Lab 4 - Gaussian Processes

David Björelind

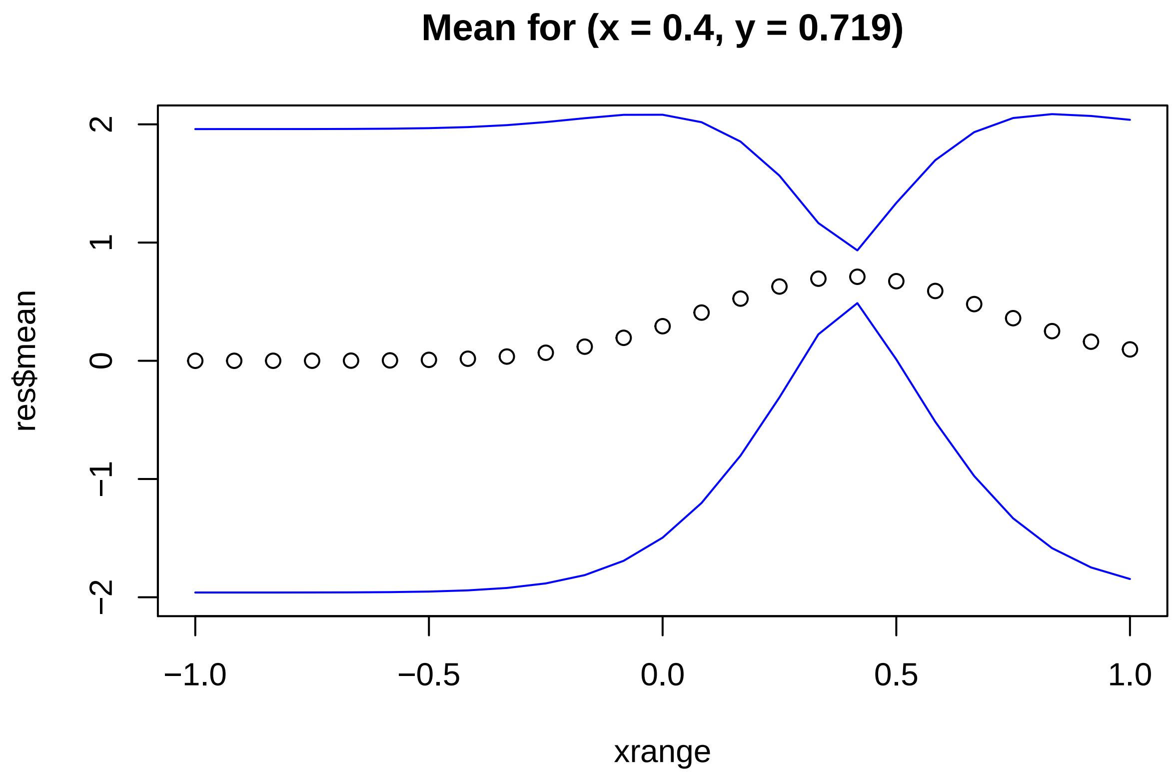
10/18/2020

2.1) (1) Implementing GP Regression

```
# Simulating from posterior distribution of f
posteriorGP = function(X, y, XStar, sigmaNoise, k, sigmaF, l){

  K = k(X,X, sigmaF, l)
  n = length(XStar)
  L = t(chol(K + sigmaNoise*diag(dim(K)[1])))
  kStar = k(X,XStar, sigmaF, l)
  alpha = solve(t(L), solve(L,y))

  FStar = t(kStar) %*% alpha
  v = solve(L, kStar)
  vf = k(XStar, XStar, sigmaF, l) - t(v)%*%v #+ sigmaNoise*diag(n) #Adding sigma for noise
  #print(k(XStar, XStar, sigmaF, l))
  #print(diag(t(v)%*%v))
  logmarglike = -t(y)%*%alpha/2 - sum(diag(L)) - n/2*log(2*pi)


  # Returns a vector with the posterior mean and variance
  return(list("mean" = FStar,"variance" = vf,"logmarglike" = logmarglike))
}
```

