lab4.rmd

10/6/2022

contribution

Used Isak Berntsson as base, minor changes made.

Answers given after group discussions. isabe723, erija971, ludkn080, linbe580

Task1

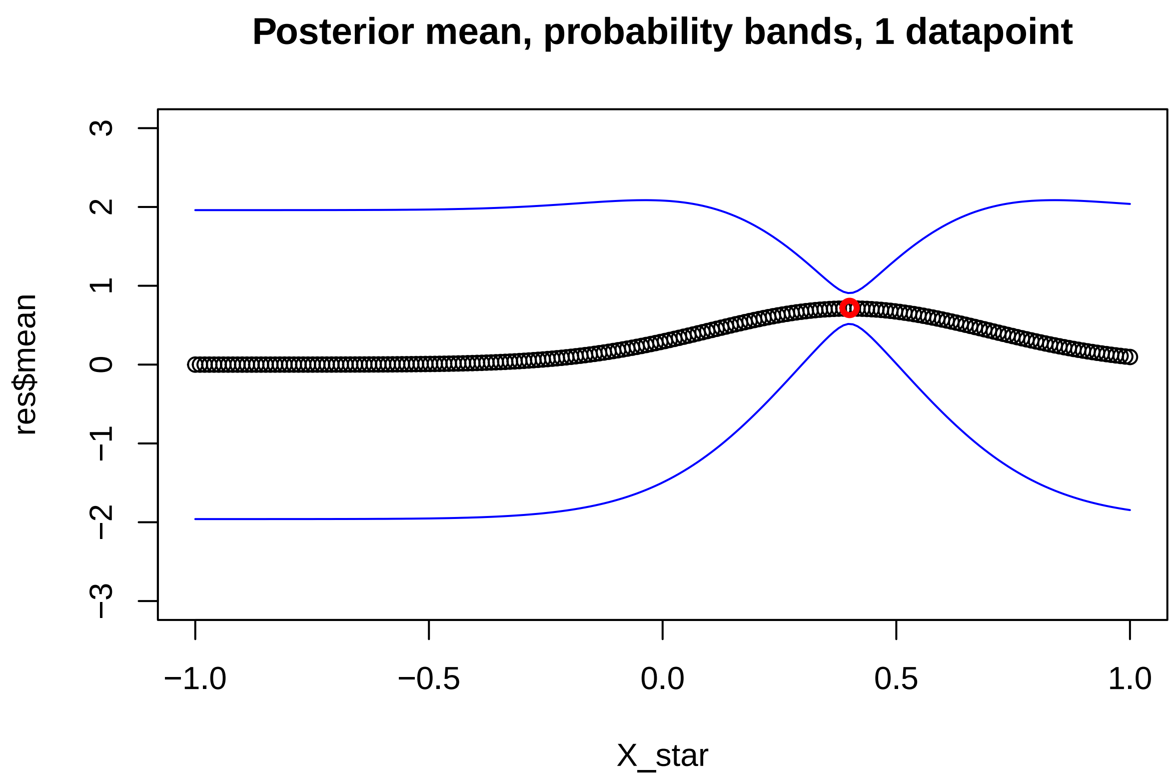
Subtask 1

```
posteriorGP = function(X, y, X_star, sigmaNoise, k, ...){
  n = length(X)

  K = k(X,X, ... )

  k_star = k(X,X_star, ... )

  L = t(chol(K + sigmaNoise^2*diag(n)))

  alpha = solve(t(L),solve(L,y))

  mean_pred = t(k_star) %*% alpha #predictive mean

  v = solve(L,k_star)

  var_pred= k(X_star, X_star) - t(v) %*% v # predictive var

  logmarglik = -.5 * t(y) %*% alpha - sum(log(diag(L))) - .5*n*log(2*pi)

  return(list(mean=mean_pred, var= var_pred, logmarglik = logmarglik))

}
```

Subtask 2

```
#priors
sigma_f = 1
sigma_n = .1
l = .3
```

```r
#k = SquaredExpKernel #using func from given script

X = .4
y = .719
X_star = seq(-1,1,length=200)

res = posteriorGP(X,y,X_star, sigma_n,SquaredExpKernel,sigmaF= sigma_f, l)

plot(X_star, res$mean, ylim = c(-3,3), main="Posterior mean, probability bands, 1 datapoint")
lines(X_star, res$mean + 1.96*sqrt(diag(res$var)), col="blue")
lines(X_star, res$mean - 1.96*sqrt(diag(res$var)), col="blue")
points(X,y, col="red", lwd=3)
```



