STAT-721 H/W (Reproduction of Figure 3.8 & 3.9)

Segbehoe, Lawrence Sethor

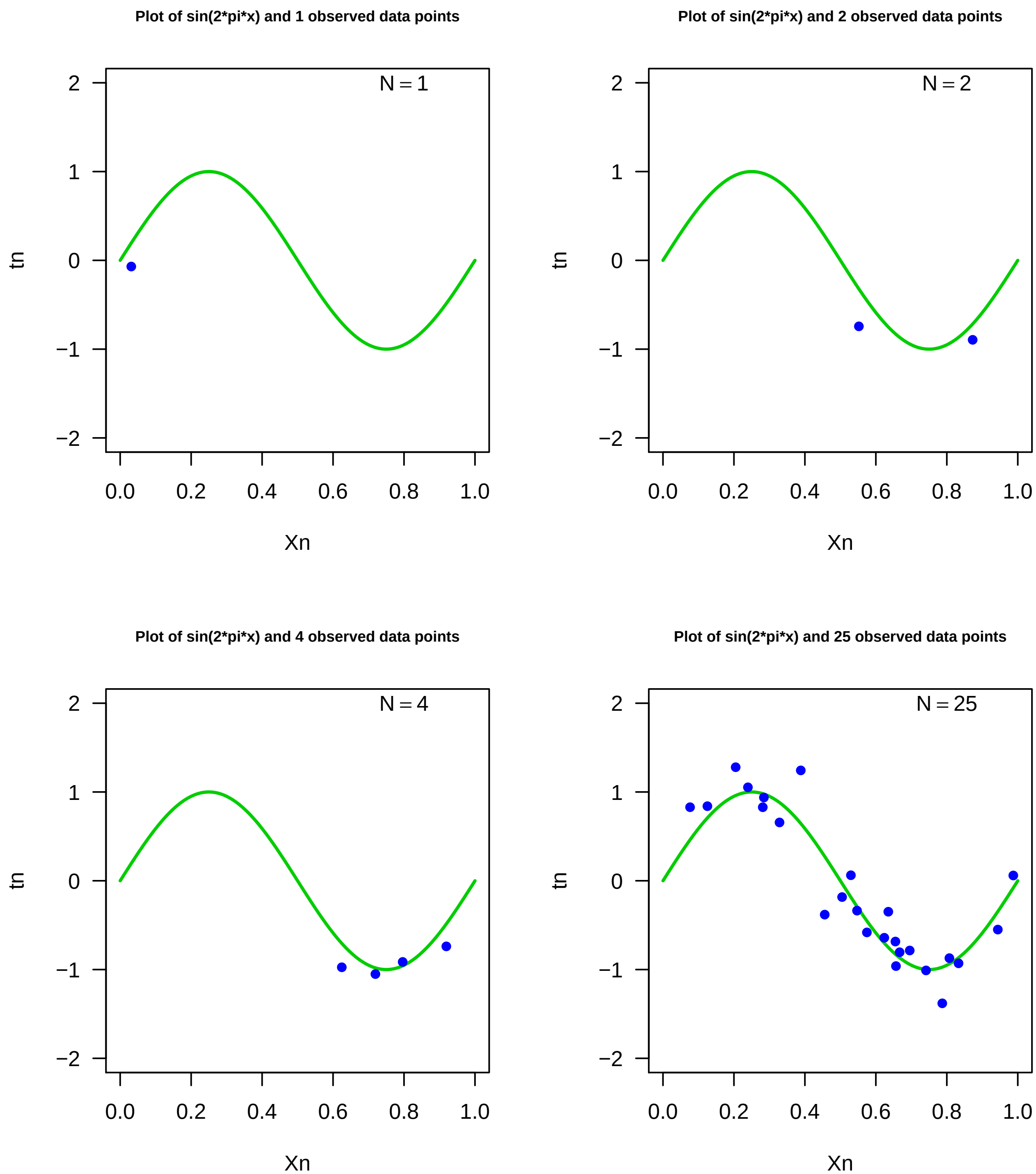
September 9, 2018

Creating the synthetic sinusoidal data set

```
library(MASS)
## 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, 0.001)
## 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)

## Creating the training data set from the range[0,1]
# X_training1 <- 0.4
# X_training2 <- c(0.4, 0.6)
# X_training3 <- c(0,0.4,0.6,1)
# X_training4 <- seq(0,1,0.04)[-1]
X_training1 <- runif(1,min = 0, max = 1)
X_training2 <- runif(2,min = 0, max = 1)
X_training3 <- runif(4,min = 0, max = 1)
X_training4 <- runif(25,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_target1 = f(X_training1) + rnorm(1, 0, sigma_squared)
t_target2 = f(X_training2) + rnorm(2, 0, sigma_squared)
t_target3 = f(X_training3) + rnorm(4, 0, sigma_squared)
t_target4 = f(X_training4) + rnorm(25, 0, sigma_squared)

## making a dataframe form the observed (training and target) dataset
datframe1 = data.frame(X_training1, t_target1)
datframe2 = data.frame(X_training2, t_target2)
datframe3 = data.frame(X_training3, t_target3)
datframe4 = data.frame(X_training4, t_target4)
```

Plotting the synthetic sinusoidal data set

```
## demacation of the plotting area
layout(matrix(1:4, ncol = 2, byrow = T ))
## Plot with of the polynomial with order = 0
plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
     main = "Plot of sin(2*pi*x) and 1 observed data points",
     cex.main = 0.7, ylim = c(-2,2))
text(.8,2, expression( N == 1))
points(t_target1~X_training1, data = datframe1, col = 4, pch = 16)


## Plot with of the polynomial with order = 1
plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
     main = "Plot of sin(2*pi*x) and 2 observed data points",
     cex.main = 0.7, ylim = c(-2,2))
text(.8,2, expression( N == 2))
points(t_target2~X_training2, data = datframe2, col = 4, pch = 16)


## Plot with of the polynomial with order = 3
plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
     main = "Plot of sin(2*pi*x) and 4 observed data points",
     cex.main = 0.7, ylim = c(-2,2))
text(.8,2, expression( N == 4))
points(t_target3~X_training3, data = datframe3, col = 4, pch = 16)


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 = 0.7,ylim = c(-2,2))
text(.8,2, expression( N == 25))
points(t_target4~X_training4, data = datframe4, col = 4, pch = 16)
```




```
layout(matrix(1:1, ncol = 2, byrow = T ))
```