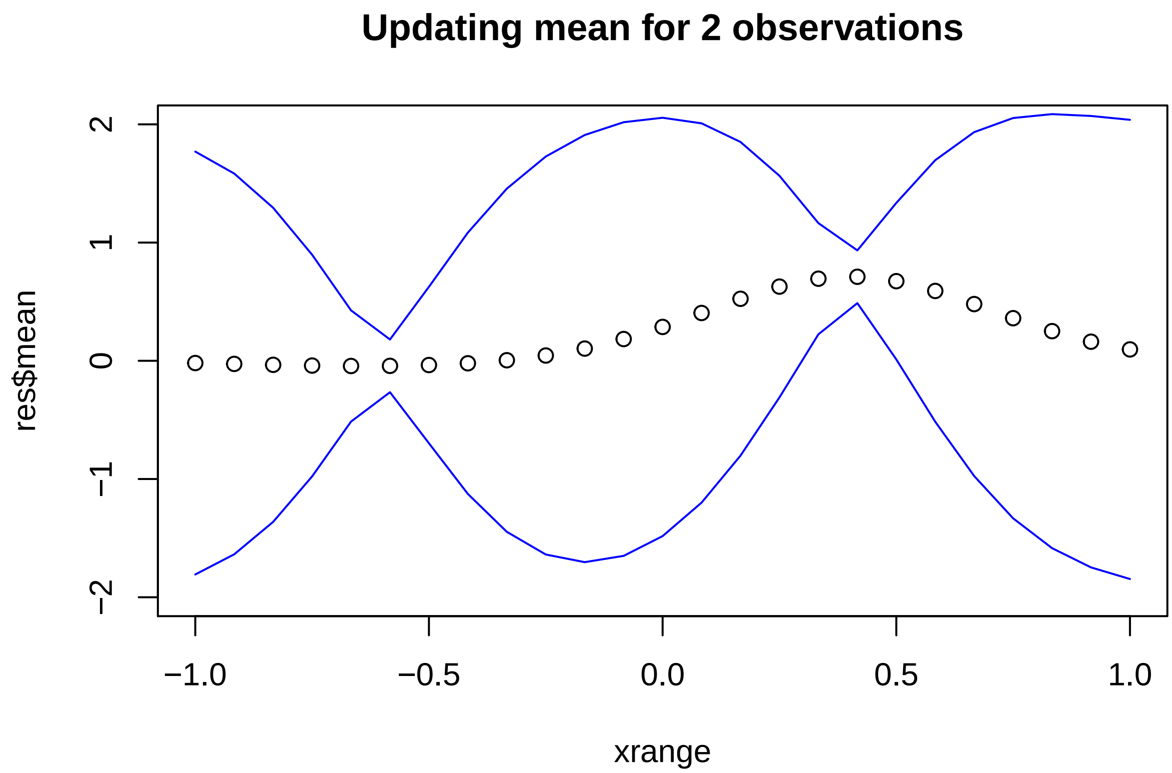
2.1) (2) GP Regression with kernlab

```
sigmaf = 1^2
sigman = 0.1^2
l = 0.3
x = 0.4
y = 0.719
xrange = seq(-1,1, length=25)

res = posteriorGP(x, y, xrange, sigman, SquaredExpKernel, sigmaF = sigmaf, l=l)

# (2) Plotting posterior mean and 95% interval bands
plot(x = xrange, y = res$mean, ylim = c(-2,2), main = "Mean for (x = 0.4, y = 0.719)")
lines(x = xrange, y = res$mean + sqrt(diag(res$variance))*1.96, col="blue", type="l")
lines(x = xrange, y = res$mean - sqrt(diag(res$variance))*1.96, col="blue",  type="l")
```