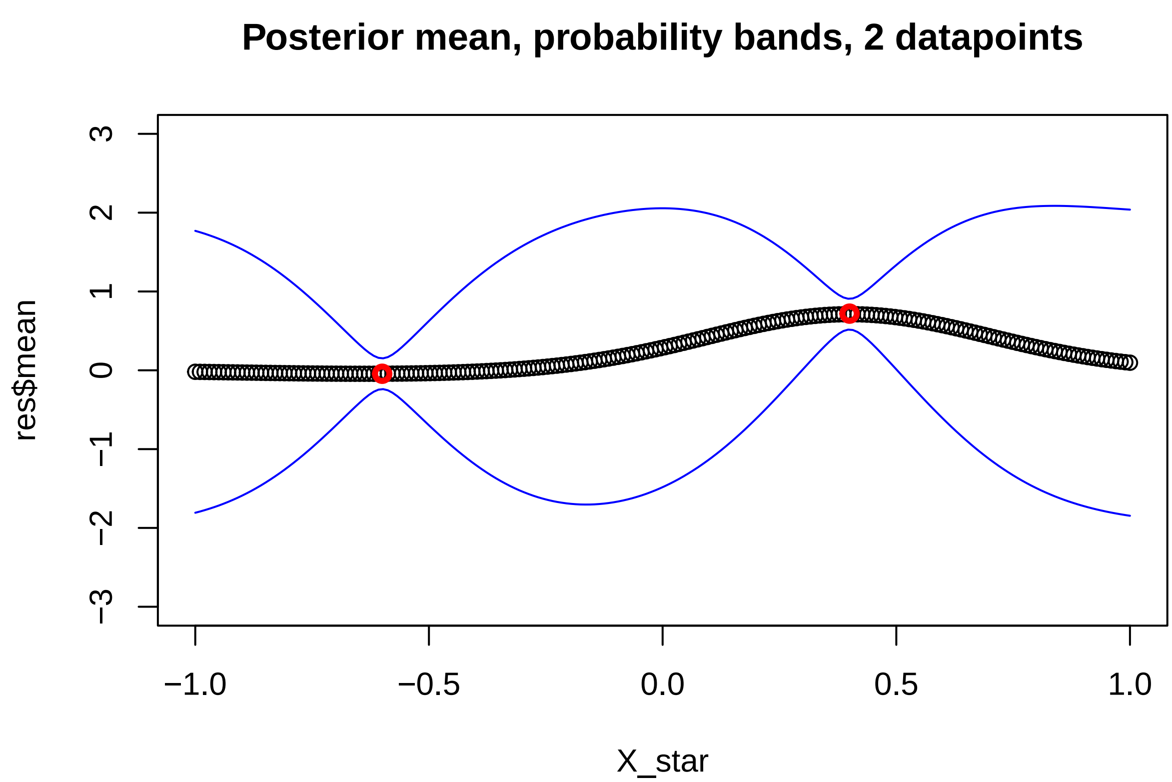
Subtask 3

```r
sigma_f = 1
sigma_n = .1
l = .3

dat = data.frame(X=c(X,-.6), y = c(y,-.044))
dat
```

```
##      X       y
```

```
## 1  0.4  0.719
## 2 -0.6 -0.044
```

```
res = posteriorGP(dat$X,dat$y,X_star, sigma_n,SquaredExpKernel,sigmaF= sigma_f, l)

plot(X_star, res$mean, ylim = c(-3,3), main="Posterior mean, probability bands, 2 datapoints")
lines(X_star, res$mean + 1.96*sqrt(diag(res$var)), col="blue")
lines(X_star, res$mean - 1.96*sqrt(diag(res$var)), col="blue")
points(dat, col="red", lwd=3)
```



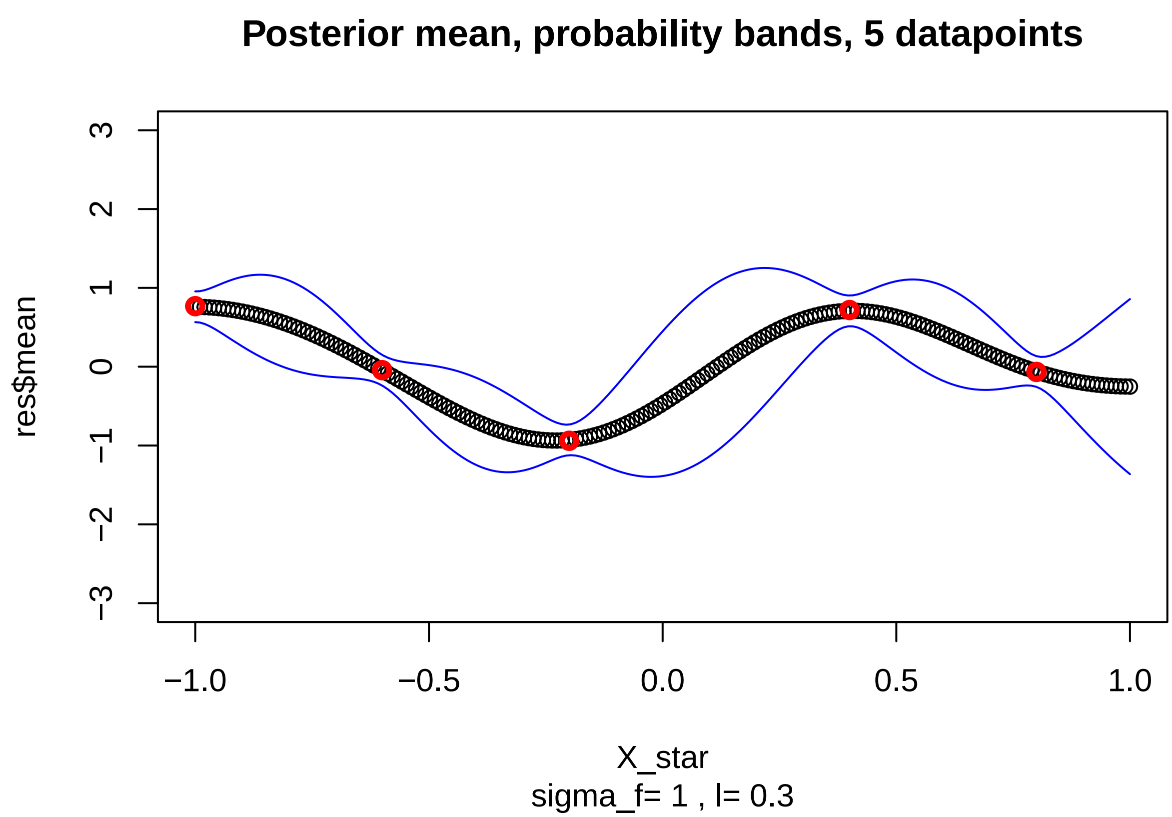
Subtask 4

```
sigma_f = 1
sigma_n = .1
l = .3

dat = data.frame(
  X = c(-1, -.6, -.2, .4, .8),
  y = c(.768, -.044, -.940, .719, -.0664)
  )
dat
```

```
##       X       y
```

```
## 1 -1.0  0.7680
## 2 -0.6 -0.0440
## 3 -0.2 -0.9400
## 4  0.4  0.7190
## 5  0.8 -0.0664
```

