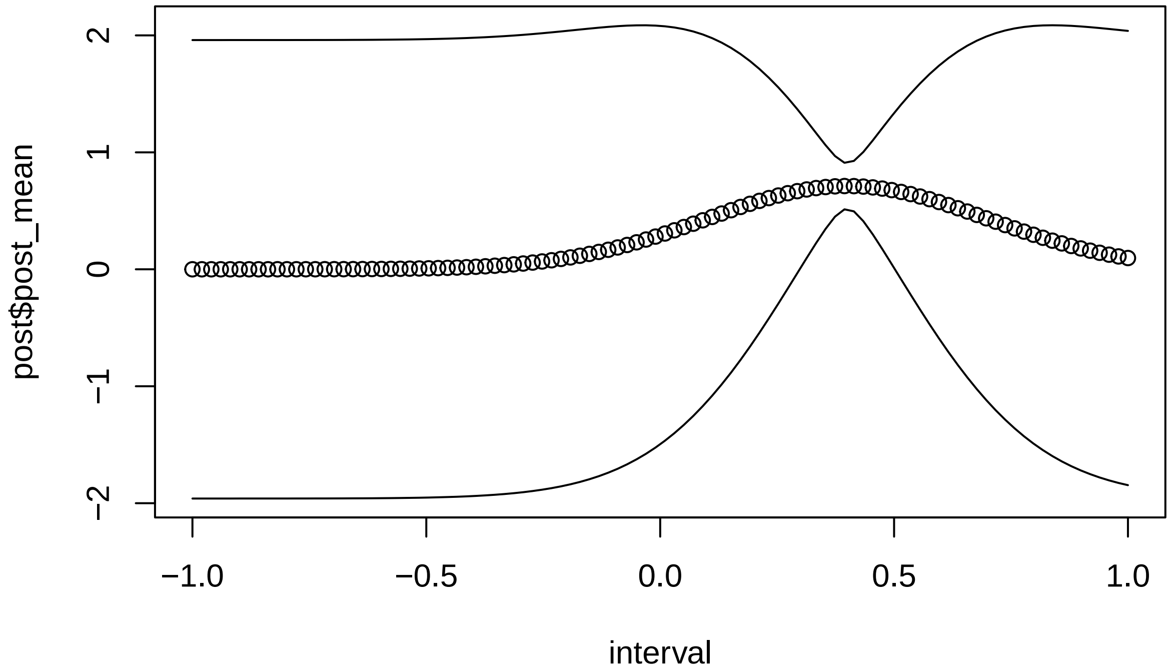
Cajsa_Schöld

2024-10-14

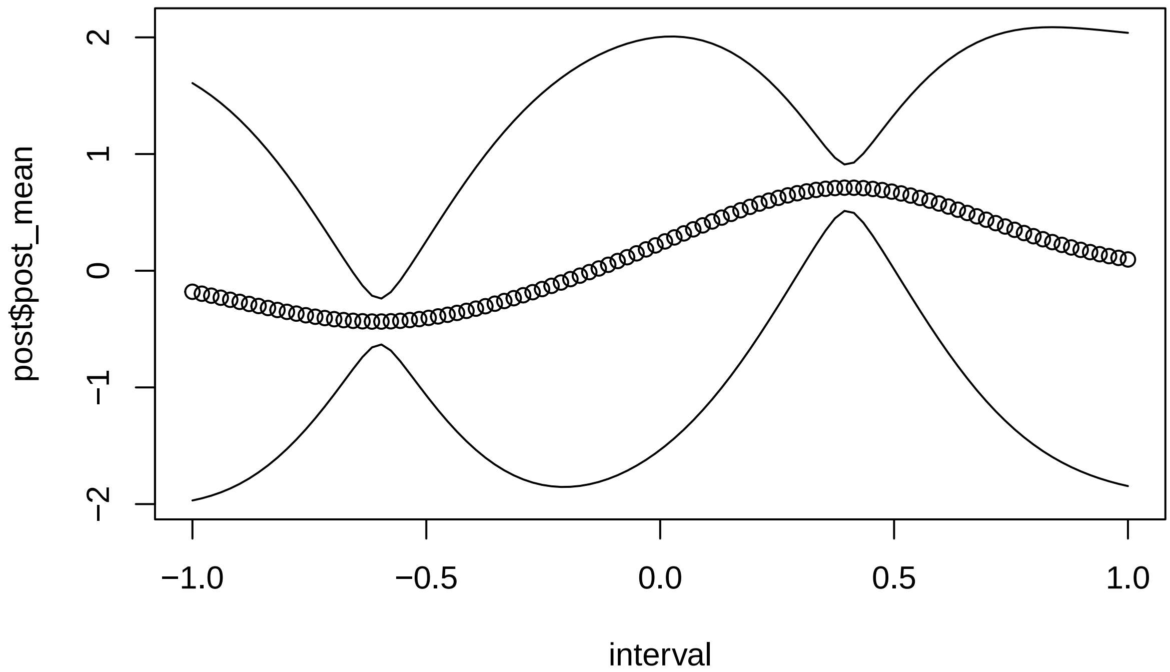
TDDE15 Lab 3

```r
# install.packages("mvtnorm")
library("mvtnorm")

# Covariance function
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)
}


# 1.1

# Posterior Distribution
posteriorGP <- function(x, y, XStar, sigmaNoise, k, ...) {
  n = length(x)
  inv = solve(k(x, x, ...) + sigmaNoise^2*diag(n))
  post_mean = k(XStar, x, ...) %*% inv %*% y
  post_cov = k(XStar, XStar, ...) - k(XStar, x, ...)%*% inv %*% k(x, XStar, ...)