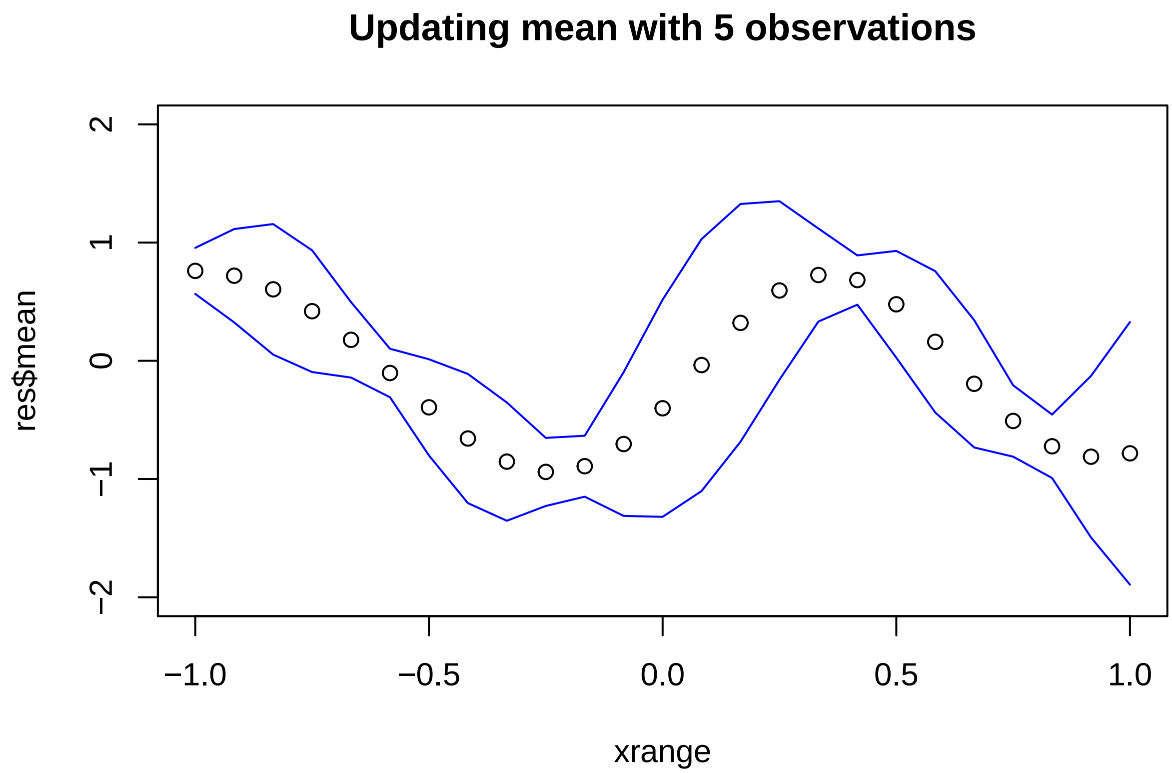
2.1) (3)

```
# (3)
x = c(0.4,-0.6)
y = c(0.719,-0.044)

res = posteriorGP(x, y, xrange, sigman, SquaredExpKernel, sigmaF = sigmaf, l=l)
plot(x = xrange, y = res$mean, ylim = c(-2,2), main = "Updating mean for 2 observations")
lines(x = xrange, y = res$mean + sqrt(diag(res$variance))*1.96, col="blue", type="l")
lines(x = xrange, y = res$mean - sqrt(diag(res$variance))*1.96, col="blue",  type="l")
```



