Untitled

BenLarson

April 29, 2016

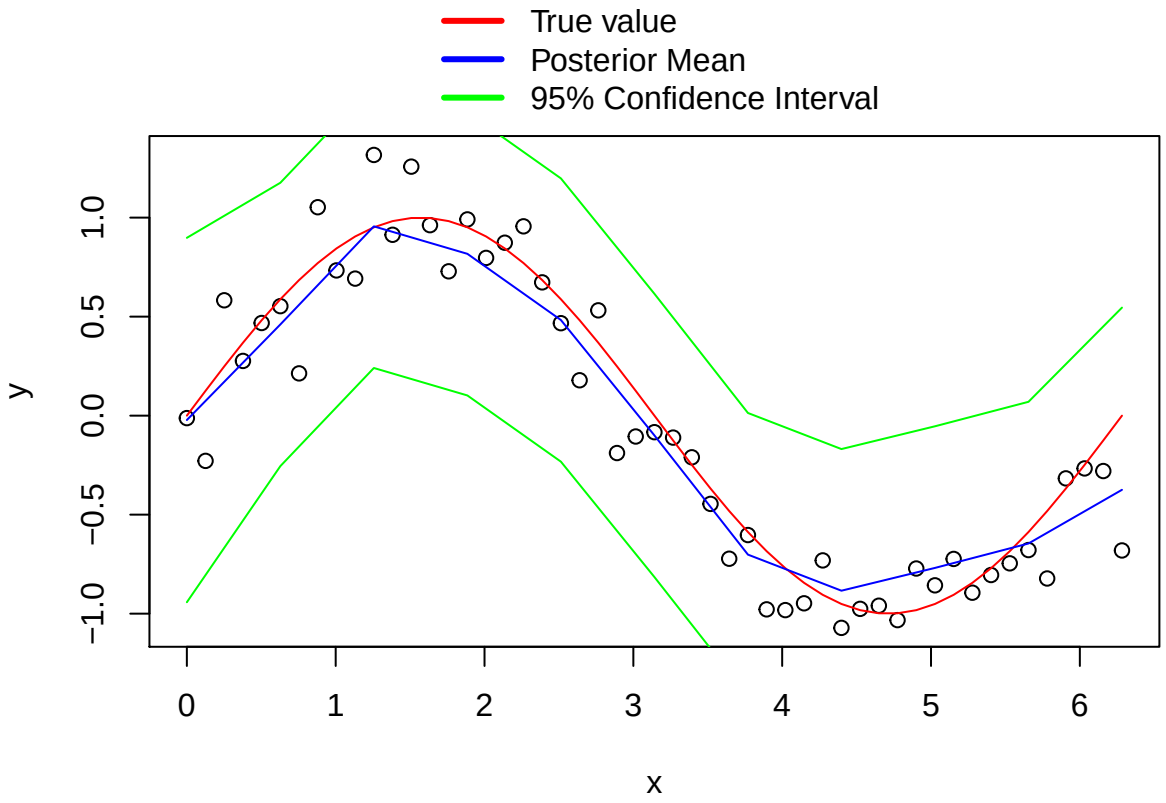
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
k_Lcov = function(x,lam)
{
  cc = matrix(1,nrow=length(x[,1]),ncol=length(x[,1]) )
  x1 = cc%*%x
  x1 = x1/length(x[,1])
  a = x - x1
  a = t(a)%*%a
  a = a/(length(x[,1])-1)
  # cc = exp(-1/(2*lam)*a)
return(a)
}

k_cov = function(x,lam)
{
  # exp(-1/(2*lam)* (x - mean(x) )^2)
  xbar = colMeans(x)
  # xbar = diag(xbar)
  # print(xbar)
  c= matrix(0,nrow=length(x[1,]),ncol=length(x[1,]) )
  for (i in 1:length(x[,1]))
  {
    temp = x[i,]-xbar
    c = c + outer(temp,temp)
  }
  c = c/(length(x[,1])-1)
  return(c)
}
k = function(x,x_,lam)
{
  #t(x) repiclate then x replicate then subtract then square
  n = length(x)
  n_ = length(x_)
  x = replicate(n_,x) #rep not dot product
  x_ = t(replicate(n,x_))
  r=exp(-1/(2*lam)*( x- x_ )^2)
  return(r)
}

dra = function(n, p , nr )
{
  x = matrix(0,n,p)
```

```
  for (i in 1:nr) {
    u = replicate(p, runif(n)) #make x a matrix
    s = cov(u)
    L = chol(s)
    x[i] = norm(L%*%t(L) - s)
    # x[i] = u%*%L
  }
  x
}
dra_prior = function(x, x_,point, lam)
{
  n = point
  p = matrix(0,nrow=point,ncol=n)
  ss = k(x,x_,lam)#this is LAMBDA Maybe change back to k_cov
  diag(ss) = diag(ss) +0.0001
  L = t(chol(ss)) #chol2inv
  d = length(L[1,])
  #n sample size
  #d dimension
  Z = matrix(rnorm(n*d),nrow=d,ncol=n)
  mu = 0# 1:d
  r = mu + L%*%Z
}
dra_samp = function(ss, point)
{
  n = point
  p = matrix(0,nrow=point,ncol=n)
  L = t(chol(ss))
  d = length(L[1,])
  #n sample size
  #d dimension
  Z = matrix(rnorm(n*d),nrow=d,ncol=n)
  mu = 0# 1:d
  r = mu + L%*%Z
}
sigma_noise = 0.5
x = seq(0,2*pi,2*pi/50)
e = rnorm(x,0,sigma_noise^2)
y = sin(x)+e
plot(x,y)
lines(x,sin(x),col='red')
#
# ## KERNEL PART for Sin model
xx = seq(0,2*pi,2*pi/10)
lambda=0.1
ker = k(x,x,lambda)
diag(ker) = diag(ker)+sigma_noise  #   sigma_noise
# k_1 = solve(ker)
k_1 = chol2inv(chol(ker))
kxx_x = k(xx,x,lambda)
## lambda big... singular problem want it to not be all 1s, all 0s. L controls this.

ystar = kxx_x%*%(k_1%*%y)
```

```
kxx_xx = k(xx,xx,lambda)
kx_xx = k(x,xx,lambda)
vstar = kxx_xx - kxx_x %*% (k_1 %*% kx_xx)
lines(xx,ystar,col='blue')
lines(xx,ystar+1.96*sqrt(diag(vstar)),col='green')
lines(xx,ystar-1.96*sqrt(diag(vstar)),col='green')
legend("bottom", inset=c(0.0,1.0),xpd=TRUE,bty='n', c("True value","Posterior Mean", "95% Confidence In
       col = c("red", "blue","green"), lwd=3)
```




```
#
```