Lab4

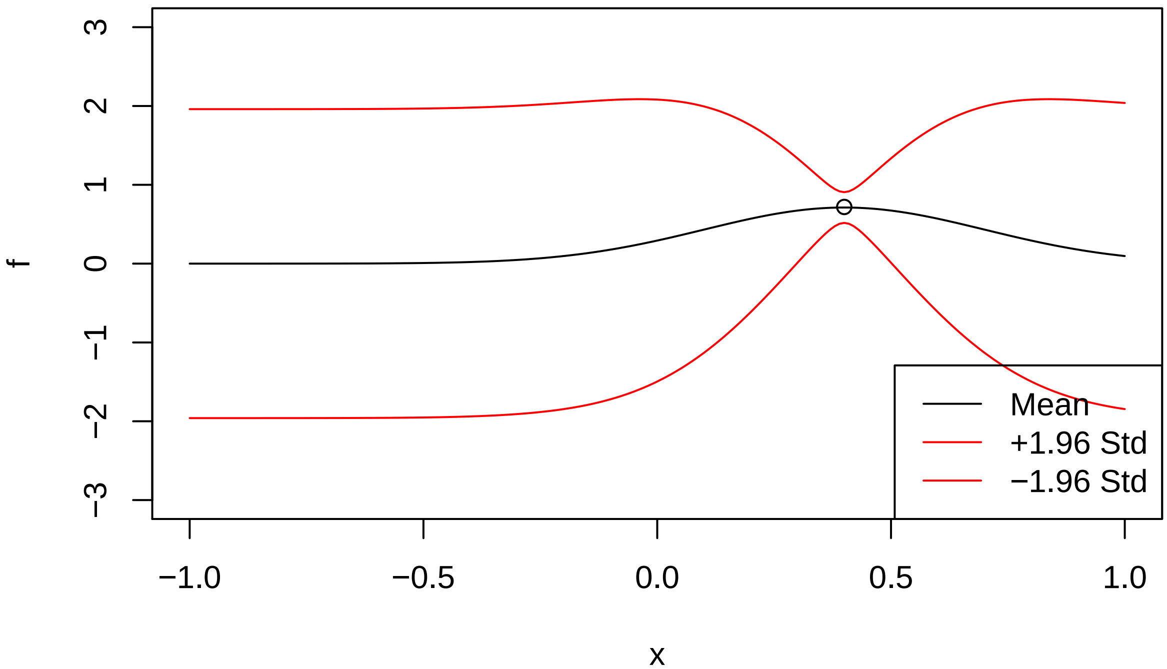
Yifan Ding

Q2.1

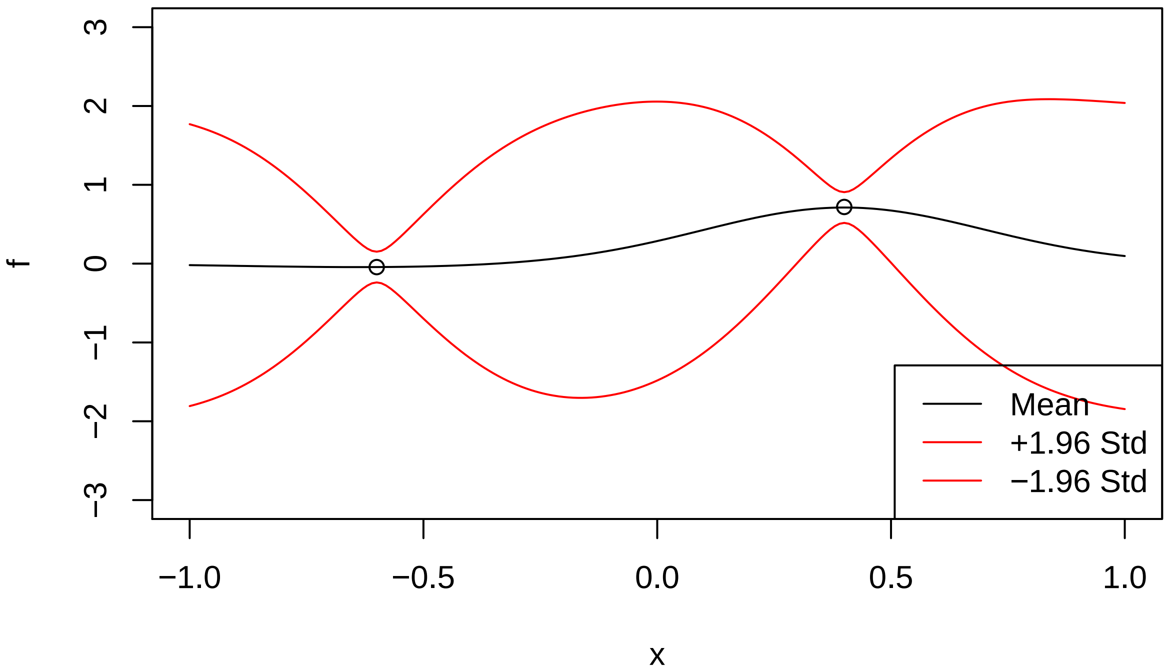
```r
SquaredExpKernel <- function(x1,x2,sigmaF=1,l=3){
  n1 <- length(x1)
  n2 <- length(x2)
  K <- matrix(NA,n1,n2)
  for (i in 1:n2){
    K[,i] <- sigmaF^2*exp(-0.5*( (x1-x2[i])/l)^2 )
  }
  return(K)
}

posteriorGP <- function(X, y, Xstar, sigmaNoise, K, ...){
  n <- length(X)
  C <- K(X, X, ...) + sigmaNoise ^ 2 * diag(n)
  L <- t(chol(C))
  L_inv <- solve(L)
  C_inv <- t(L_inv) %*% L_inv
  mean <- K(Xstar, X, ...) %*% C_inv %*% y
  var <- K(Xstar, Xstar, ...) - K(Xstar, X, ...) %*% C_inv %*% t(K(Xstar, X, ...))
  return(list(mean=mean, var=var))

}
```

Q2.2

```r
X = 0.4
y = 0.719
Xstar = seq(-1, 1, 0.01)
sigmaNoise = 0.1
K = SquaredExpKernel
f = posteriorGP(X, y, Xstar, sigmaNoise, K, sigmaF=1, l=0.3)

up_bound <- f$mean+1.96*sqrt(diag(f$var))
low_bound <- f$mean-1.96*sqrt(diag(f$var))
plot(Xstar, f$mean, type='l', ylim =c(-3,3), xlab = "x", ylab = "f")
lines(Xstar, up_bound, col='red')
lines(Xstar, low_bound, col='red')
points(X, y)
legend("bottomright", legend=c("Mean","+1.96 Std", "-1.96 Std"),
        lwd=c(1,1,1), col=c("black","red","red"))
```

