plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
main = "Plot of sin(2*pi*x) and 25 observed data points", cex.main = 0.9,
ylim = c(-2,2))
text(.8,2, expression( N == 7))
points(t_target4~X_training4, data = datframe4, col = 4, pch = 16)
```

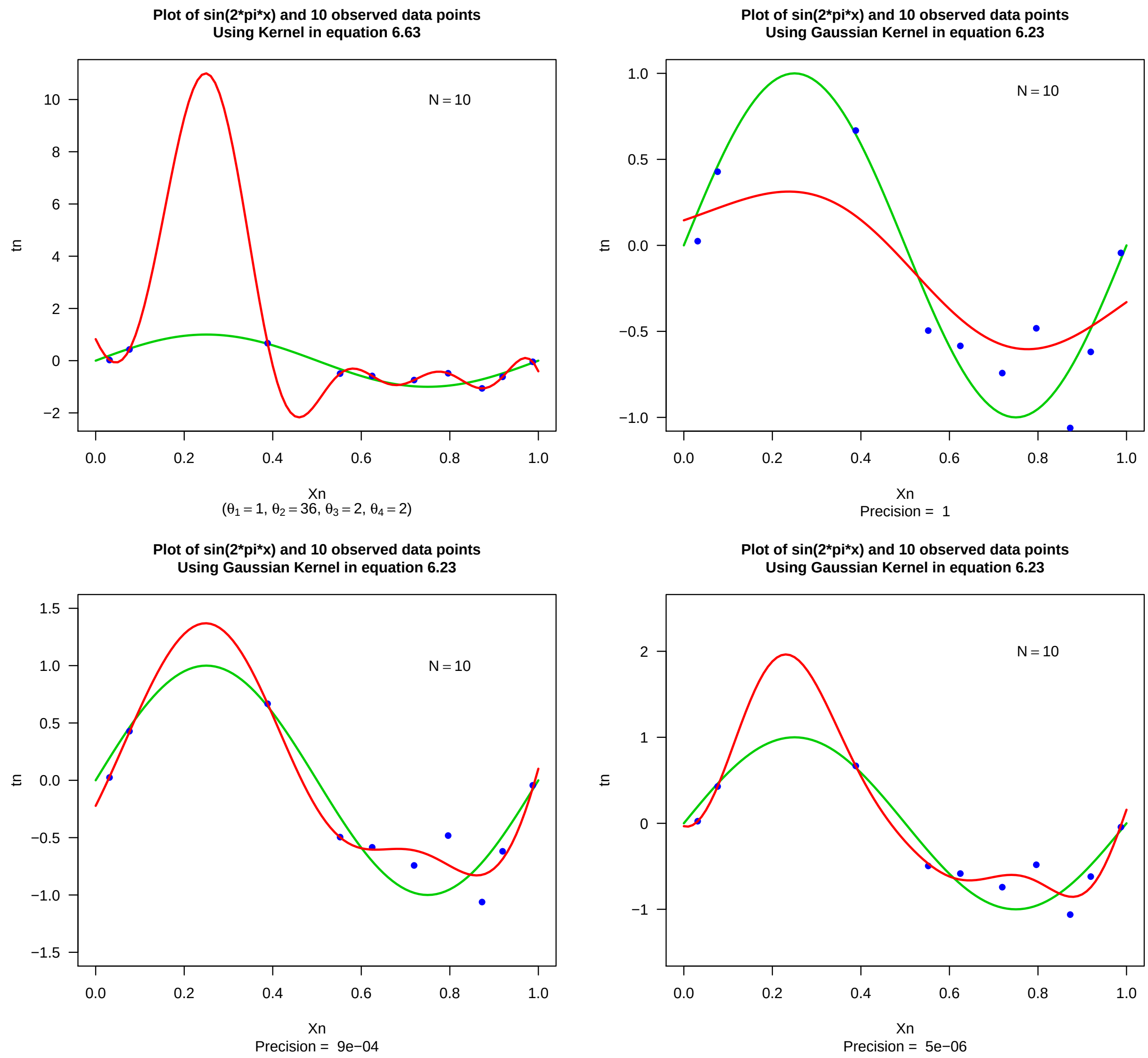



```r
# Gaussian Kernel (GK)
GK <- function(Xn, Xm){
  kern = exp(-0.5*((as.matrix(Xn-Xm))%*%t(as.matrix(Xn-Xm))))
  return(kern)
}


GPR<- function(train,test, teta = c(1,36,0,0), k = 1, Precision = 0.0005){

  library(matrixcalc)

  # extracting freatures from training data
  train = data.frame(train)
  N = dim(train)[1]
  target.train = train[,2]
  # train  = train[,1] # training data

  colnames(train)  <- c("x", "y") # column label

  # test data
  test = data.frame(test)
  N2 = dim(test)[1]
  target.test = test[,2]
  # test  = test[,1] # test data

  names(test)  <- c("x" , "y") # column label
```

```
# putting train and test data together
X =  rbind(train, test)
X = data.frame(X)
N_N2 = dim(X)[1]
X = data.frame(X[,1])

if(k == 1){
# extraction thetas from the vector
teta = as.vector(teta)
theta_0 = teta[1]
theta_1 = teta[2]
theta_2 = teta[3]
theta_3 = teta[4]