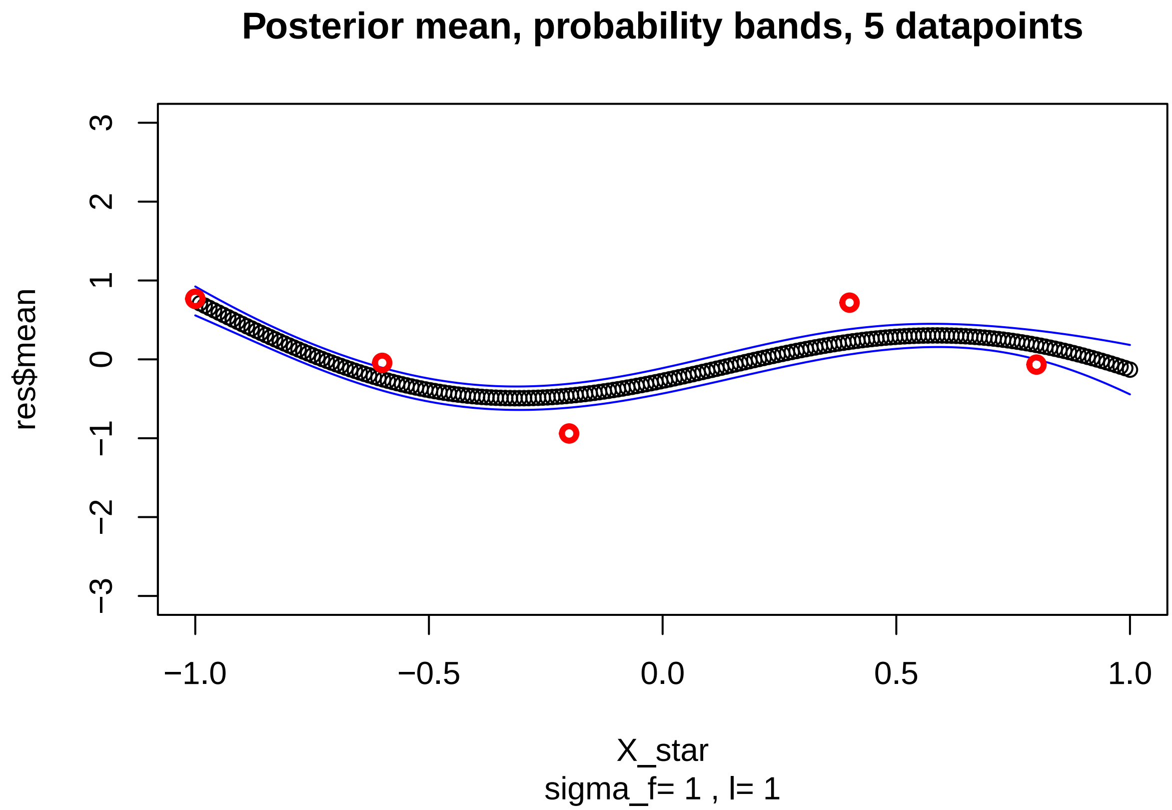
```
subtext = paste("sigma_f=", sigma_f, ", l=",l)

res = posteriorGP(dat$X,dat$y,X_star, sigma_n,SquaredExpKernel,sigmaF= sigma_f, l)

plot(X_star, res$mean, ylim = c(-3,3), main="Posterior mean, probability bands, 5 datapoints", sub=subte
lines(X_star, res$mean + 1.96*sqrt(diag(res$var)), col="blue")
lines(X_star, res$mean - 1.96*sqrt(diag(res$var)), col="blue")
points(dat, col="red", lwd=3)
```



Subtask 5

```
sigma_f = 1
l = 1

res = posteriorGP(dat$X,dat$y,X_star, sigma_n,SquaredExpKernel,sigmaF= sigma_f, l)

subtext = paste("sigma_f=", sigma_f, ", l=",l)
```

```
plot(X_star, res$mean, ylim = c(-3,3), main="Posterior mean, probability bands, 5 datapoints", sub=subt
lines(X_star, res$mean + 1.96*sqrt(diag(res$var)), col="blue")
lines(X_star, res$mean - 1.96*sqrt(diag(res$var)), col="blue")
points(dat, col="red", lwd=3)
```