Gaussian Basis

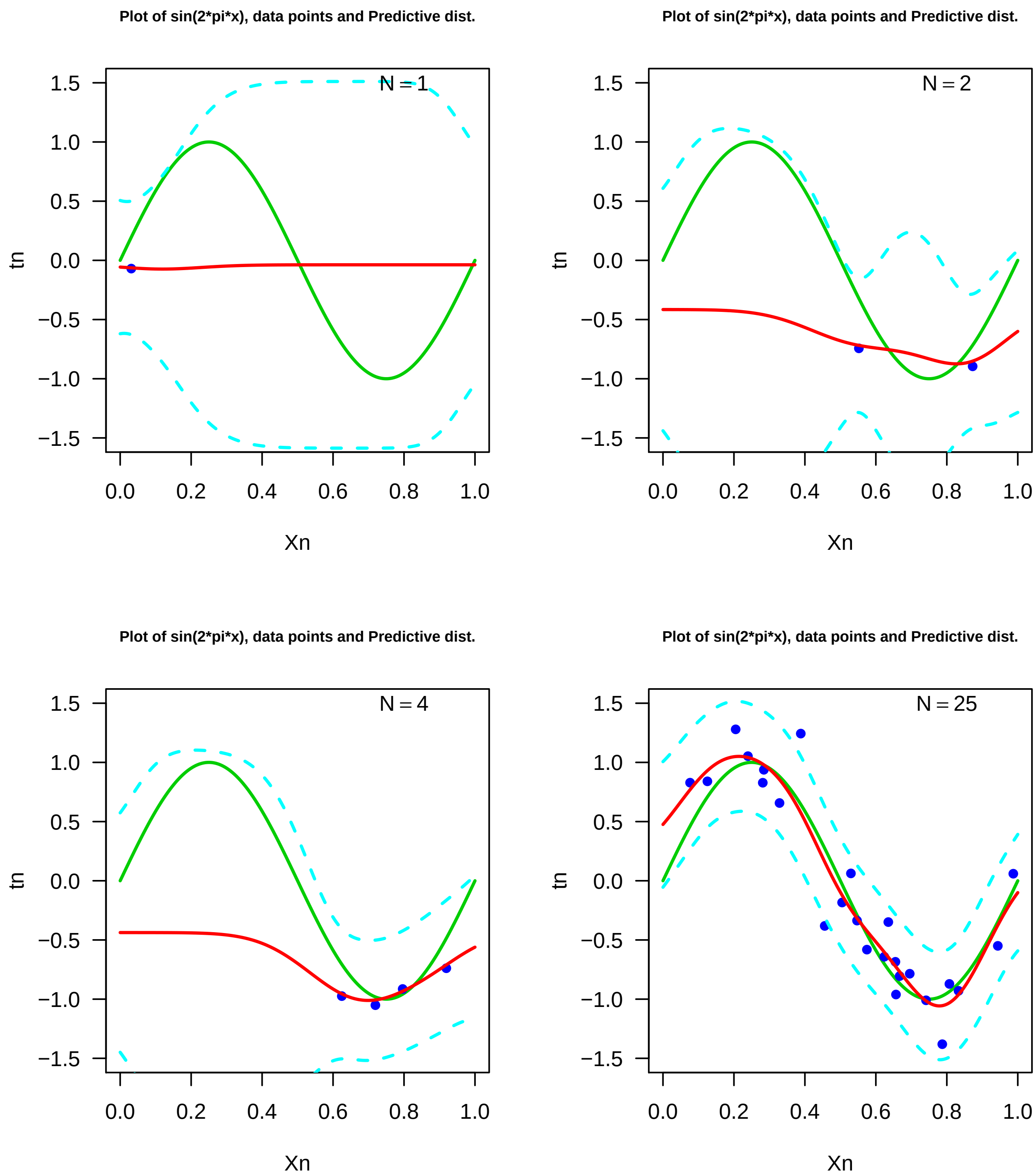
$$\phi_j(x) = \exp\left\{\frac{x-\mu_i}{2s^2}\right\}^2$$

