On the use of GPspt

Adrien Ickowicz

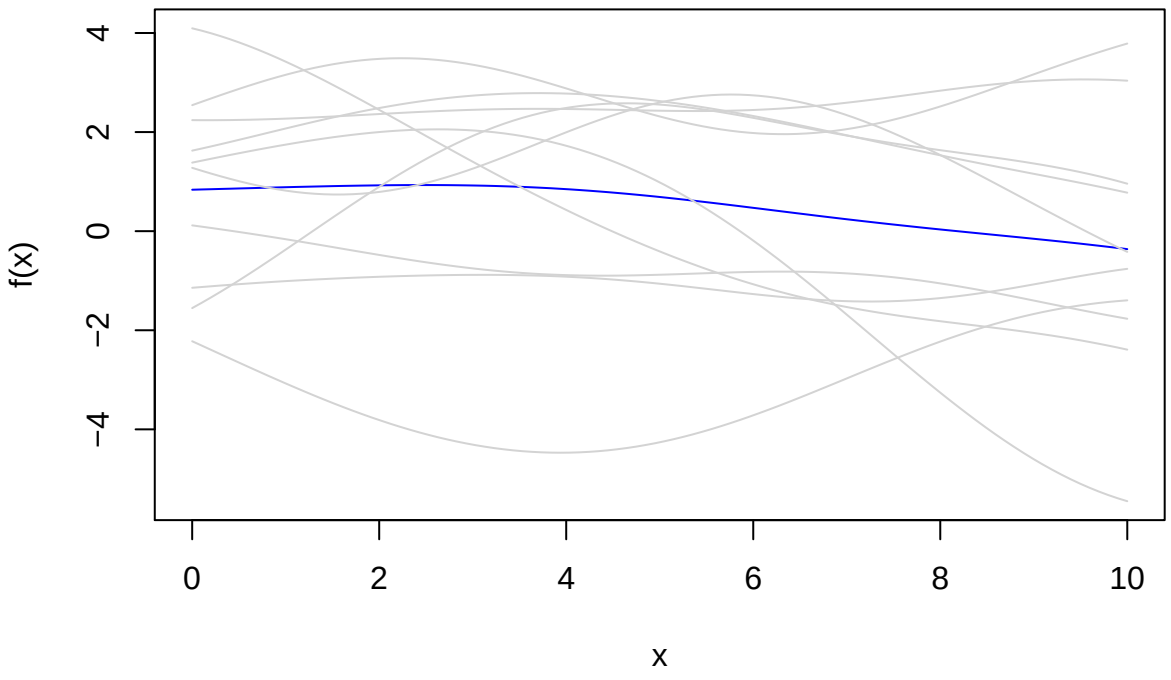
2016-11-04

Introduction to the package

Generating a GP

The very first thing you may want to do, is to generate a GP.