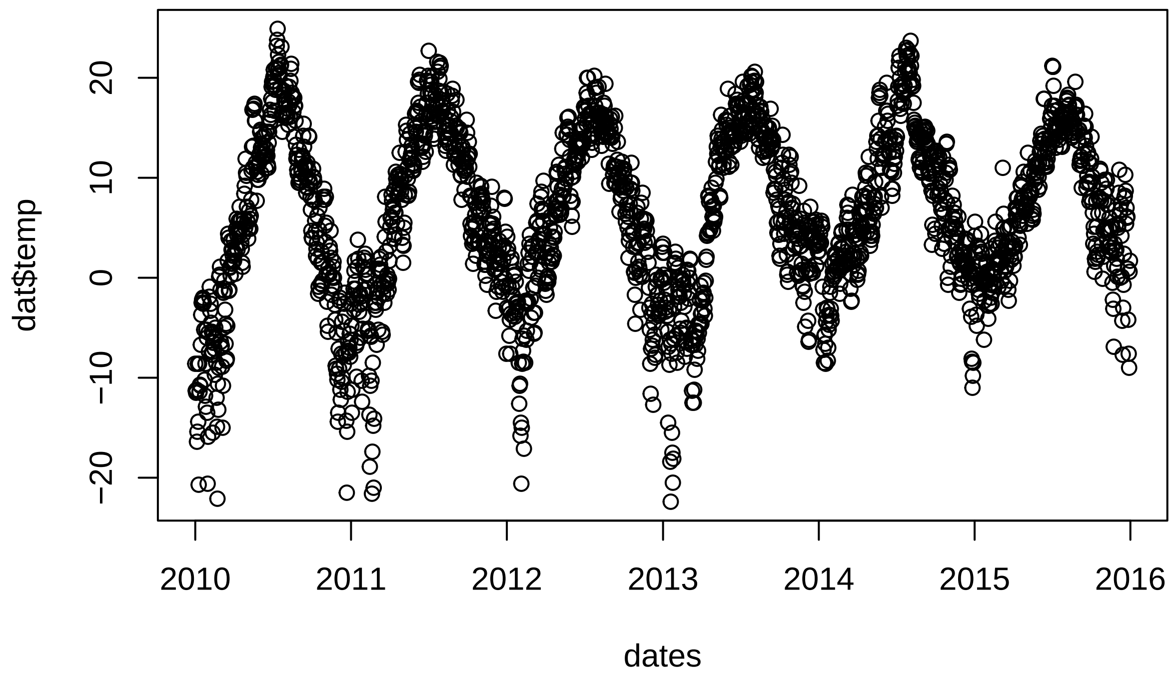


THe higher value of ell/l mean the function is overly smoothed and cannot fit the data as well as in (4)

Task 2

setup

```r
dat = read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/
Code/TempTullinge.csv", header=TRUE, sep=";")

dates = as.Date(dat$date, "%d/%m/%y")
plot(dates, dat$temp)
```




```r
n = length(dates)
time = 1:n
day = time %% 365
day[365:n] = day[365:n]+1

sample_ind = time %% 5 == 1

sample_time = time[sample_ind]
sample_day = day[sample_ind]
sample_temp = dat$temp[sample_ind]


sample_dat = data.frame(time = sample_time, day  = sample_day, temp = sample_temp)
```

subtask 1

```
library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 4.1.3
```

```
#library(AtmRay)




SEK = function(ell, sigmaf){

  SEK_kernel = function(X,X_star){

    X = as.matrix(X)
    X_star = as.matrix(X_star)

    res = matrix(0,nrow=length(X),ncol=length(X_star))

    #print(dim(res))

    for( i in 1:length(X)){
      vals = sigmaf^2 * exp(-.5 * ((X-X_star[i,])/ell)^2)

      res[i,] = vals
    }

    return(res)
  }
  class(SEK_kernel) = "kernel"
  return(SEK_kernel)
}

mu = 1
X = runif(100,-5,5)

kern= SEK(ell=1,sigmaf=1)

print(paste("kernel evaluated at (1,2)",kern(1,2)))
```

```
## [1] "kernel evaluated at (1,2) 0.606530659712633"
```

```
X = c(1,3,4)
Xprim = c(2,3,4)

kmat = kernelMatrix(kern,X,Xprim)

colnames(kmat) = X
rownames(kmat) = Xprim
print(kmat)
```

```
## An object of class "kernelMatrix"
##           1         3         4
## 2 0.6065307 0.1353353 0.0111090
## 3 0.6065307 1.0000000 0.6065307
## 4 0.1353353 0.6065307 1.0000000
```

subtask 2

```r
df_fit = data.frame(temp = sample_dat$temp, const = 1, time= sample_dat$time, time2 = (sample_dat$time)
head(df_fit)
```