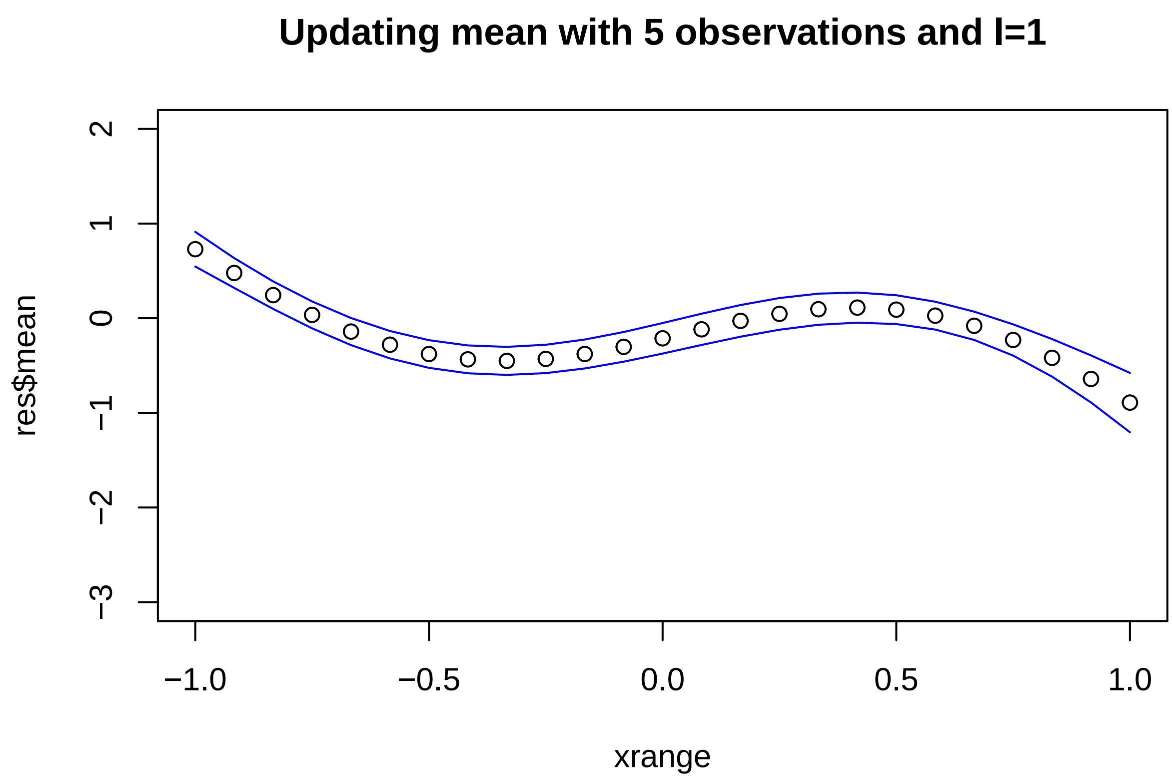
2.1) (4)

```
# (4)
x = c(-1.0, -0.6, -0.2, 0.4, 0.8)
y = c(0.768, -0.044, -0.940, 0.719, -0.664)
sigmaf = 1^2
l = 0.3
res = posteriorGP(x, y, xrange, sigman, SquaredExpKernel, sigmaF = sigmaf, l=l)
plot(x = xrange, y = res$mean, ylim = c(-2,2), main = "Updating mean with 5 observations")
lines(x = xrange, y = res$mean + sqrt(diag(res$variance))*1.96, col="blue", type="l")
lines(x = xrange, y = res$mean - sqrt(diag(res$variance))*1.96, col="blue",  type="l")
```



2.1) (5)

```
# (5)
x = c(-1.0, -0.6, -0.2, 0.4, 0.8)
y = c(0.768, -0.044, -0.940, 0.719, -0.664)
sigmaf = 1
l = 1
res = posteriorGP(x, y, xrange, sigman, SquaredExpKernel, sigmaF = sigmaf, l=l)

plot(x = xrange, y = res$mean, ylim = c(-3,2), main = "Updating mean with 5 observations and l=1")
lines(x = xrange, y = res$mean + sqrt(diag(res$variance))*1.96, col="blue", type="l")
lines(x = xrange, y = res$mean - sqrt(diag(res$variance))*1.96, col="blue",  type="l")
```



Since **l=1**, the produced function will be more smooth compared to (4). We also see that the bands created are much more narrow compared to plot produced in (4).

