BNP with Gaussian Processes

Define your Gaussian Process covariance function

```
##Covariance functions
SigmaEK <- function(X1,X2,l=1,scale=1) {
  #exponential kernel
  Sigma <- matrix(rep(0, length(X1)*length(X2)), nrow=length(X1))
  for (i in 1:nrow(Sigma)) {
    for (j in 1:ncol(Sigma)) {
      Sigma[i,j] <-scale* exp(-0.5*(abs(X1[i]-X2[j])/l)^2)
    }
  }
  return(Sigma)
}

SigmaBM <- function(X1,X2,scale=2) {
  #Sigma for random walk
  Sigma <- matrix(rep(0, length(X1)*length(X2)), nrow=length(X1))
  for (i in 1:nrow(Sigma)) {
    for (j in 1:ncol(Sigma)) {
      Sigma[i,j] <-scale*min(X1[i],X2[j])
    }
  }
  return(Sigma)
}
```

This tutorial was adapted from https://www.r-bloggers.com/gaussian-process-regression-with-r/

1. Simulate two realizations of 50 points

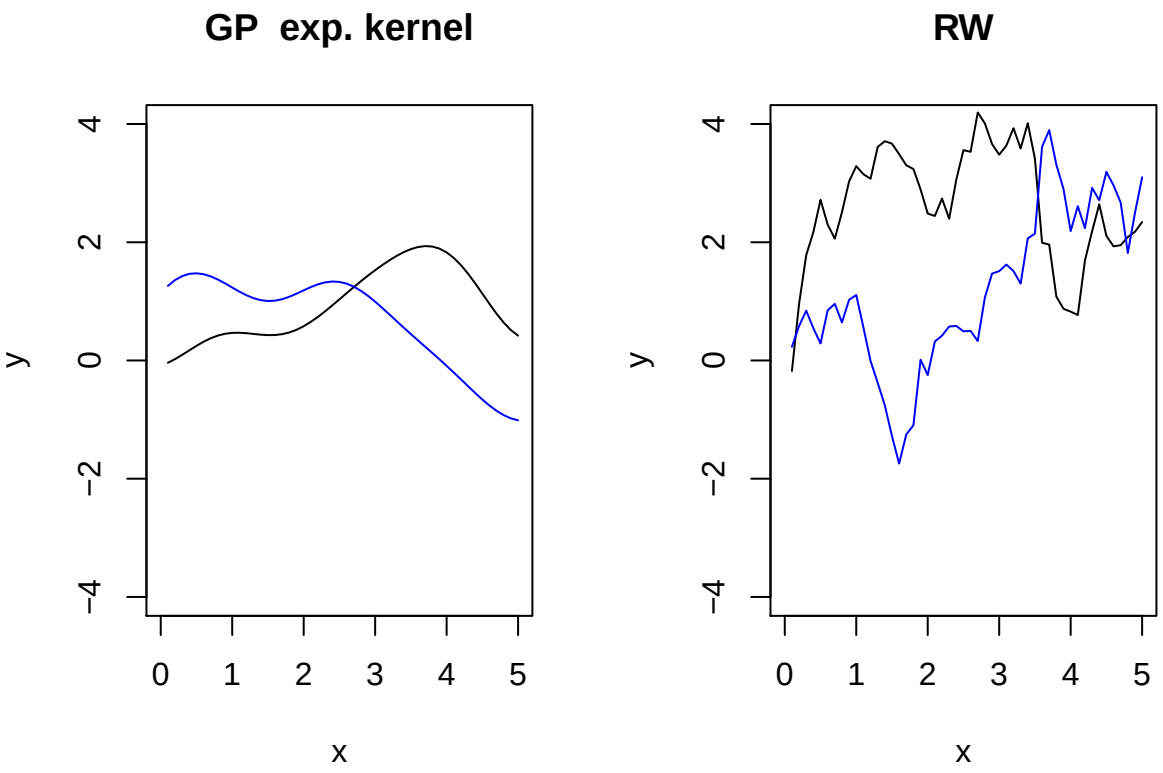
```
# Define the points at which we want to define the functions
x.star <- seq(0.1,5,len=50)

# Calculate the covariance matrix
sigma1 <- SigmaEK(x.star,x.star,1,1)
sigma2<-SigmaBM(x.star,x.star,2)
#chol2<-chol(sigma2)
# Generate a number of functions from the process
n.samples <- 2
values1 <- matrix(rep(0,length(x.star)*n.samples), ncol=n.samples)
values2 <- matrix(rep(0,length(x.star)*n.samples), ncol=n.samples)
for (i in 1:n.samples) {
  # Each column represents a sample from a multivariate normal distribution
  # with zero mean and covariance sigma
  values1[,i] <- mvrnorm(1, rep(0, length(x.star)), sigma1)
  values2[,i] <- mvrnorm(1, rep(0, length(x.star)), sigma2)

}
```

```r
par(mfrow=c(1,2))
plot(x.star,values1[,1],type="l",xlim=c(0,5),ylim=c(-4,4),main="GP  exp. kernel",ylab="y",xlab="x")
#points(x.star,values[,1],xlim=c(-5,5),ylim=c(-4,4),pch="x")
points(x.star,values1[,2],type="l",xlim=c(-5,5),ylim=c(-4,4),col="blue")

plot(x.star,values2[,1],type="l",xlim=c(0,5),ylim=c(-4,4),main="RW",ylab="y",xlab="x")
#points(x.star,values[,1],xlim=c(-5,5),ylim=c(-4,4),pch="x")
points(x.star,values2[,2],type="l",xlim=c(-5,5),ylim=c(-4,4),col="blue")
```