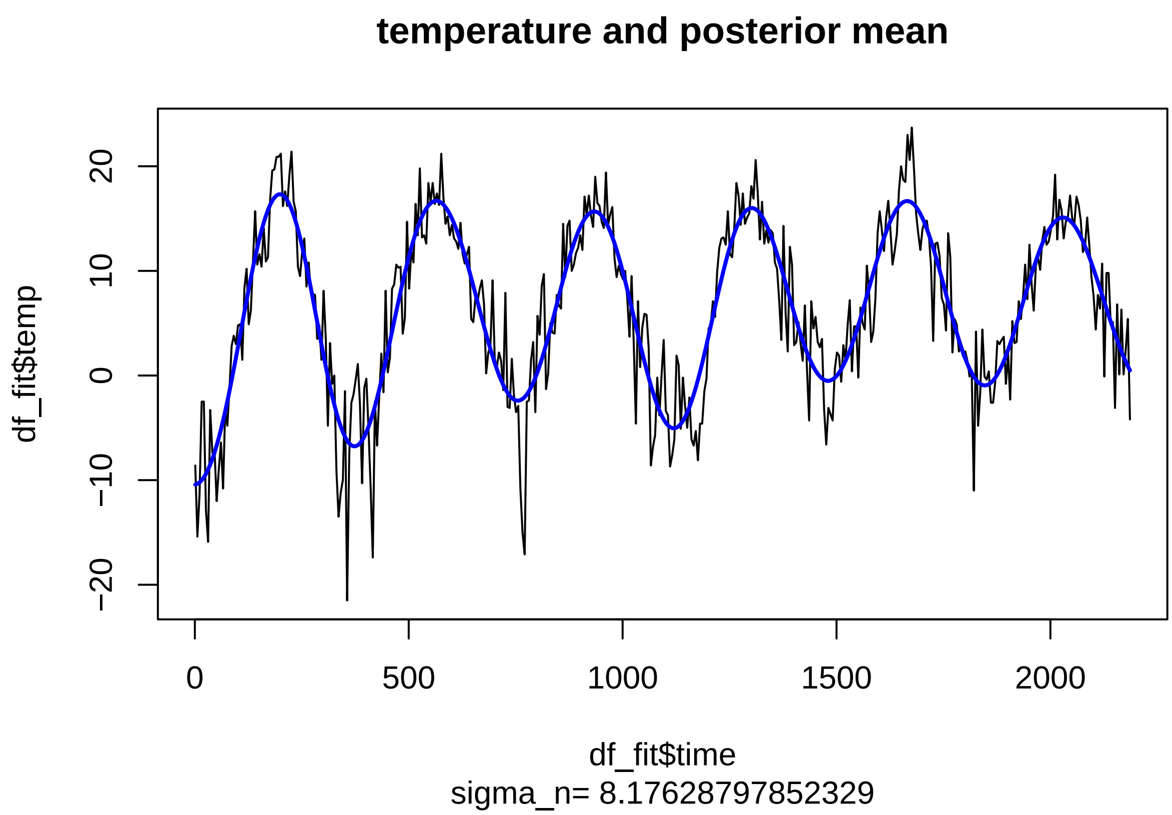
```
##    temp const time time2
## 1  -8.6     1    1     1
## 2 -15.4     1    6    36
## 3 -11.4     1   11   121
## 4  -2.5     1   16   256
## 5  -2.5     1   21   441
## 6 -12.9     1   26   676
```

```r
linmod = lm("temp ~ const + time + time2 - 1 ",data = df_fit)
linmod
```

```
##
## Call:
## lm(formula = "temp ~ const + time + time2 - 1 ", data = df_fit)
##
## Coefficients:
##      const         time        time2
##  2.892e+00    5.319e-03   -1.453e-06
```

```r
linearPrediction = predict(object = linmod, data=df_fit)

residuals = linearPrediction - df_fit$temp
sigma_n = sd(residuals)

#used for descaling later
y_Sd = sd(df_fit$temp)
y_m = mean(df_fit$temp)


ell = .2
sigma_f = 20
kern = SEK(ell,sigmaf = sigma_f)


mod = gausspr(df_fit$time, df_fit$temp , kernel=kern, var = sigma_n^2)
posterior_mean = predict(mod, df_fit$time)



plot(df_fit$time,df_fit$temp, main="temperature and posterior mean",sub=paste("sigma_n=",sigma_n), type=
```

```r
lines(df_fit$time,posterior_mean, col="blue", lwd=2)
```




```r
'legend(
  legend =c("measured temperature", "posterior Mean"),
  col = c("black", "blue"),
  lty=c("l","l"),
  lwd = c(2,2),
  inset=0.05
)'
```

```
## [1] "legend( \n  legend =c(\"measured temperature\", \"posterior Mean\"),\n  col = c(\"black\", \"blu
```

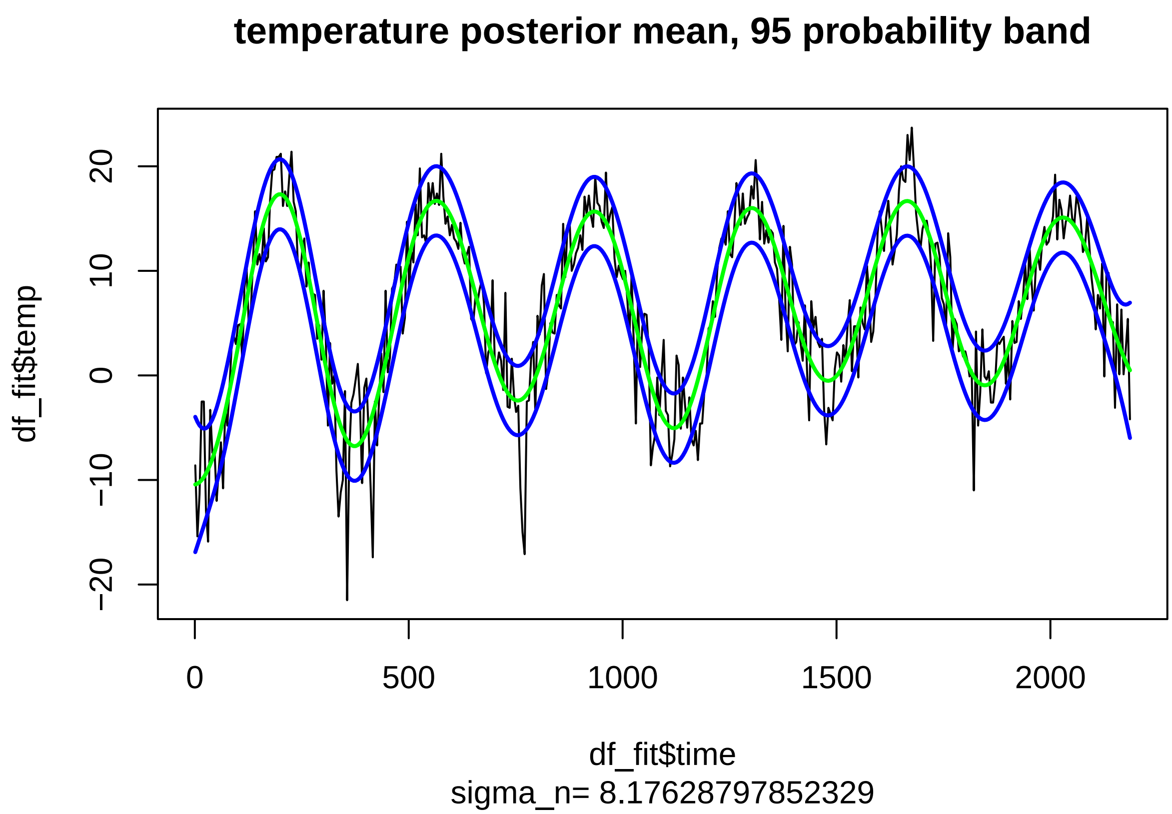
```r
#old_posterior_mean = yhat #used in subtask 4
```