  return(list(post_mean = post_mean, post_cov = post_cov))
}



# 1.2

sigmaF = 1
l = 0.3
obs_x = c(0.4)
obs_y = c(0.719)
sigma_n = 0.1

interval = seq(-1, 1, length.out = 100)

post = posteriorGP(obs_x, obs_y, interval, sigma_n, SquaredExpKernel, sigmaF, l)

# 95% probability bands
post_std = sqrt(diag(post$post_cov))
upper = post$post_mean + 1.96 * post_std
```

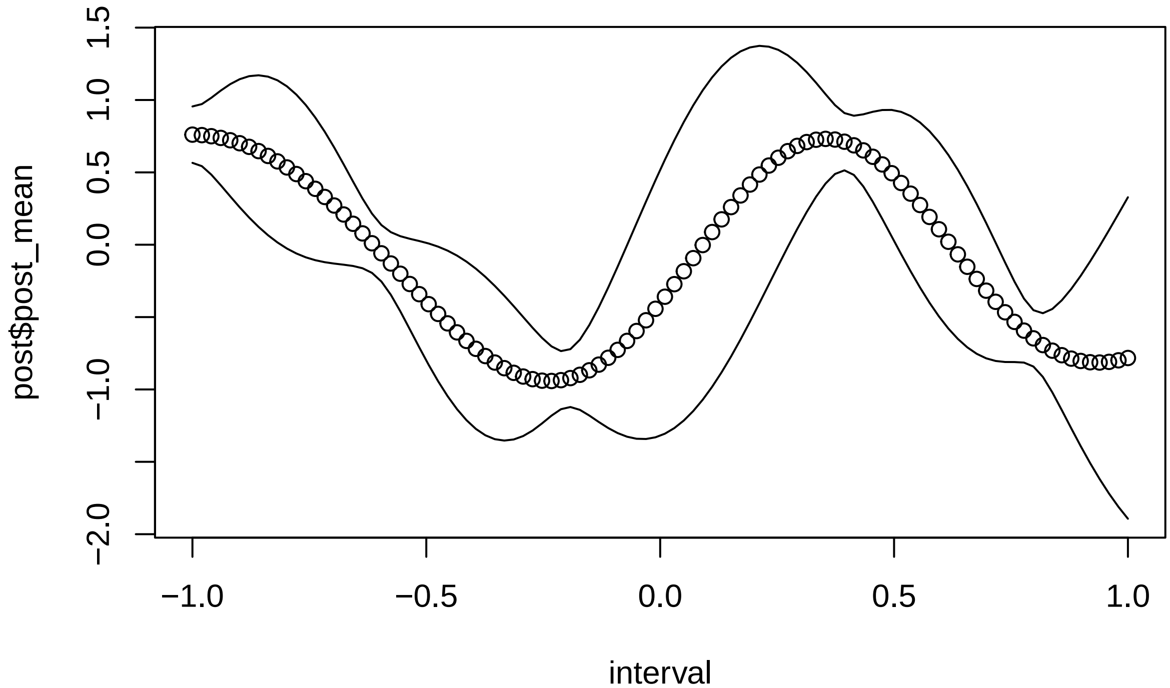
```r
lower = post$post_mean - 1.96 * post_std

plot(interval, post$post_mean, ylim = c(min(lower), max(upper)))
lines(interval, upper)
lines(interval, lower)
```