When the training data set is generated using uniform distribution

Figure 3.8

```
## function for the Gaussian Basis
## changing the sigma2 in the basis function has an effect on the graph
G <- function(x, mu, sigma2 = 0.01  ){
  exp(-((x - mu)^2)/(2*sigma2))
}


## function for the posterior over w

Norm.pred <- function(Xn, x , t ){
## Creating the design matrix of the input variable

hyperparameter.precision  = 3 ## these precision has effect on the variance of the curves
beta = hyperparameter.precision + 3
Alpha = hyperparameter.precision -2


## Creating the design matrix from the basis function from train data set
Bn <- 9 # number of basis functions
PHI <- matrix(NA, ncol = Bn, nrow = length(x), byrow = F)
k = 1
 for(i in seq(0.1,0.9,0.1)){
  PHI[,k] <- G(x = x, mu = i)
 k = k +1}
PHI <- as.matrix(cbind(rep(1,length(x)), PHI))

## creating identity matrix
I <- diag(dim(t(PHI)%*%PHI)[1])

## Creating the mean and variance of the posterior over w given t
SN <- solve(Alpha*I + beta*(t(PHI)%*%PHI))
mN <- beta*SN%*%(t(PHI)%*%t)


## Creating the design matrix from the basis function for prediction
Bn <- 9 # number of basis functions
PHI.pred <- matrix(NA, ncol = Bn, nrow = length(Xn), byrow = F)
k = 1
 for(i in seq(0.1,0.9,0.1)){
  PHI.pred[,k] <- G(x = Xn, mu = i)
 k = k +1}
PHI.pred <- as.matrix(cbind(rep(1,length(Xn)), PHI.pred))
```

```
## Creating the mean and variance of the predictive distribution of t_new given X_new,
mNp <- c()
for(i in 1:length(Xn)){
mNp[i] <- t(mN)%*%PHI.pred[i,]}

SNp <- c()
for(i in 1:length(Xn)){
SNp[i] <- (1/beta) + t(PHI.pred[i,])%*%SN%*%PHI.pred[i,]}

return(list(mN = mN, SN = SN, mNp = mNp, SNp = SNp))
}


################################################################3
plot3.8 <- function(Xn, train, target){

  ## getting the parameters from norm function
p = Norm.pred(Xn, train, target)

d <- cbind(Xn,Xn,Xn,(p$mNp + sqrt(p$SNp)), p$mNp, (p$mNp - sqrt(p$SNp)))

plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
     main = "Plot of sin(2*pi*x), data points and Predictive dist.",
     cex.main = 0.7, ylim = c(-1.5,1.5))
points(target ~ train, col = 4, pch = 16)
lines(d[,1], d[,4], col = 13, type = "l", lty = 2, lwd = 2)
lines(d[,2], d[,5], col = "red", type = "l", lwd = 2)
lines(d[,3], d[,6], col = 13, type = "l", lty = 2, lwd = 2)
}

par(mfrow = c(2,2))
plot3.8(Xn, X_training1, t_target1);text(.8,1.5, expression( N == 1))
plot3.8(Xn, X_training2, t_target2);text(.8,1.5, expression( N == 2))
plot3.8(Xn, X_training3, t_target3);text(.8,1.5, expression( N == 4))
plot3.8(Xn, X_training4, t_target4);text(.8,1.5, expression( N == 25))
```