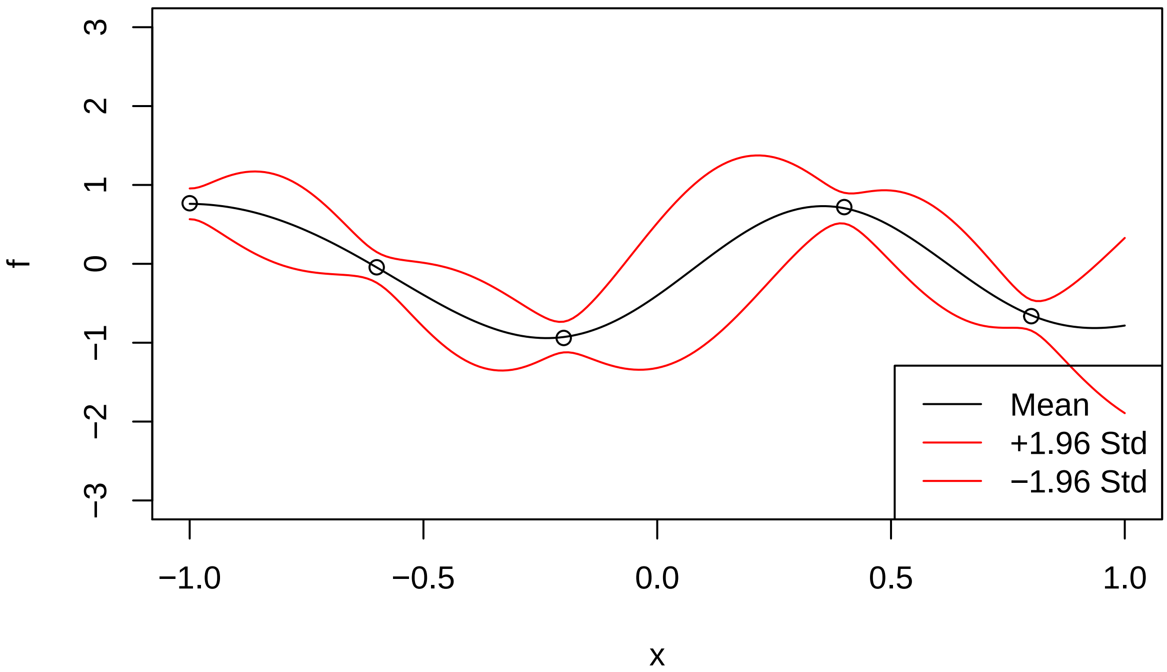



```
X = c(0.4, -0.6)
y = c(0.719, -0.044)
Xstar = seq(-1, 1, 0.01)
sigmaNoise = 0.1
K = SquaredExpKernel
f = posteriorGP(X, y, Xstar, sigmaNoise, K, sigmaF=1, l=0.3)

up_bound <- f$mean+1.96*sqrt(diag(f$var))
low_bound <- f$mean-1.96*sqrt(diag(f$var))
plot(Xstar, f$mean, type='l', ylim =c(-3,3), xlab = "x", ylab = "f")
lines(Xstar, up_bound, col='red')
lines(Xstar, low_bound, col='red')
points(X, y)
legend("bottomright", legend=c("Mean","+1.96 Std", "-1.96 Std"),
        lwd=c(1,1,1), col=c("black","red","red"))
```



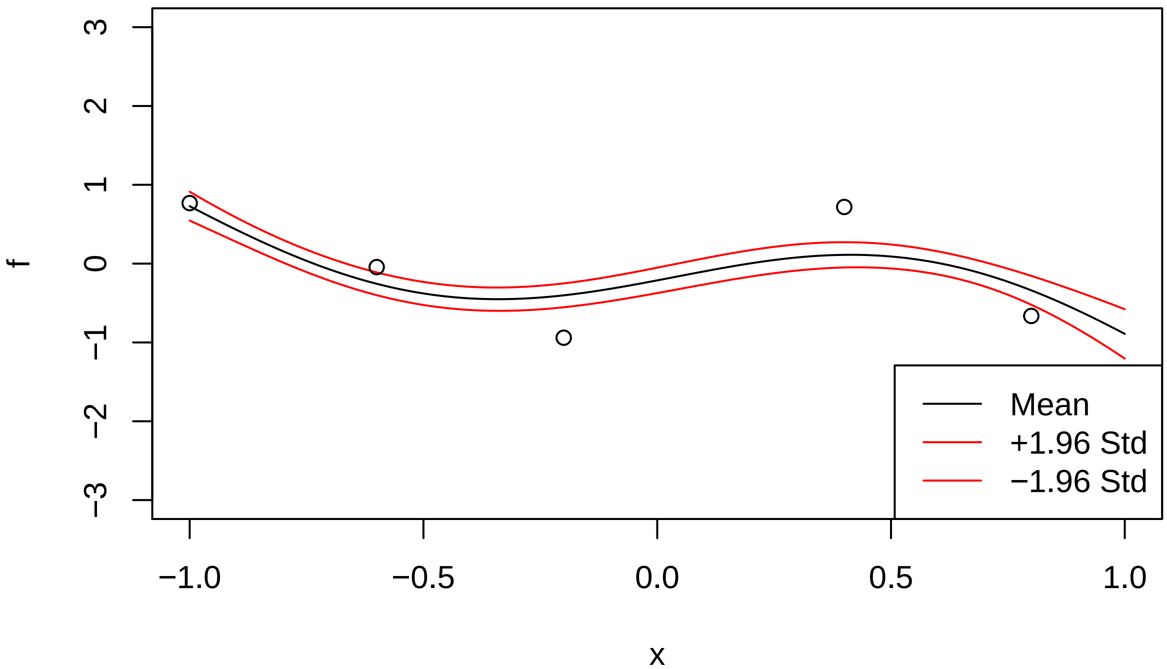
Q2.4

```
X = c(0.4, -0.6, -1, -0.2, 0.8)
y = c(0.719, -0.044, 0.768, -0.940, -0.664)
Xstar = seq(-1, 1, 0.01)
sigmaNoise = 0.1
K = SquaredExpKernel
f = posteriorGP(X, y, Xstar, sigmaNoise, K, sigmaF=1, l=0.3)

up_bound <- f$mean+1.96*sqrt(diag(f$var))
low_bound <- f$mean-1.96*sqrt(diag(f$var))
plot(Xstar, f$mean, type='l', ylim =c(-3,3), xlab = "x", ylab = "f")
lines(Xstar, up_bound, col='red')
lines(Xstar, low_bound, col='red')
points(X, y)
legend("bottomright", legend=c("Mean","+1.96 Std", "-1.96 Std"),
        lwd=c(1,1,1), col=c("black","red","red"))
```