The posterior follows the data well. The posterior has some trouble in the most extreme cases.

Subtask 3

```r
ell =  .2
sigma_f = 20
```

```
kern = SEK(ell,sigma_f)

posterior = posteriorGP(X = scale(df_fit$time), y = scale(df_fit$temp), X_star = scale(df_fit$time), si

posterior_sigma = sqrt(diag(posterior$var))
yhat = posterior$mean * y_Sd + y_m

plot(df_fit$time,df_fit$temp, type="l", main="temperature posterior mean, 95 probability band", sub=pas

lines(df_fit$time, yhat, col="green", lwd = 2)

lines(df_fit$time, yhat + 1.96 * posterior_sigma, col="blue", lwd=2)
lines(df_fit$time, yhat - 1.96 * posterior_sigma, col="blue", lwd =2)
```




```
yhat_from_3 = yhat
```

Bands seem to capture the variance of the measuremetns weell.

Subtask 4

```
df_fit = data.frame(temp = sample_dat$temp, const = 1, day= sample_dat$day, day2 = (sample_dat$day)^2 )
head(df_fit)
```

```
##    temp const day day2
## 1  -8.6     1   1    1
## 2 -15.4     1   6   36
## 3 -11.4     1  11  121
## 4  -2.5     1  16  256
## 5  -2.5     1  21  441
## 6 -12.9     1  26  676
```

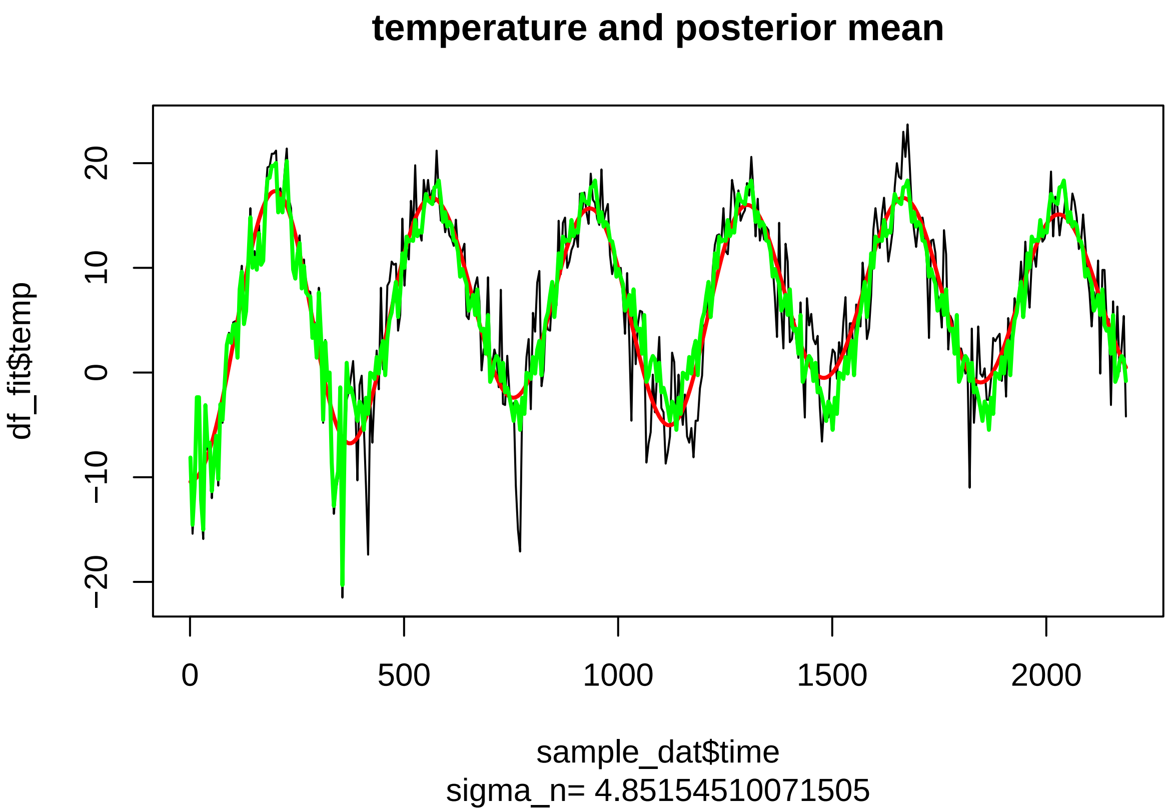
```
mod = lm("temp ~ const + day + day2 - 1 ",data = df_fit)
mod
```

```
##
## Call:
## lm(formula = "temp ~ const + day + day2 - 1 ", data = df_fit)
##
## Coefficients:
##      const          day         day2
## -1.159e+01   2.536e-01  -6.372e-04
```

```
yhat = predict(object = mod, data=df_fit)
head(yhat) # posterior mean
```

```
##          1          2          3          4          5          6
## -11.333082 -10.087352  -8.873480  -7.691466  -6.541309  -5.423010
```

```
residuals = yhat - df_fit$temp
sigma_n = sd(residuals)

ell = .2
sigma_f = 20
kern = SEK(ell,sigmaf = sigma_f)

posterior = posteriorGP(X = df_fit$day, y = df_fit$temp, X_star = df_fit$day, sigmaNoise = sigma_n ,k =

plot(sample_dat$time,df_fit$temp, type="l", main="temperature and posterior mean", sub=paste("sigma_n="


lines(sample_dat$time,yhat_from_3, col="red", lwd=2)
lines(sample_dat$time,posterior$mean, col="green", lwd =2)
```