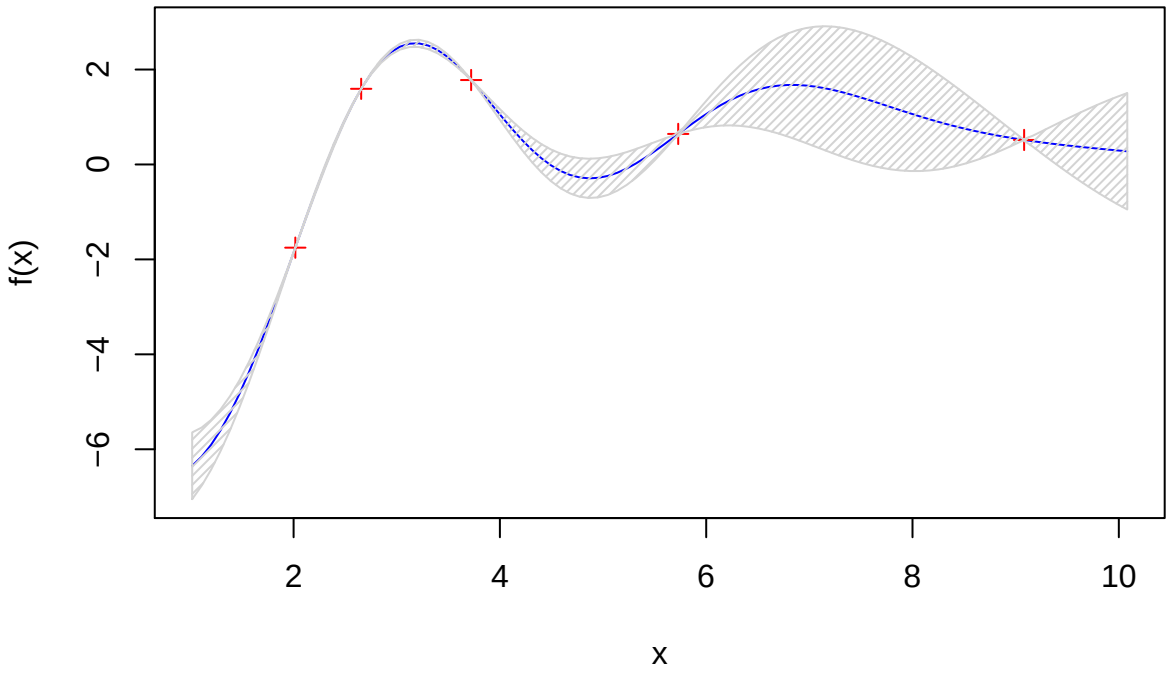
```
library(GPspt, quietly = TRUE)
x = seq(0,10,length.out = 100)
param = list(list(q1=2, q2=5))
nb = 10
kernel = list(k_longterm)
attr(kernel, "type") <- "temporal"
tt = GPgen(x, param, nb, kernel)
plot(tt$x, colMeans(tt$f), type = "l", col = "blue", ylim = range(tt$f),
     xlab = "x", ylab = "f(x)")
for(i in 1:10){
  points(tt$x, tt$f[i,], type= "l", col = "lightgrey")
}
```



Generating a GP with anchoring points

Now let's assume that you observe a number of realisations, and you want to fit a GP regression to them.

```
set.seed(1)
x = list(matrix(runif(5,0,10), ncol = 1))
y = runif(5,-2,2)
xPred = list(matrix(seq(min(x[[1]])-1,max(x[[1]])+1,length.out = 100), ncol = 1))
param = list(list(q1=1, q2=2))
kernel = list(k_longterm)
tt = GPpred(xPred, x,y, param, kernel)
CI =  c(tt$mp -1.96*sqrt(diag(tt$sp)), rev(tt$mp +1.96*sqrt(diag(tt$sp))))
plot(xPred[[1]], tt$mp, type = "l", col = "blue", xlab = "x", ylab = "f(x)", ylim = range(CI))
points(x[[1]],y, col="red", pch = 3)
polygon(c(xPred[[1]], rev(xPred[[1]])),CI, col = "lightgrey", density = 30)
```