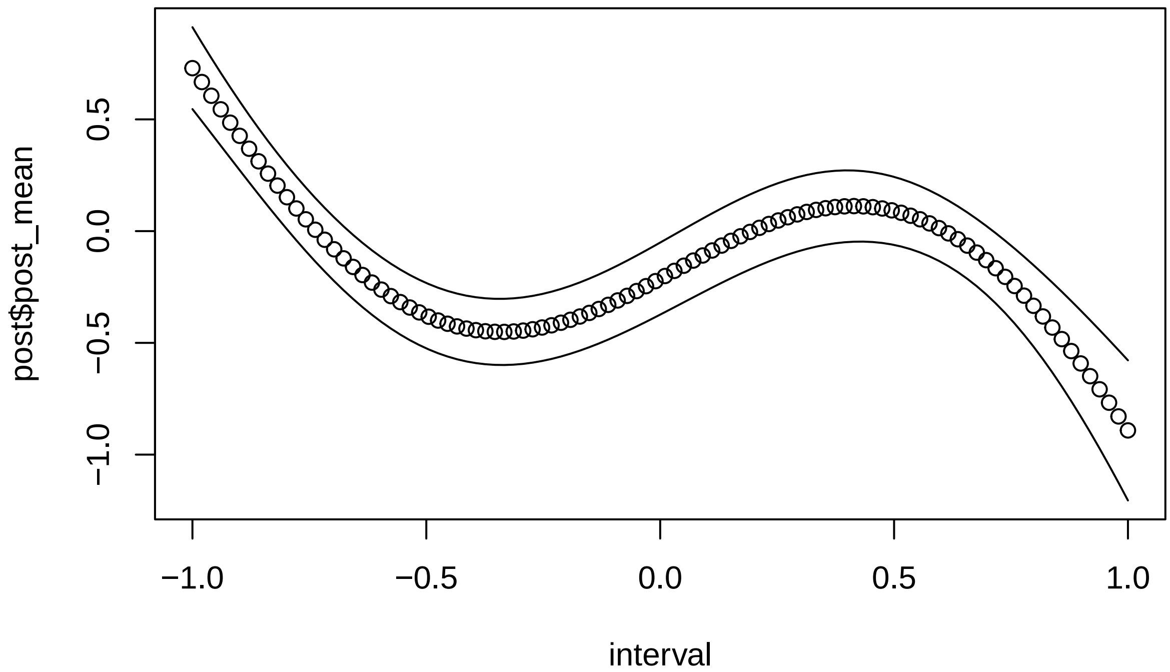

```r
# 1.3

obs_x = append(obs_x, -0.6)
obs_y = append(obs_y, -0.44)

post = posteriorGP(obs_x, obs_y, interval, sigma_n, SquaredExpKernel, sigmaF, l)

# 95% probability bands
post_std = sqrt(diag(post$post_cov))
upper = post$post_mean + 1.96 * post_std
lower = post$post_mean - 1.96 * post_std

plot(interval, post$post_mean, ylim = c(min(lower), max(upper)))
lines(interval, upper)
lines(interval, lower)
```




```
# 1.4

obs_x = c(-1, -0.6, -0.2, 0.4, 0.8)
obs_y = c(0.768, -0.044, -0.94, 0.719, -0.664)

post = posteriorGP(obs_x, obs_y, interval, sigma_n, SquaredExpKernel, sigmaF, l)

# 95% probability bands
post_std = sqrt(diag(post$post_cov))
upper = post$post_mean + 1.96 * post_std
lower = post$post_mean - 1.96 * post_std

plot(interval, post$post_mean, ylim = c(min(lower), max(upper)))
lines(interval, upper)
lines(interval, lower)
```




```
# 1.5

sigmaF = 1
l = 1

post = posteriorGP(obs_x, obs_y, interval, sigma_n, SquaredExpKernel, sigmaF, l)

# 95% probability bands
post_std = sqrt(diag(post$post_cov))
upper = post$post_mean + 1.96 * post_std
lower = post$post_mean - 1.96 * post_std

plot(interval, post$post_mean, ylim = c(min(lower), max(upper)))
lines(interval, upper)
lines(interval, lower)
```



We can see that the new plot is a lot smoother, this is because l is the smoothing factor and a higher value for l gives smoother confidence bands.