Q2.5

```
X = c(0.4, -0.6, -1, -0.2, 0.8)
y = c(0.719, -0.044, 0.768, -0.940, -0.664)
Xstar = seq(-1, 1, 0.01)
sigmaNoise = 0.1
K = SquaredExpKernel
f = posteriorGP(X, y, Xstar, sigmaNoise, K, sigmaF=1, l=1)

up_bound <- f$mean+1.96*sqrt(diag(f$var))
low_bound <- f$mean-1.96*sqrt(diag(f$var))
plot(Xstar, f$mean, type='l', ylim =c(-3,3), xlab = "x", ylab = "f")
lines(Xstar, up_bound, col='red')
lines(Xstar, low_bound, col='red')
points(X, y)
legend("bottomright", legend=c("Mean","+1.96 Std", "-1.96 Std"),
        lwd=c(1,1,1), col=c("black","red","red"))
```



Q3.1

```r
df <- read.csv("TempTullinge.csv", sep=";", header=TRUE)
time <- seq(1, 2190)
time <- (time -mean(time)) / sd(time)
day <- rep(1:365,6)
day <- (day -mean(day)) / sd(day)
df$time <- time

RBFkernel <- function(sigmaf = 1, ell = 1) {
  rval <- function(x, y = NULL) {
      res <- sigmaf^2*exp(-0.5*( (x-y)/ell)^2 )
      return(res)
    }
  class(rval) <- "kernel"
  return(rval)
}

SEkernel <- RBFkernel(sigmaf = 1, ell = 1)
SEkernel(1,2)
```