2.2 (1) GP Regression with kernlab

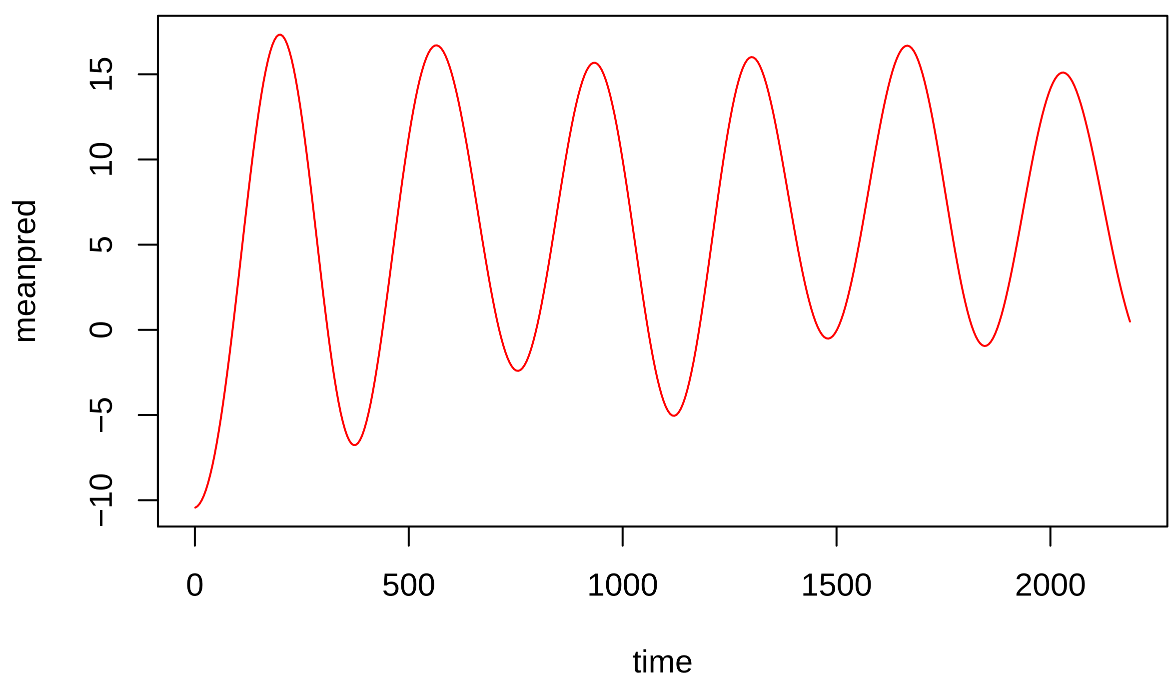
2.2 (1)

```
SEKernel = function(x1, x2, ell = 1, sigmaf = 1){
  r = sqrt(sum((x1 - x2)^2))
  return(sigmaf^2*exp(-r^2/(2*ell^2)))
}

kernel = SEKernel(x1=1, x2=2, ell = 1, sigmaf = sigmaf)

X = c(1,3,4)
XStar = c(2,3,4)
covarmatrix = kernelMatrix(kernel = SEKernel, x=X, y=XStar)

newKernelMatrix = function(kernel, x, y){
  SEKernel = function(x1, x2, ell = 1, sigmaf = 1){
    r = sqrt(sum((x1 - x2)^2))
    return(sigmaf*exp(-r^2/(2*ell^2)))
  }
  return(covarmatrix = kernelMatrix(kernel = SEKernel, x=x, y=y))
}
```

```r
SEKernelfunc = function(ell, sigmaf){

  kernel = function(x1, x2){
  r = sqrt(sum((x1 - x2)^2))
  return(sigmaf^2*exp(-r^2/(2*ell^2)))
  }
  class(kernel) <- "kernel"
  return(kernel)
}
```

##2.2 (2)