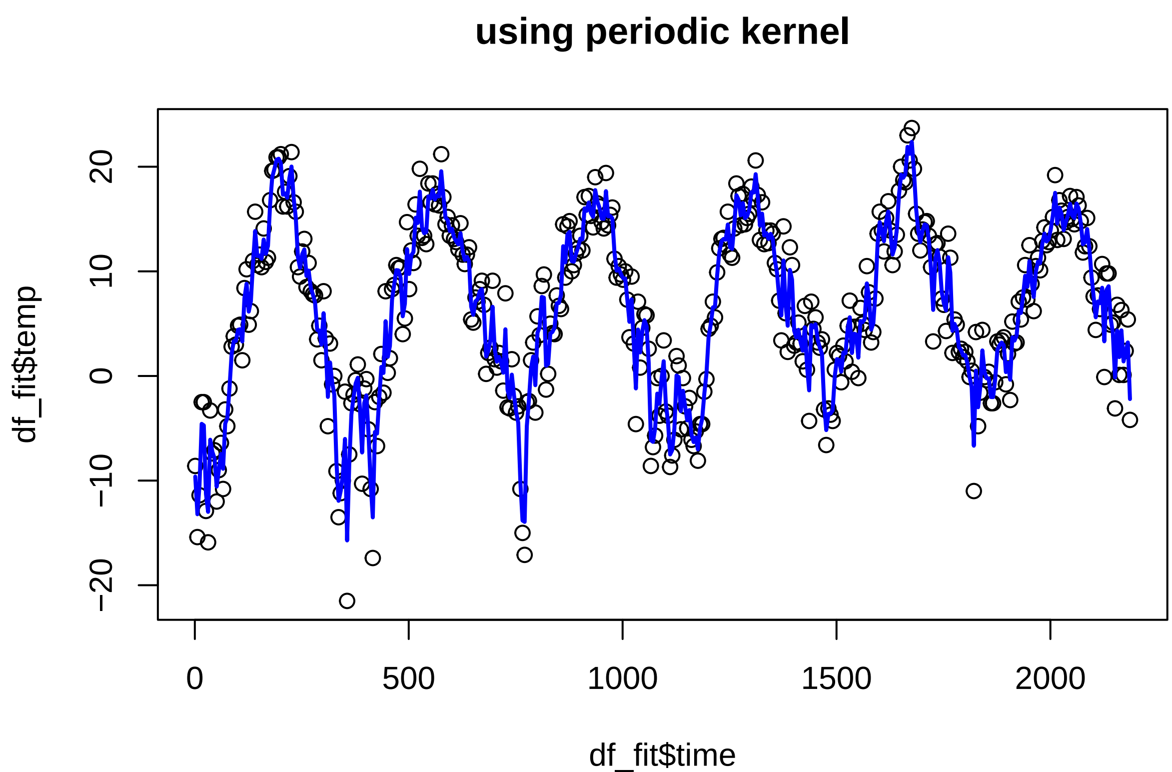
The new models is not as smooth as the old one. The fitted function fits the data better.

sigma_n takes on a much smaller value with the new model. 8.17 -> 4.85 because the residuals from the linear model are much smaller / fit is better.

Subtask 5

```
df_fit = data.frame(temp = sample_dat$temp, const = 1, time= sample_dat$time, time2 = (sample_dat$time)


periodicKernelFunction <- function(sigmaf, l1, l2, d)
{
  periodicKernel <- function(x, xStar)
  {
    tmp = abs(x-xStar)
    p1 = sigmaf^2*exp(-( (2*sin(pi*tmp))/d)/l1^2)
    p2 = exp(-0.5 * tmp^2 / l2^2)
    return(p1*p2)
  }
  class(periodicKernel) = "kernel"
  return (periodicKernel)
}
```

```r
sigma_f = 20
l1 = 1
l2 = 10
d = 365/sd(df_fit$time)

kern = periodicKernelFunction(sigma_f,l1, l2, d)



mod = gausspr(x = df_fit$time, y=df_fit$temp, kernel = kern, var= sigma_n^2)

yhat = predict(mod, df_fit$time)

residuals = yhat - df_fit$temp


plot(df_fit$time, df_fit$temp, main ="using periodic kernel")
lines(df_fit$time, yhat, col="blue", lwd=2 )
```



This model generates predictions that are periodic and smoother while still following the general upwards trend.