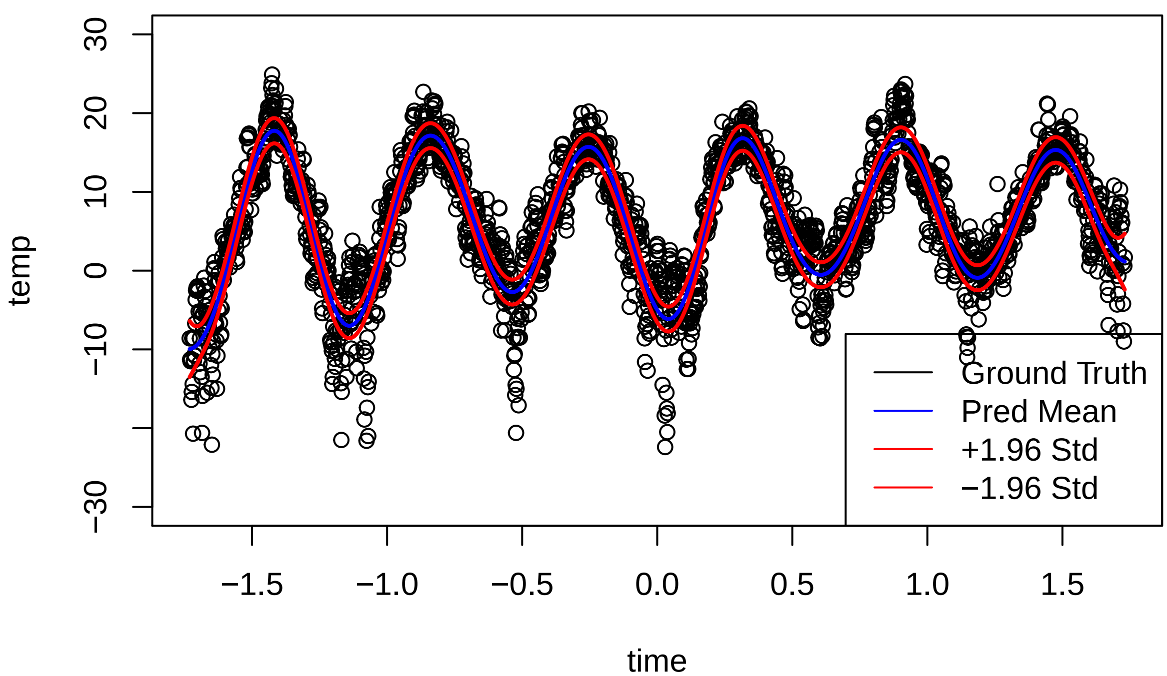
```
## [1] 0.6065307
```

```r
X <- matrix(c(1,3,4), 3, 1)
Xstar <- matrix(c(2,3,4), 3, 1)
kernelMatrix(kernel = SEkernel, x = X, y = Xstar)
```

```
## An object of class "kernelMatrix"
##           [,1]      [,2]      [,3]
## [1,] 0.6065307 0.1353353 0.0111090
## [2,] 0.6065307 1.0000000 0.6065307
## [3,] 0.1353353 0.6065307 1.0000000
```

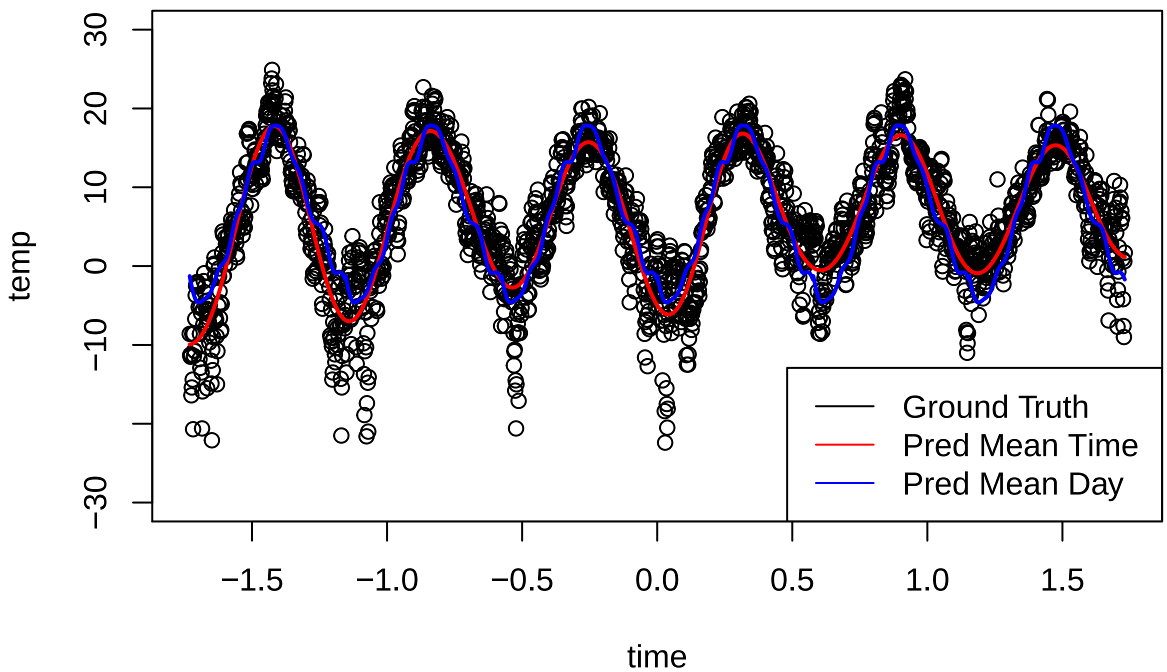
Q3.2

```
temp <- df$temp
plot(time, temp, ylim=c(-30, 30))

polyFit <- lm(temp ~  time + I(time^2))
sigmaNoise <- sd(polyFit$residuals)
sigmaf <- 20
ell <- 0.2
RBF <- RBFkernel(sigmaf = sigmaf, ell=ell)
GPfit <- gausspr(time, temp, kernel = RBF, var = sigmaNoise^2)
meanPred <- predict(GPfit, time)
lines(time, meanPred, lwd = 2, col='blue')

x<-time
xs<-time # XStar.
n <- length(x)
Kss <- kernelMatrix(kernel = RBF, x = xs, y = xs)
Kxx <- kernelMatrix(kernel = RBF, x = x, y = x)
Kxs <- kernelMatrix(kernel = RBF, x = x, y = xs)
Covf = Kss-t(Kxs)%*%solve(Kxx + sigmaNoise^2*diag(n), Kxs) # Covariance matrix of fStar.
lines(xs, meanPred - 1.96*sqrt(diag(Covf)), col = "red", lwd = 2)
lines(xs, meanPred + 1.96*sqrt(diag(Covf)), col = "red", lwd = 2)
legend("bottomright", legend=c("Ground Truth","Pred Mean ","+1.96 Std", "-1.96 Std"),
        lwd=c(1,1,1,1), col=c("black","blue","red","red"))
```




```
temp <- df$temp
plot(time, temp, ylim=c(-30, 30))

polyFit <- lm(temp ~  time + I(time^2))
sigmaNoise <- sd(polyFit$residuals)
sigmaf <- 20
ell <- 0.2
```