Part 2

```
data = read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/TempTullinge.c

time = seq(1, 2190, by = 5)
year = seq(1,365)
day = rep(year, times = 6)
day = day[time]
temp = data$temp[time]

# 2.1

# install.packages("kernlab")
library(kernlab)

l = 1
sigmaF = 1

new_SquaredExpKernel <- function(sigmaF, l) {
  inner <- function(x, XStar) {
    n1 <- length(x)
    n2 <- length(XStar)
    K <- matrix(NA,n1,n2)
    for (i in 1:n2){
      K[,i] <- sigmaF^2*exp(-0.5*( (x-XStar[i])/l)^2 )
    }
    return(K)
  }
  class(inner) = 'kernel'
```

```r
  return(inner)
}

new_kernel = new_SquaredExpKernel(sigmaF, l)
new_kernel(1,2)
```

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

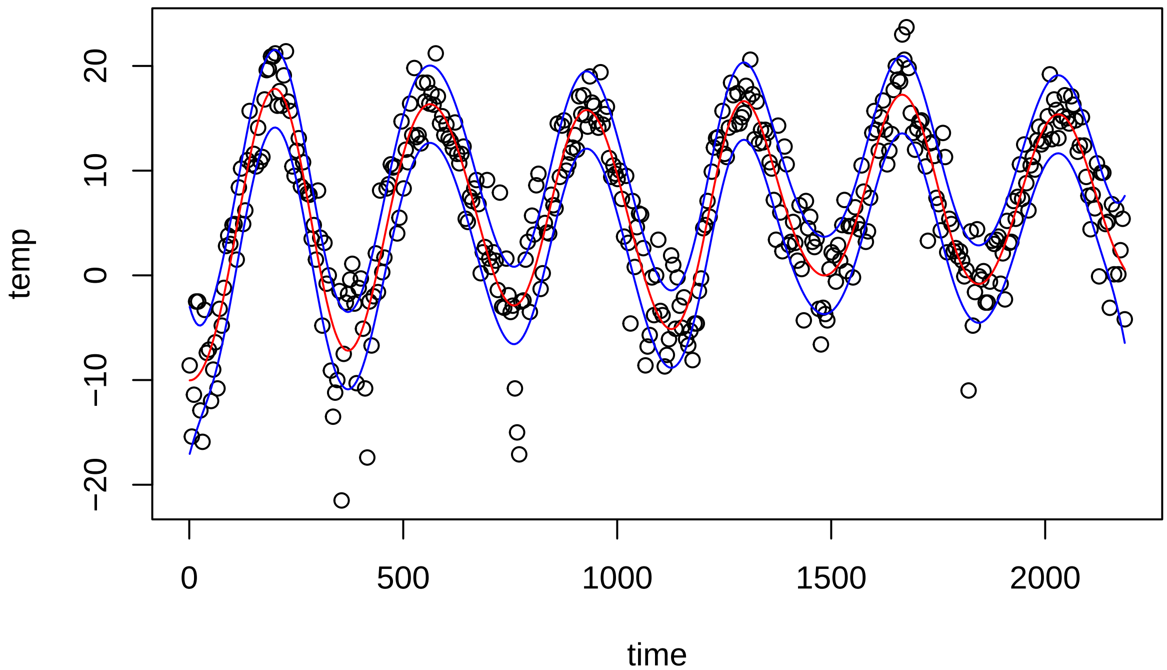
```r
x = c(1,3,4)
XStar = c(2,3,4)

kernelMatrix(kernel = new_kernel, 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
```