Estimating the hyperparameters

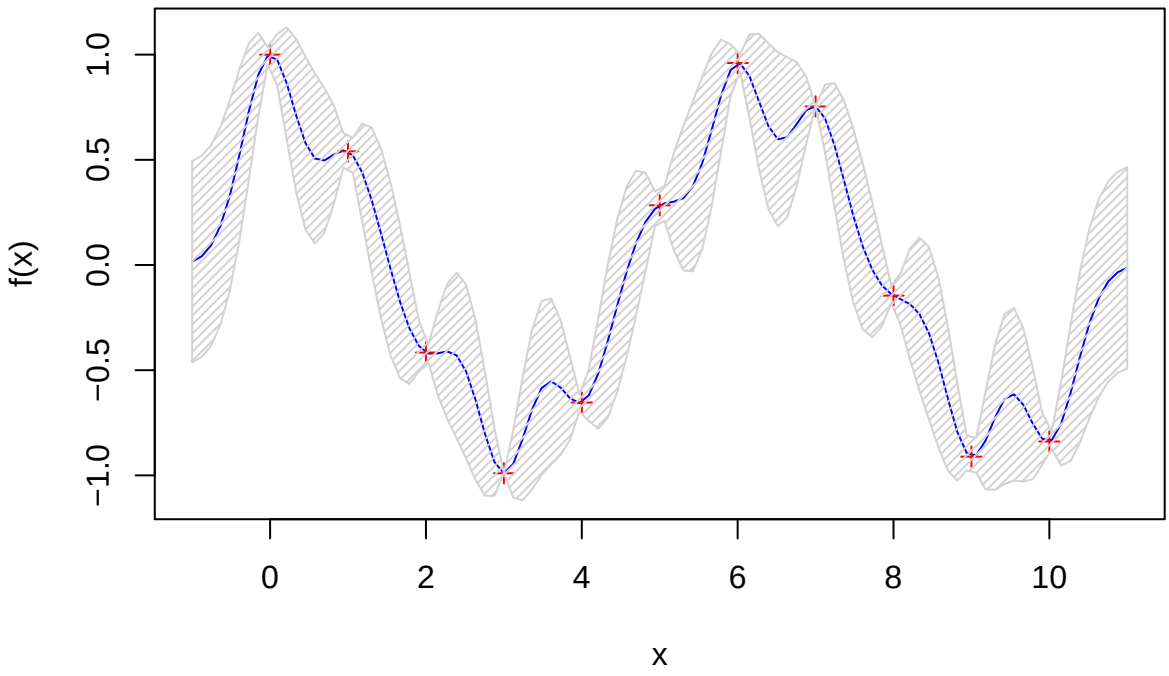
Let's assume we have a set of realisation, but we have no idea what the hyperparameters are.

```
kernelList = list(k_longterm)
attr(kernelList, "name") <- c("long term")
attr(kernelList, "parameters") <- list(c("q1","q2"))
attr(kernelList, "type") <- c("temporal")
thetaInit = c(log(2),log(1))
x  = matrix(seq(0,10,1), ncol = 1)
y = cos(x)

xList = list(x)
res = optifix(thetaInit, fixed = c(FALSE,FALSE),fn = wrap_ll, y=y, kernel = kernelList,
              x=xList,  control = list(fnscale = -1, maxit = 10000), method = "Nelder-Mead")
param = parToList(res$fullpar, kernelList)

xPred = list(matrix(seq(min(x)-1,max(x)+1,length.out = 100), ncol = 1))
```

```
tt = GPpred(xPred, xList,y, param, kernelList)
CI =  c(tt$mp -1.96*sqrt(diag(tt$sp)), rev(tt$mp +1.96*sqrt(diag(tt$sp))))
plot(xPred[[1]], tt$mp, type = "l", col = "blue", xlab = "x", ylab = "f(x)", ylim = range(CI))
points(xList[[1]],y, col="red", pch = 3)
polygon(c(xPred[[1]], rev(xPred[[1]])),CI, col = "lightgrey", density = 30)
```



Advanced features

Spatial model

```
kernelList = list(k_spatial_iso)
attr(kernelList, "name") <- c("spatial")
attr(kernelList, "parameters") <- list(c("s1", "s2"))
attr(kernelList, "type") <- c("spatial")
thetaInit = c(log(1), log(5), log(0.1))
s = expand.grid(seq(1,10,length.out=5), seq(1,10,length.out = 5))
y = exp(-1/25*((s[,1]-5)^2 + (s[,2]-5)^2))
xList = list(s)
res = optifix(thetaInit, fixed = c(FALSE, FALSE,FALSE),fn = wrap_ll, y=y, kernel = kernelList, x=xList,
param = parToList(res$fullpar, kernelList)
#param = parToList(thetaInit, kernelList)
```