Task 3

Subtask1

```
data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/
GaussianProcess/Code/banknoteFraud.csv", header=FALSE, sep=",")
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)

df_train = data[SelectTraining,]
head(df_train)
```

```
##      varWave skewWave  kurtWave entropyWave fraud
## 507  2.09300  8.30610  0.022844    -3.27240     0
## 979  0.75896  0.29176 -1.650600     0.83834     1
## 175  1.87990  2.47070  2.493100     0.37671     0
## 793 -2.73380  0.45523  2.439100     0.21766     1
## 699  3.24030 -3.70820  5.280400     0.41291     0
## 69   1.00090  7.78460 -0.282190    -2.66080     0
```

```
mod = gausspr(x = df_train[, 1:2], y=df_train$fraud, type="classification")
```

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

```
yhat_proba = predict(mod, df_train[,1:2], type="probabilities")
yhat_bin = yhat_proba[,2]>.5


conf_mat = table(yhat_bin,df_train$fraud)
conf_mat
```

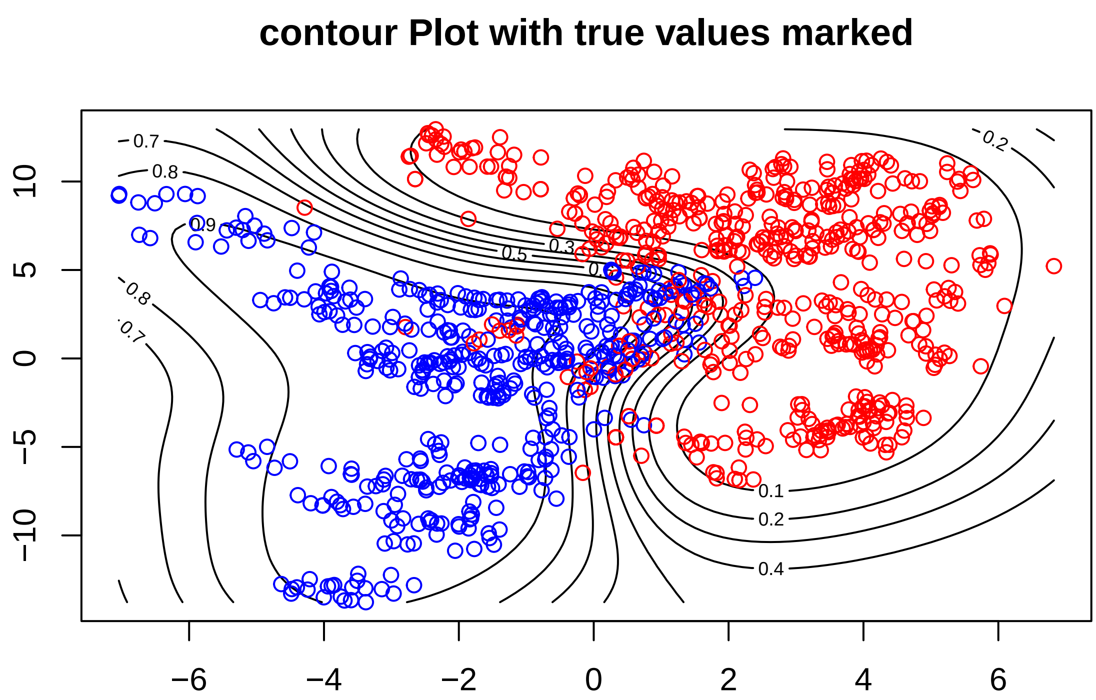
```
##
## yhat_bin   0   1
##    FALSE 503  18
##    TRUE   41 438
```

```
acc = sum(diag(conf_mat)) / sum(conf_mat)
acc
```

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

```
n=100
g1 = seq(min(df_train$varWave),max(df_train$varWave), length.out=n )
g2 = seq(min(df_train$skewWave),max(df_train$skewWave), length.out=n )
```

```r
x1matrix = matrix(nrow=100, ncol=100)
for (i in 1:100){
  x1matrix[i,] = g1
}
x2matrix = matrix(nrow=100, ncol=100, data=g2)
grid <- cbind(c(x1matrix), c(x2matrix))
grid <- data.frame(grid)
names(grid) <- names(data)[1:2]
prediction_prob <- predict(mod, grid, type="probabilities")



fraud_bool = df_train$fraud == 1
cols = array("red",length(fraud_bool))
cols[fraud_bool] = "blue"


#plot()
contour(g1,g2,matrix(prediction_prob[,2],100,byrow = TRUE), 10, main="contour Plot with true values mar
points(df_train$varWave,df_train$skewWave, col=cols)
```



Subtask 2

```
df_test = data[-SelectTraining,c(1,2,5)]

yhat = predict(mod,df_test[,1:2])

conf_mat = table(yhat,df_test$fraud)
conf_mat
```

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

```
acc = sum(diag(conf_mat)) / sum(conf_mat)
acc
```

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

Subtask 3

```
df_train = data[SelectTraining,]
df_test = data[-SelectTraining,]
mod = gausspr(df_train[,-5], df_train[,5], type="classification")
```

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

```
yhat = predict(mod, df_test[,-5])

conf_mat = table(yhat,df_test$fraud)
conf_mat
```

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

```
acc = sum(diag(conf_mat)) / sum(conf_mat)
acc
```

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

Accuracy using the final model is better (99.46% vs 92.47%) in terms of accuracy