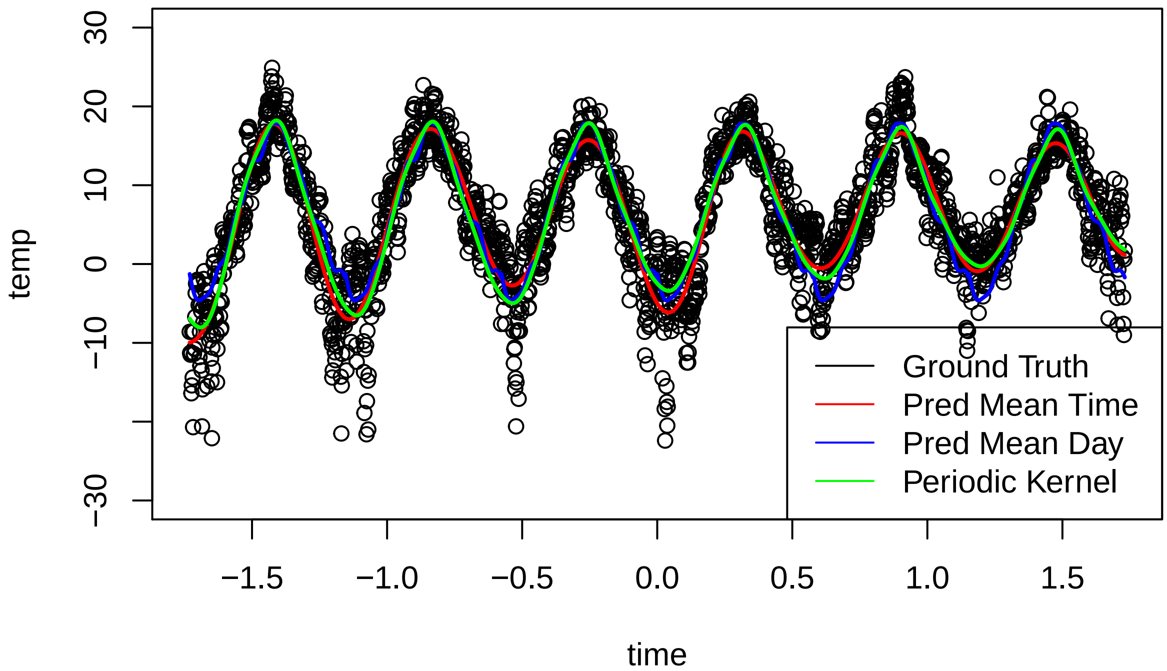
```r
RBF <- RBFkernel(sigmaf = sigmaf, ell=ell)
GPfit_day <- gausspr(day, temp, kernel = RBF, var = sigmaNoise^2)
meanPred_day <- predict(GPfit_day, day)
plot(time, temp, ylim=c(-30, 30))
lines(time, meanPred, lwd = 2, col='red')
lines(time, meanPred_day, lwd = 2, col='blue')
legend("bottomright", legend=c("Ground Truth","Pred Mean Time ","Pred Mean Day"),
        lwd=c(1,1,1), col=c("black","red","blue"))
```