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

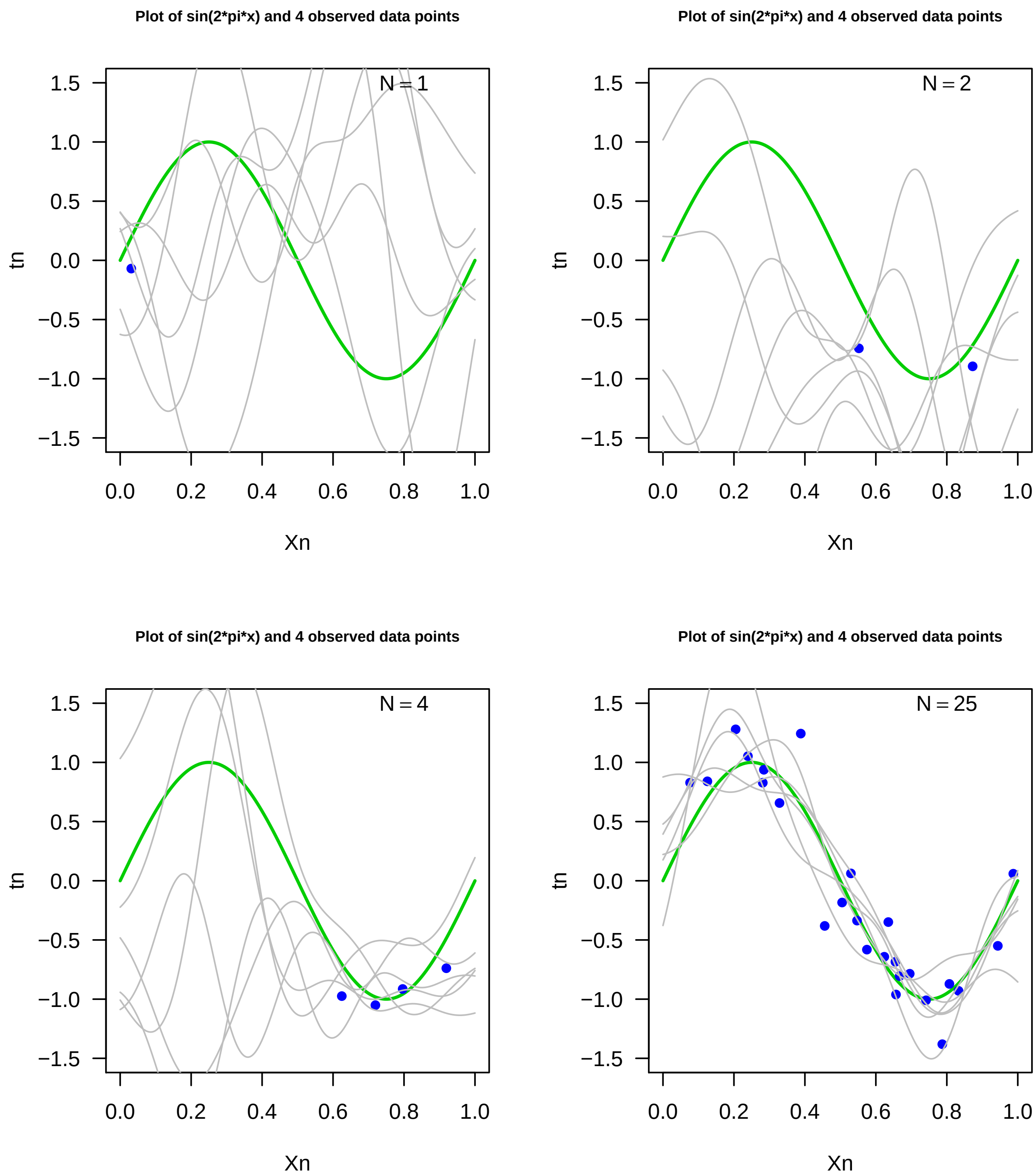
Comment 3.8

It can be noticed that the variability in the model is influence by the training data set.