2. Fitting a GP

Now let's assume that we have some known data points

```r
x <- c(-4,-3,-1,0,2)
y <- c(-2,0,1,2,-1)
f <- data.frame(x=x,y=y)
```

Calculate the covariance matrices

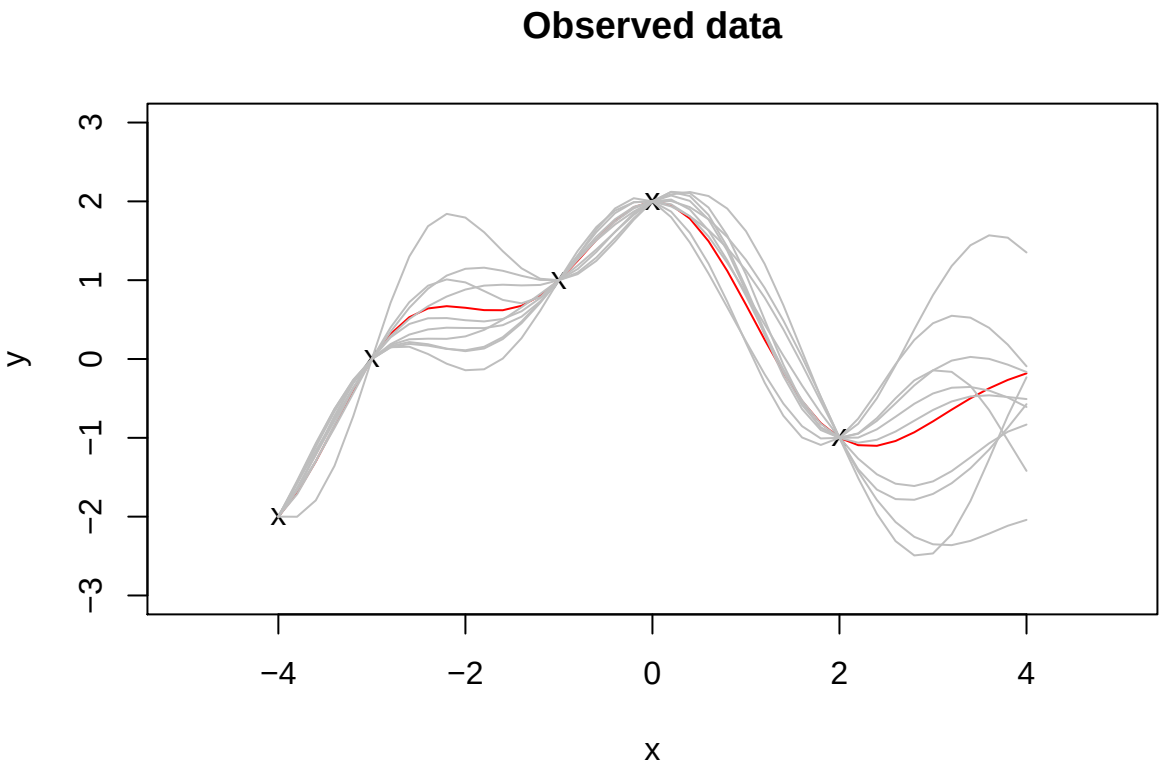
```r
par(mfrow=c(1,1))
plot(x,y,xlim=c(-5,5),ylim=c(-3,3),main="Observed data",pch="x")
x.star<-seq(-4,4,by=.2)
k.xx <- SigmaEK(x,x)
k.xxs <- SigmaEK(x,x.star)
```

```
k.xsx <- SigmaEK(x.star,x)
k.xsxs <- SigmaEK(x.star,x.star)

# These matrix calculations correspond to equation (2.19)
# in the book.
f.star.bar <- k.xsx%*%solve(k.xx)%*%f$y
cov.f.star <- k.xsxs - k.xsx%*%solve(k.xx)%*%k.xxs

points(x.star,f.star.bar,col="red",type="l")
# This time we'll plot more samples.  We could of course
# simply plot a +/- 2 standard deviation confidence interval
# as in the book but I want to show the samples explicitly here.
n.samples <- 10
values <- matrix(rep(0,length(x.star)*n.samples), ncol=n.samples)
for (i in 1:n.samples) {
  values[,i] <- mvrnorm(1, f.star.bar, cov.f.star)
  points(x.star,values[,i],type="l",col="gray")

}
```



3. Now assume that we do not want to overfit and add some noise

```
sigma.n <- 0.1

# Recalculate the mean and covariance functions
f.bar.star <- k.xsx%*%solve(k.xx + sigma.n^2*diag(1, ncol(k.xx)))%*%f$y
```

```
cov.f.star <- k.xsxs - k.xsx%*%solve(k.xx + sigma.n^2*diag(1, ncol(k.xx)))%*%k.xxs
par(mfrow=c(1,1))
plot(x,y,xlim=c(-5,5),ylim=c(-3,3),main="Observed data",pch="x")

# Recalulate the sample functions
values <- matrix(rep(0,length(x.star)*n.samples), ncol=n.samples)
for (i in 1:n.samples) {
  values[,i] <- mvrnorm(1, f.bar.star, cov.f.star)
points(x.star,values[,i],type="l",col="blue")
  }

points(x,y,xlim=c(-5,5),ylim=c(-3,3),pch="x")
```