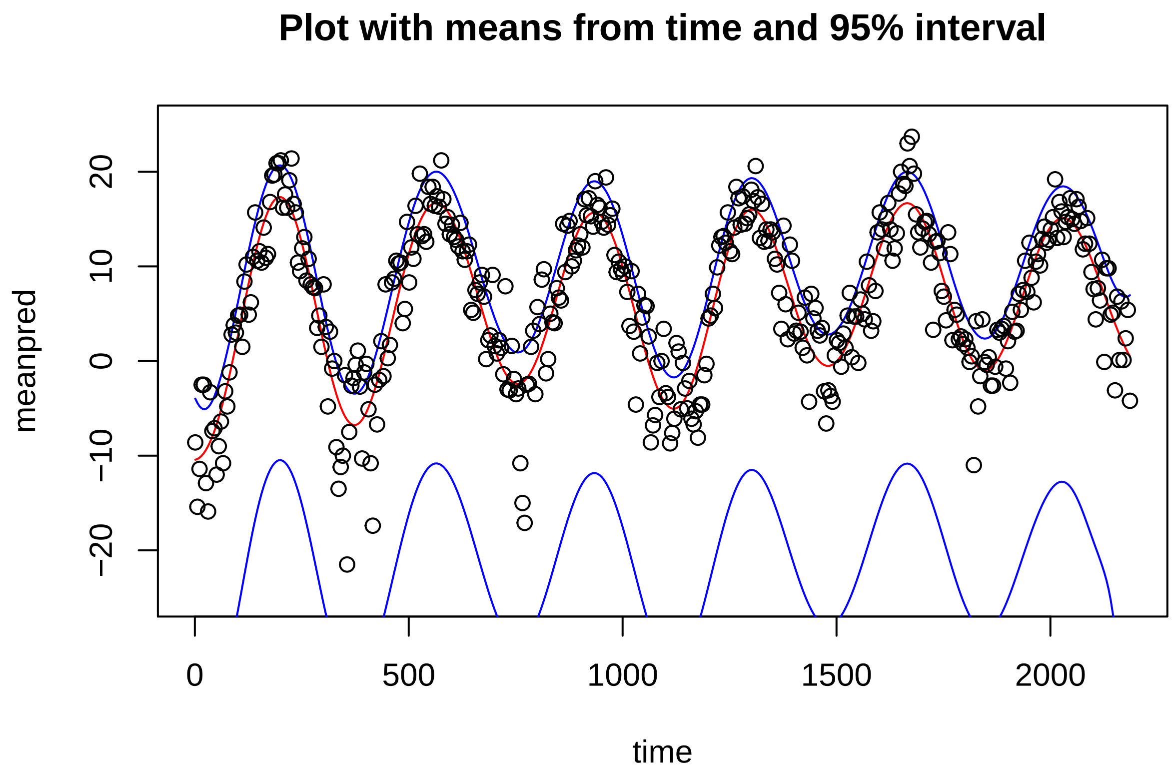
```r
ell = 0.2
sigmaf = 20

# Getting the error term for the GP
polyFit <- lm(temps ~  time + I(time^2))
sigma = sd(polyFit$residuals)
GPfit = gausspr(x=time, y=temps, kernel = SEKernelfunc(ell, sigmaf), var = sigma^2, type="regression")
meanpred = predict(GPfit, time)

# Plotting the means!
plot(time, meanpred, col="red", type='l')
```



A higher value for ell increases smoothness. A higher value for sigmaf increases the possible ranges of the predicted means. A lower value gives lower covariance between prediction. This explains the difference in the highest and lowest values for different sigmaf's.