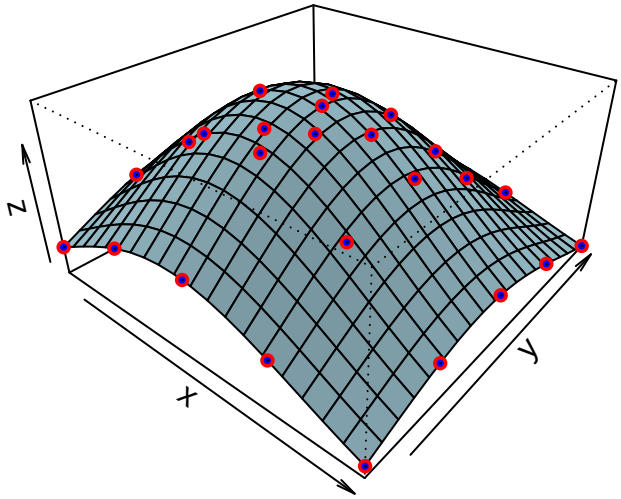
Then we can look at a prediction

```
sPred = expand.grid(seq(1,10,length.out=20), seq(1,10,length.out = 20))

xPred = list(sPred)
tt = GPpred(xPred, xList,y, param, kernelList)
```

```
xPred2 =  list(s)
tt2 = GPpred(xPred2, xList,y, param, kernelList)

p <- persp(x =seq(1,10,length.out=20) , y=seq(1,10,length.out=20) , z=matrix(tt$mp, ncol=20), phi = 30,
obs <- trans3d(xList[[1]][,1], xList[[1]][,2],y,p)
pred <- trans3d(xPred2[[1]][,1], xPred2[[1]][,2],tt2$mp,p)
points(obs, col="red",pch=16)
points(pred, col="blue",pch=16, cex=0.5)
segments(obs$x, obs$y, pred$x, pred$y)
```



Spatio-temporal model

The GPspt package has a set of implemented kernels, for temporal covariance (long term, seasonal, short term) and for for spatial covariance (exponential isotropic, mattern). Kernels can be added to each others, leading to a potential spatio-temporal model. For example,

$$k_{full} = k_{seasonal} + k_{spatial} \tag{1}$$

```
kernelList = list(k_longterm, k_spatial_iso)
attr(kernelList, "name") <- c("temporal", "spatial")
attr(kernelList, "parameters") <- list(c("q1","q2"), c("s1", "s2"))
attr(kernelList, "type") <- c("temporal", "spatial")
thetaInit = c(log(0.54),log(2.17), log(0.28), log(6.70), log(0.5))
t  = matrix(rep(seq(1,6,length.out = 6),25), ncol = 1)
s = expand.grid(seq(1,10,length.out=5), seq(1,10,length.out = 5))
s = s[rep(1:nrow(s),each = 6),]
y = cos(2*pi*t/5) + exp(-1/25*((s[,1]-5)^2 + (s[,2]-5)^2))
xList = list(t,s)
wrap_ll(thetaInit, y = y, kernelList = kernelList, xList = xList)
```

```
##           [,1]
## [1,] -108.0092
```

```
res = optifix(thetaInit, fixed = c(FALSE, TRUE, FALSE, TRUE,FALSE),fn = wrap_ll, y=y, kernel = kernelLis
param = parToList(res$fullpar, kernelList)
```