Figure 3.9

```
## function for the Gaussian Basis
## changing the sigma2 in the basis function has an effect on the graph
G <- function(x, mu, sigma2 = 0.01  ){
  exp(-((x - mu)^2)/(2*sigma2))
}


## function for the posterior over w

Norm <- function(x , t ){
## Creating the design matrix of the input variable

hyperparameter.precision  = 3 ## these precision has effect on the variance of the curves
beta = hyperparameter.precision + 3
Alpha = hyperparameter.precision -2


## Creating the design matrix from the basis function

Bn <- 9 # number of basis functions
PHI <- matrix(NA, ncol = Bn, nrow = length(x), byrow = F)
k = 1
 for(i in seq(0.1,0.9,0.1)){
  PHI[,k] <- G(x = x, mu = i)
 k = k +1}
PHI <- as.matrix(cbind(rep(1,length(x)), PHI))

## creating identity matrix
I <- diag(dim(t(PHI)%*%PHI)[1])

## Creating the mean and variance of the posterior over w given t
SN <- solve(Alpha*I + beta*(t(PHI)%*%PHI))
mN <- beta*SN%*%(t(PHI)%*%t)

## Creating the mean and variance of the predictive distribution of t_new given X_new,
Xnew <- sample(x, 1)
PHIp <- matrix(NA, ncol = 1 , nrow = Bn, byrow = F)
k = 1
for(i in seq(0.1,0.9,0.1)){
  PHIp[k,] <- G(x = Xnew, mu = i)
 k = k +1}
PHIp <- rbind(1, PHIp)
mNp <- t(mN)%*%PHIp
SNp <- (1/beta) + t(PHIp)%*%SN%*%PHIp
return(list(mN = mN, SN = SN, mNp = mNp, SNp = SNp))
}


################################################################
```

```r
plot3.9 <- function(Xn, train, target){

simN = 6 # number of samples