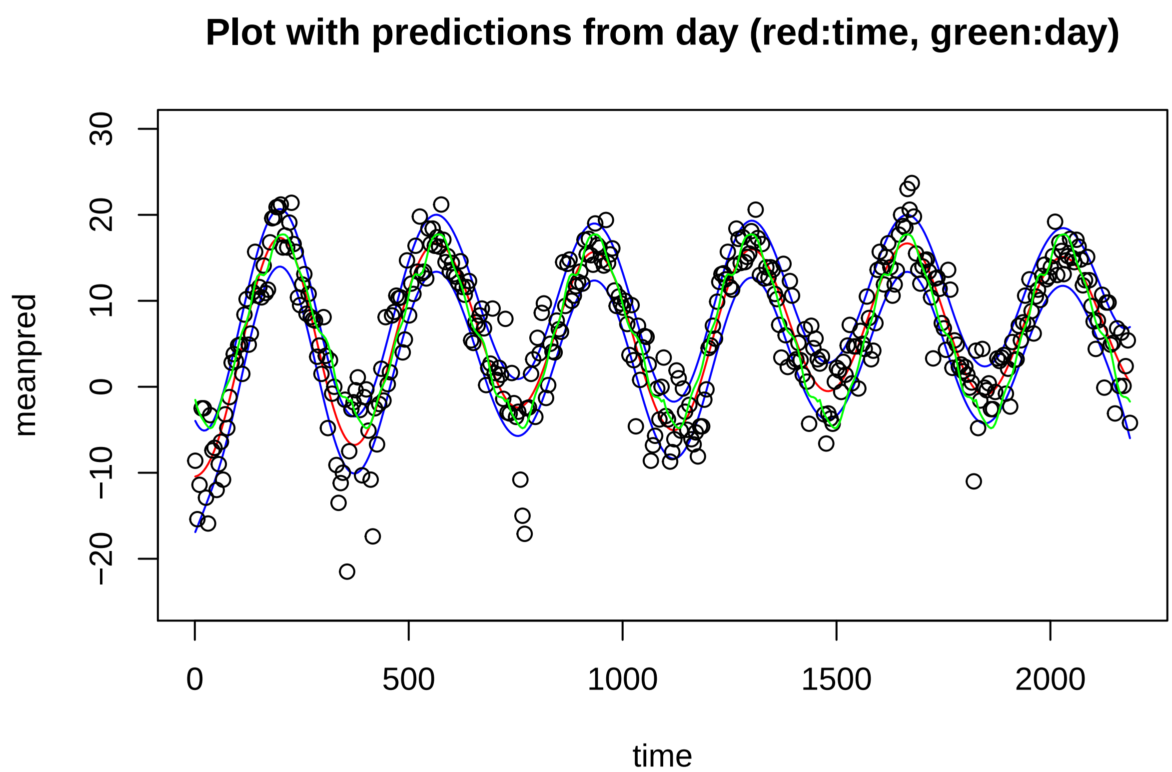
##2.2 (3)

```
# Modified posteriorGP function
posteriorGP = function(X, y, XStar, sigmaNoise, k){
  K = kernelMatrix(k,X,X)
  n = length(XStar)
  L = t(chol(K + sigmaNoise*diag(dim(K)[1])))
  kStar = kernelMatrix(k,X,XStar)
  alpha = solve(t(L), solve(L,y))
  FStar = (t(kStar)) %*% alpha
  v = solve(L, kStar)
  vf = kernelMatrix(k,XStar, XStar) - t(v)%*%v #+ sigmaNoise*diag(n) #Adding sigma for noise
  logmarglike = -t(y)%*%alpha/2 - sum(diag(L)) - n/2*log(2*pi)

  # Returns a vector with the posterior mean and variance
  return(list("mean" = FStar,"variance" = vf,"logmarglike" = logmarglike))
}

var = posteriorGP(X=scale(time), y=scale(temps), XStar=scale(time), sigmaNoise=sigma^2, k=SEKernelfunc(
varr = sqrt(var$variance)*sd(temps)

plot(time, meanpred, col="red", type='l', ylim=c(-25,25), main="Plot with means from time and 95% inter
#lines(x = time, y = meanpred + sqrt(diag(var$variance))*1.96, col="blue", type="l")
#lines(x = time, y = meanpred - sqrt(diag(var$variance))*1.96, col="blue",  type="l")
lines(x = time, y = meanpred + sqrt(diag(var$variance))*1.96, col="blue", type="l")
lines(x = time, y = meanpred - sqrt(diag(var$variance))*1.96*sd(temps), col="blue",  type="l")
lines(x = time, y = temps,  type="p")
```



##2.2 (4)