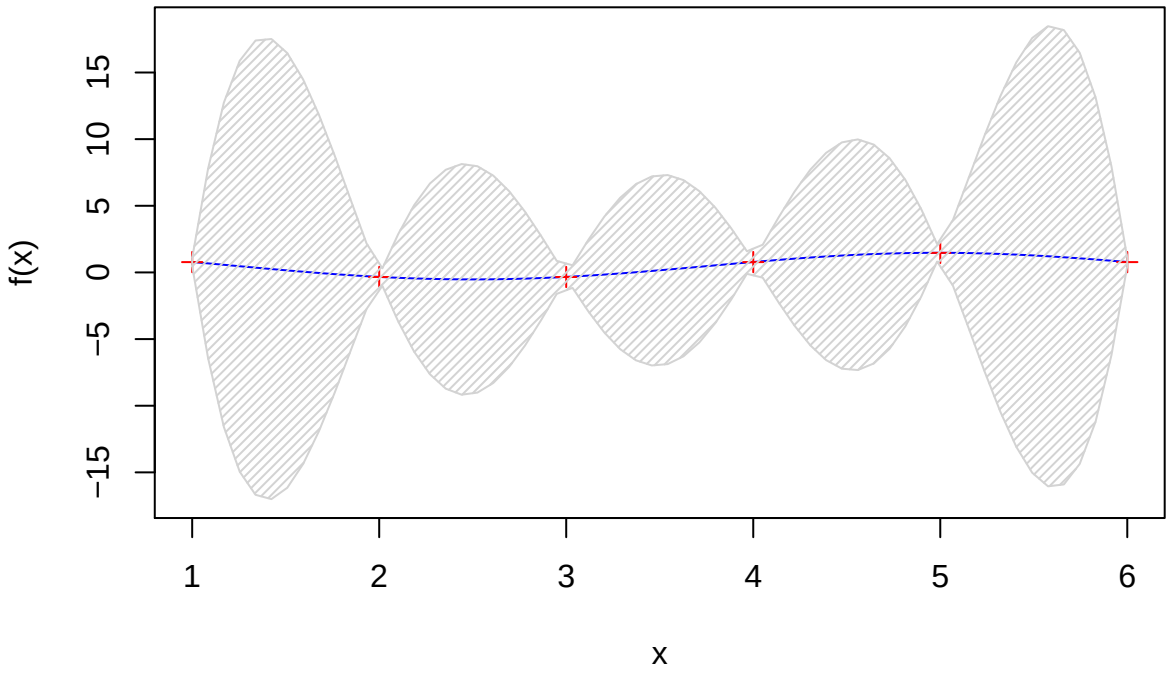
Then we can look at a prediction

```
tPred = matrix(rep(seq(1,6,length.out = 60),400), ncol = 1)
sPred = expand.grid(seq(1,10,length.out=20), seq(1,10,length.out = 20))
sPred = sPred[rep(1:nrow(sPred),each = 60),]

# Plot a temporal prediction at a precise location s = (3.25,1)
ds = (s[,1] - 3.25)^2 +  (s[,2] - 1)^2; dsP = (sPred[,1] - 3.25)^2 +  (sPred[,2] - 1)^2
idx.s = which(ds == ds[which.min(ds)])
idx.sP = which(dsP == dsP[which.min(dsP)])

xPred = list(matrix(tPred[idx.sP],ncol=1), sPred[idx.sP,])

tt = GPpred(xPred, xList,y, param, kernelList)
par(mfrow = c(1,1))
CI =  c(tt$mp -1.96*sqrt(diag(tt$sp)), rev(tt$mp +1.96*sqrt(diag(tt$sp))))
plot(xPred[[1]], tt$mp, type = "l", col = "blue", xlab = "x", ylab = "f(x)", ylim = range(CI))
points(t[idx.s],y[idx.s], col="red", pch = 3)
polygon(c(xPred[[1]], rev(xPred[[1]])),CI, col = "lightgrey", density = 30)
```




```
# Plot a spatial prediction at a particular time t = 3
ts = (t - 3)^2; tsP = (tPred - 3)^2
idx.s = which(ts == ts[which.min(ts)])
idx.sP = which(tsP == tsP[which.min(tsP)])
par(mfrow = c(1,1), mar = c(5,4,4,2))
xPred = list(matrix(tPred[idx.sP],ncol=1), sPred[idx.sP,])
tt = GPpred(xPred, xList,y, param, kernelList)
```

```
xPred2 =  list(matrix(t[idx.s],ncol=1), s[idx.s,])
tt2 = GPpred(xPred2, xList,y, param, kernelList)

p <- persp(x =seq(1,10,length.out=20) , y=seq(1,10,length.out=20) , z=matrix(tt$mp, ncol=20), phi = 30,
obs <- trans3d(xList[[2]][idx.s,1], xList[[2]][idx.s,2],y[idx.s],p)
pred <- trans3d(xPred2[[2]][,1], xPred2[[2]][,2],tt2$mp,p)
points(obs, col="red",pch=16)
points(pred, col="blue",pch=16, cex=0.5)
segments(obs$x, obs$y, pred$x, pred$y)
```