par = Norm(train, target) # getting the parameters of the posterior


# Sampling from multivariate dist of the posterior
# Creating a container for 6 samples
s <- matrix(NA, nrow = dim(par$mN)[1], ncol = simN, byrow = F)
k= 1
for(i in 1:simN){
  s[,k]  = mvrnorm(n = 1, mu = par$mN, Sigma = par$SN)
  k = k +1}

## Creating a new design matrix from the basis function for plotting
Bn <- 9 # number of basis functions
PHI_plot <- matrix(NA, ncol = Bn, nrow = length(Xn) , byrow = T)
k = 1
 for(i in seq(0.1,0.9,0.1)){
  PHI_plot[,k] <- G(x = Xn, mu = i)
 k = k +1}
PHI_plot <- as.matrix(cbind(rep(1,length(Xn)), PHI_plot))

plot(tn~Xn, data = data_sinx, col = 3, type = "l", las = 1, lwd = 2,
     main = "Plot of sin(2*pi*x) and 4 observed data points",
     cex.main = 0.7, ylim = c(-1.5,1.5))
points(target~train,col = 4, pch = 16)
for(i in 1:simN){
  lines(Xn, PHI_plot%*%s[,i], col = "grey", type = "l")
}}


par(mfrow = c(2,2))
plot3.9(Xn, X_training1, t_target1);text(.8,1.5, expression( N == 1))
plot3.9(Xn, X_training2, t_target2);text(.8,1.5, expression( N == 2))
plot3.9(Xn, X_training3, t_target3);text(.8,1.5, expression( N == 4))
plot3.9(Xn, X_training4, t_target4);text(.8,1.5, expression( N == 25))
```




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

Comment 3.9

It can be noticed that the variability in the model is influence by the training data set.

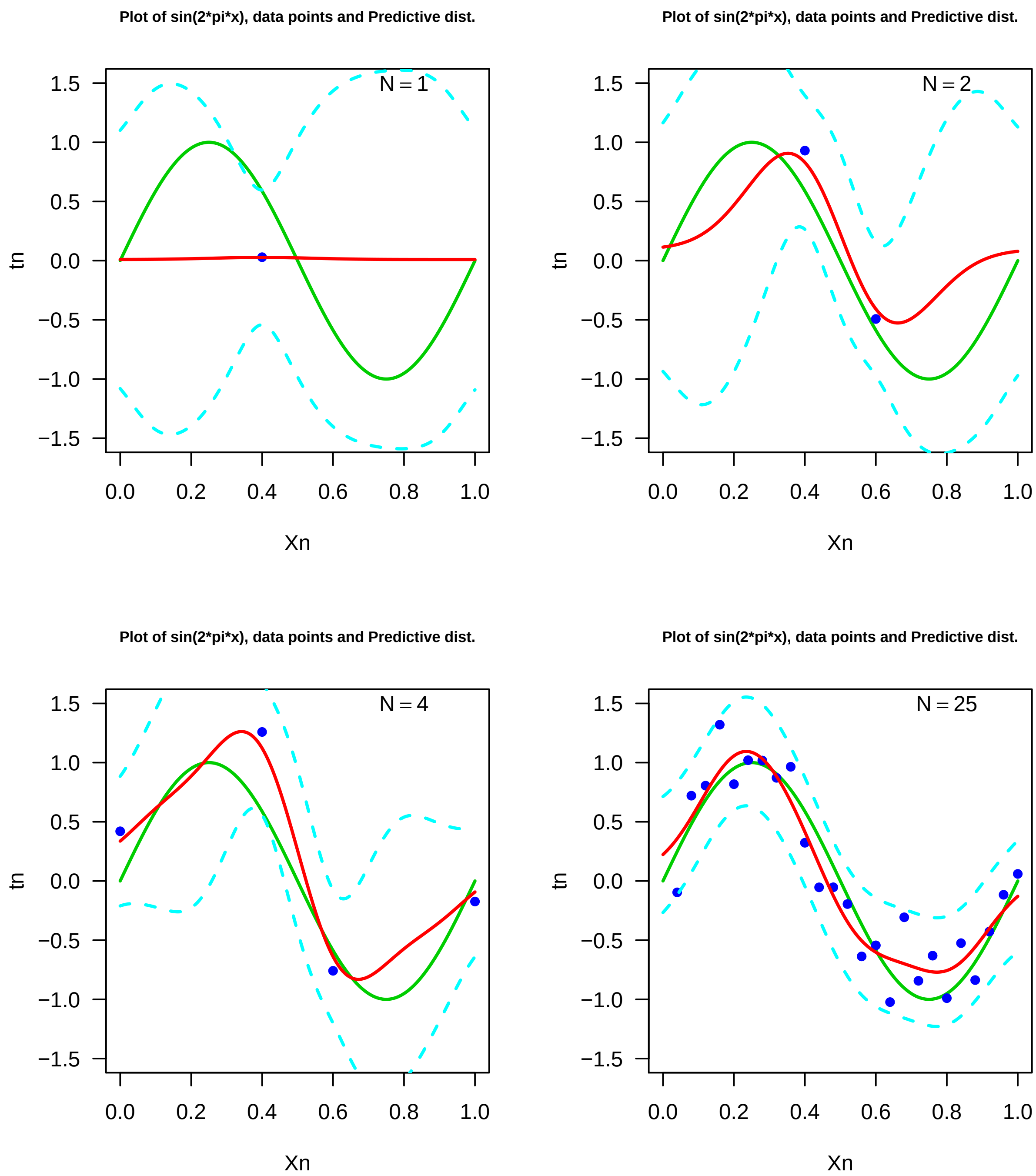
When data train data is positioned to portray what the text book has