Q3.5

```r
RBFPeriodickernel <- function(sigmaf = 1, ell1 = 1, ell2 = 1, d = 1) {
  rval <- function(x, y = NULL) {
      res <- sigmaf^2*exp(-0.5*( (x-y)/ell2)^2 )*exp(-2*(sin(pi*abs(x-y)/d)/ell1)^2 )
      return(res)
    }
  class(rval) <- "kernel"
  return(rval)
}


sigmaf <- 20
ell1 <- 1
ell2 <- 10
d <- 365 / sd(c(1:2190))

RBF_P <- RBFPeriodickernel(sigmaf = sigmaf, ell1 = ell1, ell2 = ell2, d = d)
GPfit_PK <- gausspr(time, temp, kernel = RBF_P, var = sigmaNoise^2)
meanPred_PK <- predict(GPfit_PK, time)
plot(time, temp, ylim=c(-30, 30))
lines(time, meanPred, lwd = 2, col='red')
lines(time, meanPred_day, lwd = 2, col='blue')
lines(time, meanPred_PK, lwd = 2, col='green')
```

```r
legend("bottomright", legend=c("Ground Truth","Pred Mean Time ","Pred Mean Day", "Periodic Kernel"),
        lwd=c(1,1,1,1), col=c("black","red","blue","green"))
```



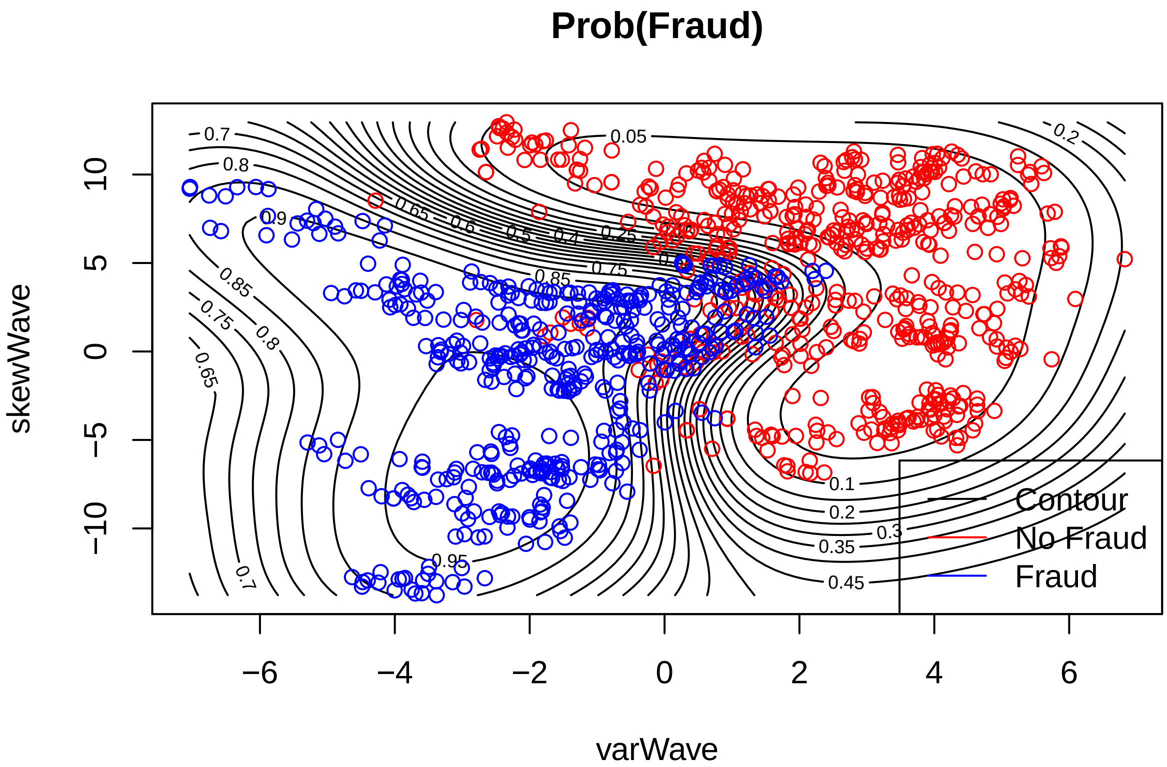
Q4.1

```r
data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/banknoteFraud
names(data) <- c("varWave","skewWave","kurtWave","entropyWave","fraud")
data[,5] <- as.factor(data[,5])

set.seed(111)
SelectTraining <- sample(1:dim(data)[1], size = 1000, replace = FALSE)
train=data[SelectTraining,]
test=data[-SelectTraining,]


GPClassifier <- gausspr(fraud ~  varWave + skewWave, data=train)
```