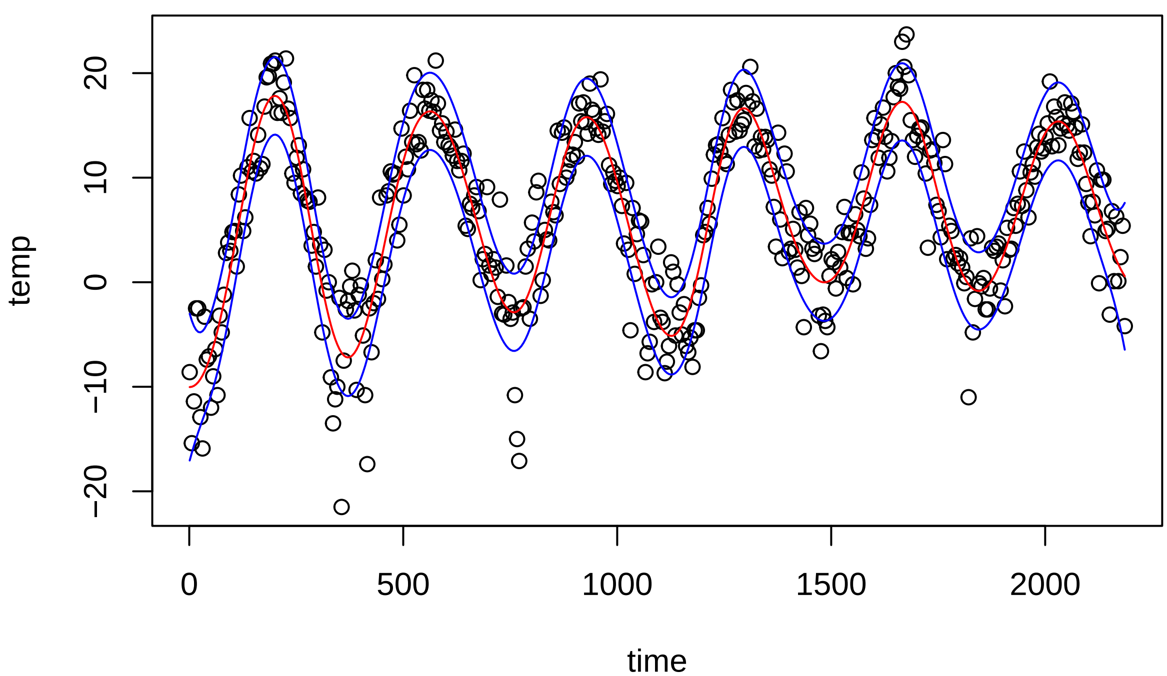
```r
# 2.2
data = data.frame(time, temp)

quad_regression = lm(temp ~ (time + time^2))
sigma_n = sd(quad_regression$residuals)

sigmaF = 20
l = 100

new_kernel = new_SquaredExpKernel(sigmaF, l)
new_kernel(1,2)
```

```
##        [,1]
## [1,] 399.98
```

```r
gaussian = gausspr(time, temp, kernel = new_kernel, var = sigma_n^2, scaled = FALSE, variance.model = TR

preds = predict(gaussian, time)
pred_variance = predict(gaussian, time, type = "variance")


plot(time, temp)
lines(time, preds, type="l", col="red")
lines(time, preds + 1.96*sqrt(pred_variance), type="l", col="blue")
lines(time, preds - 1.96*sqrt(pred_variance), type="l", col="blue")
```

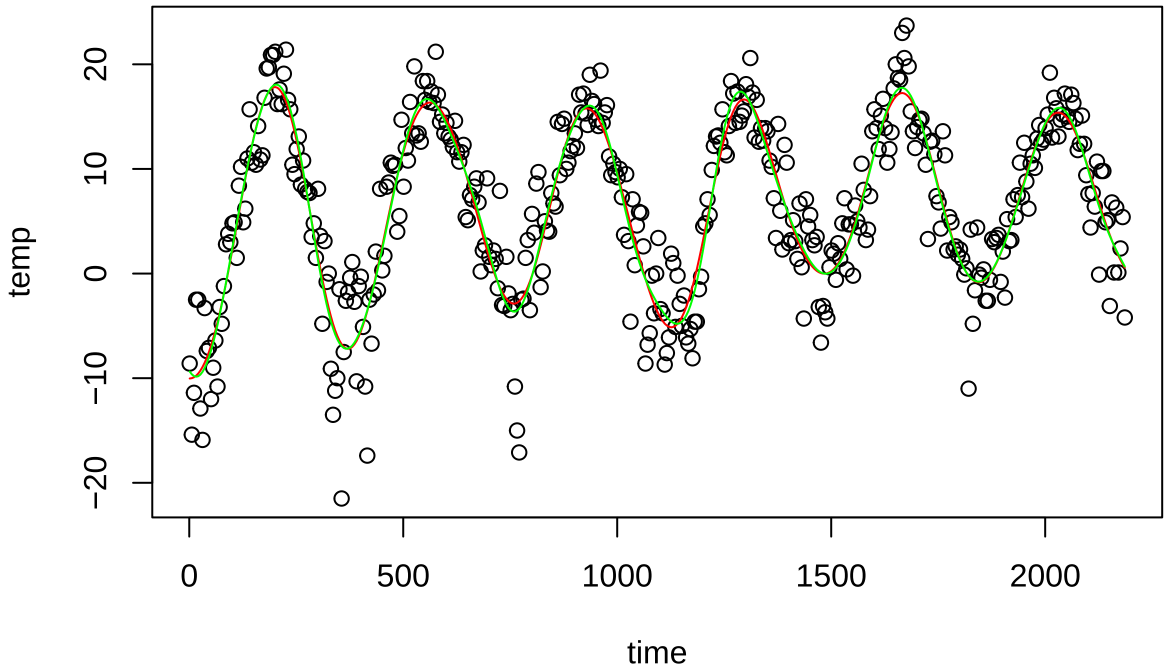



```r
# 2.3

post = posteriorGP(time, temp, time, sigmaNoise = sigma_n, SquaredExpKernel, sigmaF= 20, l=100)

plot(time, temp)
lines(time, post$post_mean, type="l", col="red")
lines(time, post$post_mean + 1.96*sqrt(diag(post$post_cov)), type="l", col="blue")
lines(time, post$post_mean - 1.96*sqrt(diag(post$post_cov)), type="l", col="blue")
```