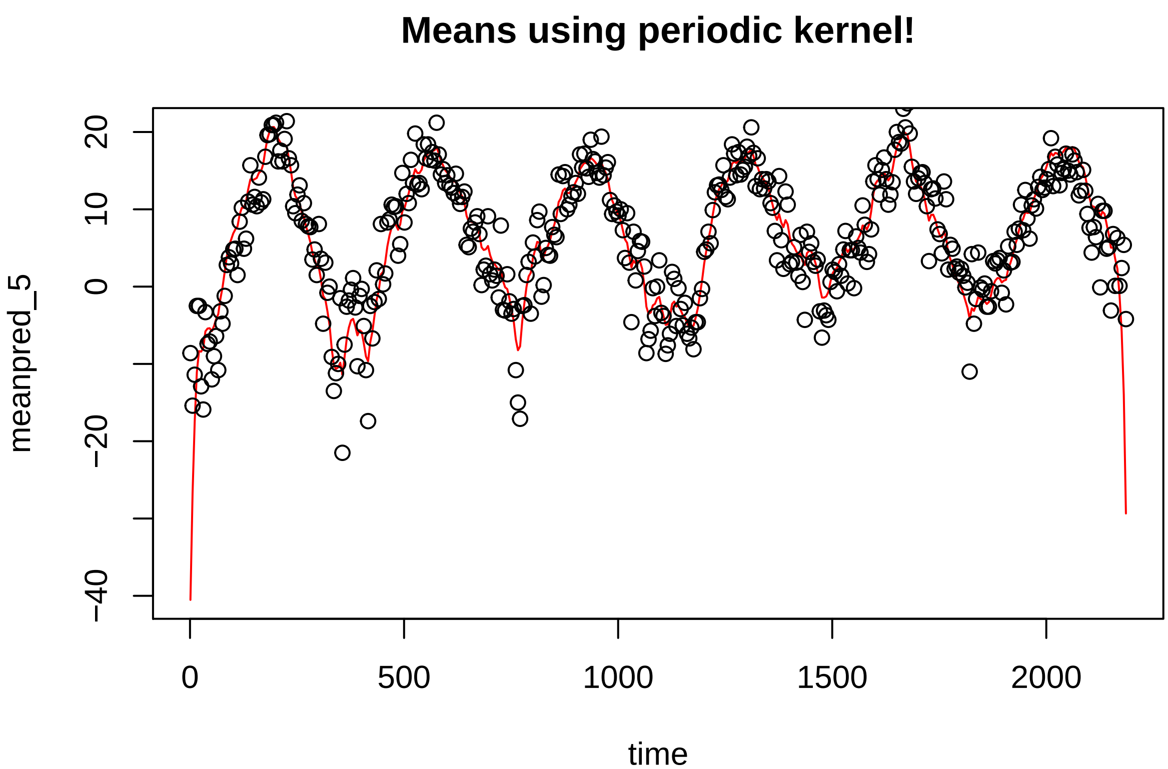
```
ell = 0.2
sigmaf = 20

# Getting the error term for the GP, using day instead of time
polyFit <- lm(temps ~  day + I(day^2))
sigma = sd(polyFit$residuals)
GPfit = gausspr(x=day, y=temps, kernel = SEKernelfunc(ell, sigmaf), var = sigma^2, type="regression")
meanpred_day = predict(GPfit, day)

plot(time, meanpred, col="red", type='l', ylim=c(-25,30), main="Plot with predictions from day (red:time
lines(x = time, y = meanpred + sqrt(diag(var$variance))*1.96, col="blue", type="l")
lines(x = time, y = meanpred - sqrt(diag(var$variance))*1.96, col="blue",  type="l")
lines(x = time, y = temps,  type="p")
lines(x = time, y = meanpred_day, col="green",  type="l")
```



##2.2 (5)

```
periodickernelfunc = function(sigmaf, l1, l2, d){
  periodicKernel = function(x1, x2){
  r = sqrt(sum((x1 - x2)^2))
  one = exp(-2*sin(r/d)/l1^2)
  two = exp(-r^2/(2*l2^2))
  return(sigmaf^2*one*two)
  }
  class(periodicKernel) <- "kernel"
  return(periodicKernel)
}

# Hyperparameters
sigmaf = 20
l1 = 1
l2 = 10
d = 365/sd(time)

# Getting the error term for the GP # USING TIME MAYBE CHANGE
polyFit <- lm(temps ~  time + I(time^2))
sigma = sd(polyFit$residuals)
GPfit = gausspr(x=time, y=temps, kernel = periodickernelfunc(sigmaf, l1, l2, d), var = sigma^2, type="r
meanpred_5 = predict(GPfit, time)

# Plotting the means!
```

```r
plot(x = time, y = meanpred_5, col="red", type='l', main = "Means using periodic kernel!")
lines(x = time, y = temps,  type="p")
```



We can see that the periodic kernel produces

##2.3 (1)

```r
#data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/banknoteFrau
```

Another chunk

```r
# (3)
```

Another chunk

```r
# (3)
```