Creating the synthetic sinusoidal data set

```r
library(MASS)
## 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, 0.001)
## 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)

## Creating the training data set from the range[0,1]
X_training1 <- 0.4
X_training2 <- c(0.4, 0.6)
X_training3 <- c(0,0.4,0.6,1)
X_training4 <- seq(0,1,0.04)[-1]
# X_training1 <- runif(1,min = 0, max = 1)
# X_training2 <- runif(2,min = 0, max = 1)
# X_training3 <- runif(4,min = 0, max = 1)
# X_training4 <- runif(25,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_target1 = f(X_training1) + rnorm(1, 0, sigma_squared)
t_target2 = f(X_training2) + rnorm(2, 0, sigma_squared)
t_target3 = f(X_training3) + rnorm(4, 0, sigma_squared)
t_target4 = f(X_training4) + rnorm(25, 0, sigma_squared)

## making a dataframe form the observed (training and target) dataset
datframe1 = data.frame(X_training1, t_target1)
datframe2 = data.frame(X_training2, t_target2)
datframe3 = data.frame(X_training3, t_target3)
datframe4 = data.frame(X_training4, t_target4)
```

```r
par(mfrow = c(2,2))
plot3.8(Xn, X_training1, t_target1);text(.8,1.5, expression( N == 1))
plot3.8(Xn, X_training2, t_target2);text(.8,1.5, expression( N == 2))
plot3.8(Xn, X_training3, t_target3);text(.8,1.5, expression( N == 4))
```