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```r
# class probabilities
probPreds <- predict(GPClassifier, train[,1:2], type="probabilities")
x1 <- seq(min(train[,1]),max(train[,1]),length=100)
x2 <- seq(min(train[,2]),max(train[,2]),length=100)
gridPoints <- meshgrid(x1, x2)
gridPoints <- cbind(c(gridPoints$x), c(gridPoints$y))

gridPoints <- data.frame(gridPoints)
names(gridPoints) <- names(train)[1:2]
probPreds <- predict(GPClassifier, gridPoints, type="probabilities")

# Plotting for Prob(setosa)
contour(x1,x2,matrix(probPreds[,2],100,byrow = TRUE), 20, xlab = "varWave", ylab = "skewWave", main = 'P
```

```r
points(train[train[,5]=='0',1],train[train[,5]=='0',2],col="red")
points(train[train[,5]=='1',1],train[train[,5]=='1',2],col="blue")
legend("bottomright", legend=c("Contour","No Fraud", "Fraud"),
       lwd=c(1,1,1), col=c("black","red","blue"))
```




```r
print('Confusion matrix and accuracy of training set')
```

```
## [1] "Confusion matrix and accuracy of training set"
```

```r
confmatrix <- table(predict(GPClassifier,train[,1:2]), train[,5])
confmatrix
```

```
##
##       0   1
##   0 503  18
##   1  41 438
```

```r
sum(diag(confmatrix)) / sum(confmatrix)
```

```
## [1] 0.941
```

```r
print('Confusion matrix and accuracy of test set')
```

```
## [1] "Confusion matrix and accuracy of test set"
```

```r
confmatrix <- table(predict(GPClassifier,test[,1:2]), test[,5])
confmatrix
```

```
##
##       0   1
##   0 199   9
##   1  19 145
```

```
sum(diag(confmatrix)) / sum(confmatrix)
```

```
## [1] 0.9247312
```

Q4.3

```
GPClassifier <- gausspr(fraud ~ ., data=train)
```

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```
print('Confusion matrix and accuracy of test set with all 4 features')
```

```
## [1] "Confusion matrix and accuracy of test set with all 4 features"
```

```
confmatrix <- table(predict(GPClassifier,test[,1:4]), test[,5])
confmatrix
```

```
##
##       0   1
##   0 216   0
##   1   2 154
```

```
sum(diag(confmatrix)) / sum(confmatrix)
```

```
## [1] 0.9946237
```