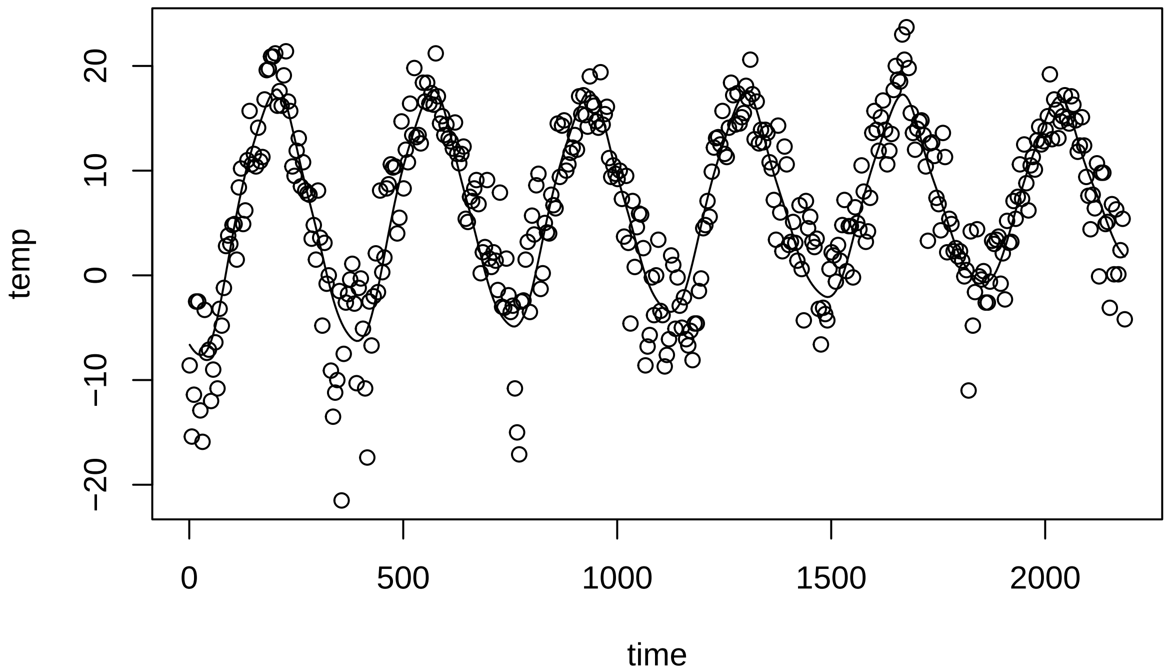



```r
# 2.4
```

```r
post2 = gausspr(time, temp, kernel=new_SquaredExpKernel(20, 100), scaled = FALSE)
post2_mean = predict(post2, newdata = time)

plot(time, temp)
lines(time, post$post_mean, type="l", col="red")
lines(time, post2_mean, type="l", col="green")
```




```r
# 2.5

periodic_kernel <- function(sigmaF, l1, l2, d) {
  inner <- function(x, XStar) {
    diff = abs(x - XStar)
    exp1 = exp(-2 * (sin(pi * diff / d)^2) / (l1^2))
    exp2 = exp(-0.5 * (diff^2) / (l2^2))
    return(sigmaF^2 * exp1 * exp2)
  }
  class(inner) = 'kernel'
  return(inner)
}

gp_pk = gausspr(time, temp, kernel=periodic_kernel(sigmaF = 20, l1=1, l2=10, d=365/sd(time)), var=sigma

post_pk = predict(gp_pk, newdata = time)

plot(time, temp)
lines(time, post_pk)
```