# container for the gram matrix
GM <- matrix(NA, ncol = N_N2, nrow = N_N2)

# a for loop for creating the  Gram Matrix
for (i in 1:N_N2){
    for(j in 1:N_N2){
      # Gram Matrix for Kernel in Equation 6.63
      GM[i,j] = (theta_0)*exp((-(theta_1/2)*((norm(as.matrix(X[i,]-X[j,]),
                                                   type = "F"))^2)))
      + (theta_2) + (theta_3)*(as.matrix(X[i,])%*%t(as.matrix(X[j,])))
    }
}

# Gram matrix based on the train data
C.N = GM[1:N, 1:N]

# matrix k associated with the test data
k_matrix = GM[1:N, (N+1): N_N2]

# matrix c associated with the test data
c_matrix = GM[(N+1): N_N2, (N+1): N_N2]


 # mean of the predictive distribution
 MU = t(k_matrix)%*%solve(C.N)%*%target.train

 # variance of the predictive distribution
 SIGMA2 = c_matrix - (t(k_matrix)%*%solve(C.N)%*%k_matrix)


 } else if (k != 1  | Precision != 0){


  # container for the gram matrix for both train and test data
K.gram.Gaussian = matrix(NA, nrow = N_N2, ncol = N_N2)
SIG = 0.05
for (i in 1:N_N2) {
  for (j in 1: N_N2) {
    ## Gaussian Kernel
```

```r
      K.gram.Gaussian[i,j] = exp(-(1/(2*SIG))*(norm(as.matrix(X[i,] - X[j,]),
                                                        type = "F"))^2)
    }

  }

  GM = K.gram.Gaussian + Precision*diag(N_N2)

  # Gram matrix based on the train data
  C.N = GM[1:N, 1:N]

  # matrix k associated with the test data
  k_matrix = GM[1:N, (N+1): N_N2]

  # matrix c associated with the test data
  c_matrix = GM[(N+1): N_N2, (N+1): N_N2]


   # mean of the predictive distribution
   MU = t(k_matrix)%*%solve(C.N)%*%target.train

   # variance of the predictive distribution
   SIGMA2 = c_matrix - (t(k_matrix)%*%solve(C.N)%*%k_matrix)


  }

   # checking if the Gram Matrix is positive definite
  PSD = is.positive.semi.definite(GM)
  PD = is.positive.definite(C.N)


  return(list(PSD = PSD, PD = PD, Precision = Precision
              , MU = MU, SIGMA2 = SIGMA2, theta = teta
              ))

}



par(mfrow = c(2,2))
res.GPR1 = GPR(datframe4, data_sinx, k= 1, teta = c(1,36,2,2))

plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
main = c(paste("Plot of sin(2*pi*x) and 10 observed data points"),
         paste("Using Kernel in equation 6.63"))
, cex.main = 1,
ylim = c(min(res.GPR1$MU), max(res.GPR1$MU)), sub =expression( paste( "(" ,
                                                                  theta[1] == 1,
                                                                  ", ", theta[2]== 36,
              ", ",theta[3] == 2, ", ", theta[4]== 2 ,")")))
text(.8,10, expression( N == 10))
points(t_target4~X_training4, data = datframe4, col = 4, pch = 16)
```

```r
lines(Xn , res.GPR1$MU, lwd = 2, col  = 2)

res.GPR2 = GPR(datframe4, data_sinx, k= 3, Precision = 1)
plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
main = c(paste("Plot of sin(2*pi*x) and 10 observed data points"),
         paste("Using Gaussian Kernel in equation 6.23")), cex.main = 1,
ylim = c(min(tn), max(tn)),sub = paste("Precision = ", res.GPR2$Precision))
text(.8,.9, expression( N == 10))
points(t_target4~X_training4, data = datframe4, col = 4, pch = 16)
lines(Xn , res.GPR2$MU, lwd = 2, col  = 2)



res.GPR3 = GPR(datframe4, data_sinx, k= 3, Precision = 0.0009)
plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
main = c(paste("Plot of sin(2*pi*x) and 10 observed data points"),
         paste("Using Gaussian Kernel in equation 6.23")), cex.main = 1,
ylim = c(-1.5,1.5),sub = paste("Precision = ", res.GPR3$Precision))
text(.8,1.0, expression( N == 10))
points(t_target4~X_training4, data = datframe4, col = 4, pch = 16)
lines(Xn , res.GPR3$MU, lwd = 2, col  = 2)


res.GPR3 = GPR(datframe4, data_sinx, k= 3, Precision = 0.000005)
plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
main = c(paste("Plot of sin(2*pi*x) and 10 observed data points"),
         paste("Using Gaussian Kernel in equation 6.23")), cex.main = 1,
ylim = c(-1.5,2.5),sub = paste("Precision = ", res.GPR3$Precision))
text(.8,2, expression( N == 10))
points(t_target4~X_training4, data = datframe4, col = 4, pch = 16)
lines(Xn , res.GPR3$MU, lwd = 2, col  = 2)
```




```
par(mfrow = c(1,1))
```