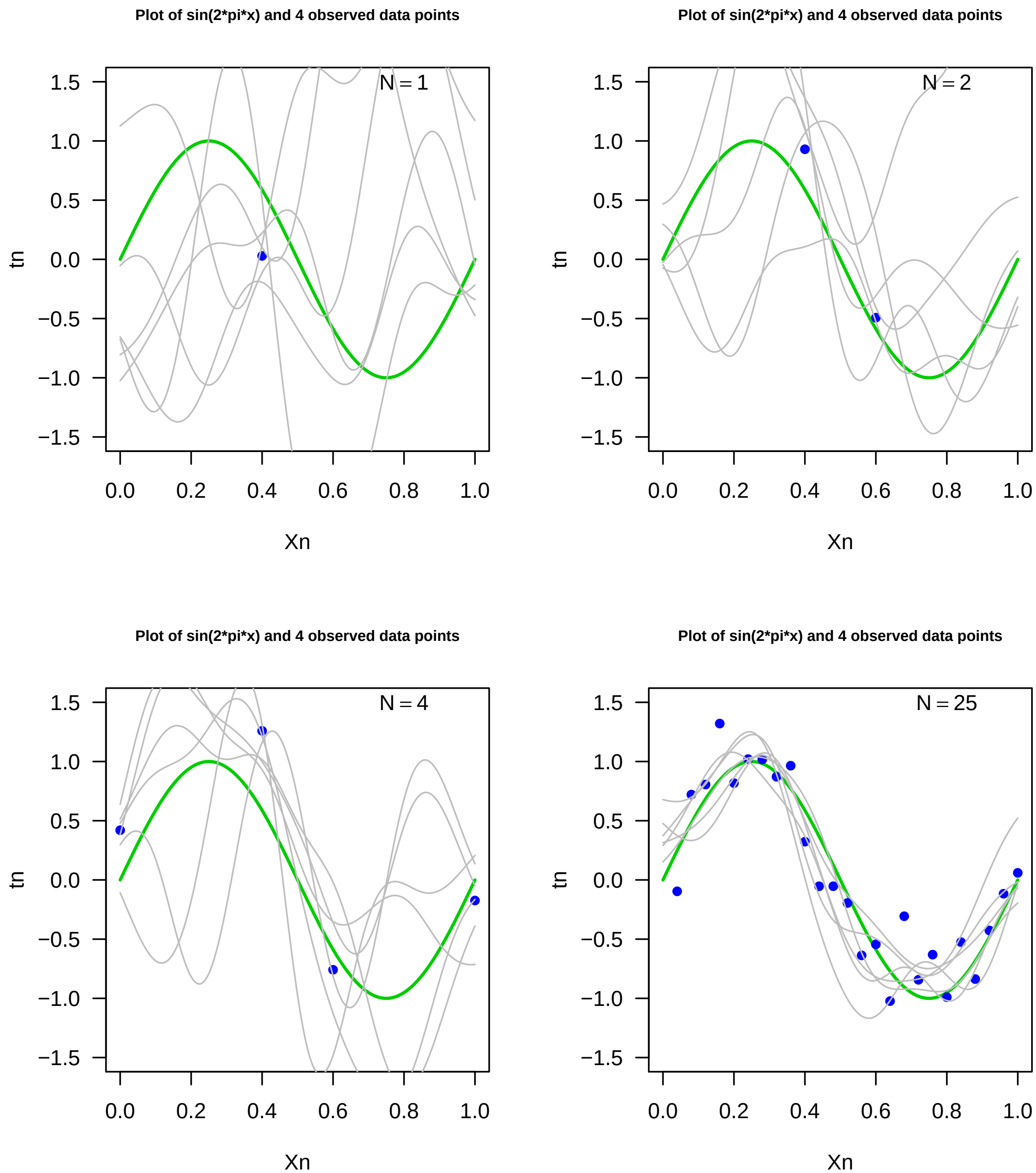
```r
plot3.8(Xn, X_training4, t_target4);text(.8,1.5, expression( N == 25))
```




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

```r
par(mfrow = c(2,2))
plot3.9(Xn, X_training1, t_target1);text(.8,1.5, expression( N == 1))
plot3.9(Xn, X_training2, t_target2);text(.8,1.5, expression( N == 2))
```

```r
plot3.9(Xn, X_training3, t_target3);text(.8,1.5, expression( N == 4))
plot3.9(Xn, X_training4, t_target4);text(.8,1.5, expression( N == 25))
```




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