The periodic model should look more "spiky" than the gaussian one (but mine does not, so I have some mistake in my code).

Part 3

You can also embed plots, for example:

```
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, ]


# 3.1

gp = gausspr(fraud ~ varWave + skewWave, data = train)
```

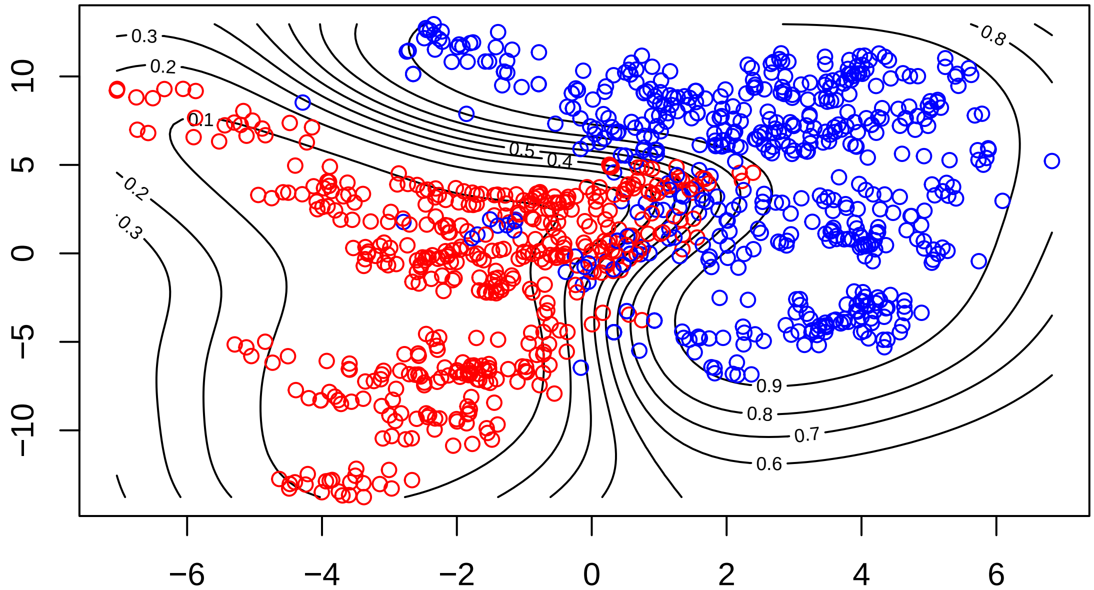
```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```
fraud_pred = predict(gp, train[,1:2])

table(fraud_pred, train$fraud)
```

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

```
x1 = seq(min(train$varWave), max(train$varWave), length = 100)
x2 = seq(min(train$skewWave), max(train$skewWave), length = 100)

grid = expand.grid(varWave = x1, skewWave = x2)
```

```r
prob_grid <- predict(gp, grid, type = "probabilities")

contour(x1, x2, matrix(prob_grid[, 1], 100, 100))
points(train$varWave, train$skewWave, col = ifelse(train$fraud == 1, "red", "blue"))
```




```r
# 3.2

y_pred = predict(gp, newdata=test)
cm = table(y_pred, test$fraud)
cm
```

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

```r
acc = sum(diag(cm)) / sum(cm)
acc
```

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

```r
# 3.3

gp2 = gausspr(fraud ~ ., data = train)
```

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

```r
y_pred2 = predict(gp2, newdata=test)

cm2 = table(y_pred2, test$fraud)
cm2
```

```
##
## y_pred2   0   1
##       0 216   0
##       1   2 154
```

```r
acc2 = sum(diag(cm2)) / sum(cm2)
acc2
```

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

We can see that when including all 4 covarieties we get a better test accuracy meaning that the model is better at generalizing.