Section 2

PSTAT 220C

4/10/2022

```
x=as.numeric(seq(0.01,1,0.01))
n=length(x)
R0=abs(outer(x,(x),'-'))
##the larger the gamma, the more singular the matrix is
gamma=1
Sigma=4*exp(-(R0/gamma)^{1.9}) ##suppose this is the covariance
L=t(chol(Sigma))
y=rep(1,n)+L%*%rnorm(n)
#y
##since I code the covariance by power exponential kernel, it is like a function
#plot(x,y)

###2. RobustGaSP is surrogate model to approximate nonlinear function
###see the gu.R code for the package paper
library(RobustGaSP)
```

```
## #########

## ##
## ## Robust Gaussian Stochastic Process, RobustGaSP Package

## ## Copyright (C) 2016-2022 Mengyang Gu, Jesus Palomo and James O. Berger

## #########

##
## Attaching package: 'RobustGaSP'

## The following object is masked from 'package:stats':
##
##     simulate
```

```
set.seed(1)
#library(lhs)
#input <- 10*maximinLHS(n=15, k=1)
input=10*seq(0,1,1/14) ##equally spaced
output<-higdon.1.data(input)
model<- rgasp(design = input, response = output)
```

```
## The upper bounds of the range parameters are 672.344
## The initial values of range parameters are 13.44688
## Start of the optimization  1  :
## The number of iterations is  11
##  The value of the  marginal posterior  function is  -2.570526
##  Optimized range parameters are 1.725711
##  Optimized nugget parameter is 0
##  Convergence:  TRUE
## The initial values of range parameters are 0.08888889
## Start of the optimization  2  :
## The number of iterations is  10
##  The value of the  marginal posterior  function is  -2.570526
##  Optimized range parameters are 1.725711
##  Optimized nugget parameter is 0
##  Convergence:  TRUE
```

```
model
```

```
##
## Call:
## rgasp(design = input, response = output)
## Mean parameters:  5.226551e-16
## Variance parameter:  0.6613543
## Range parameters:  1.725711
## Noise parameter:  0
```

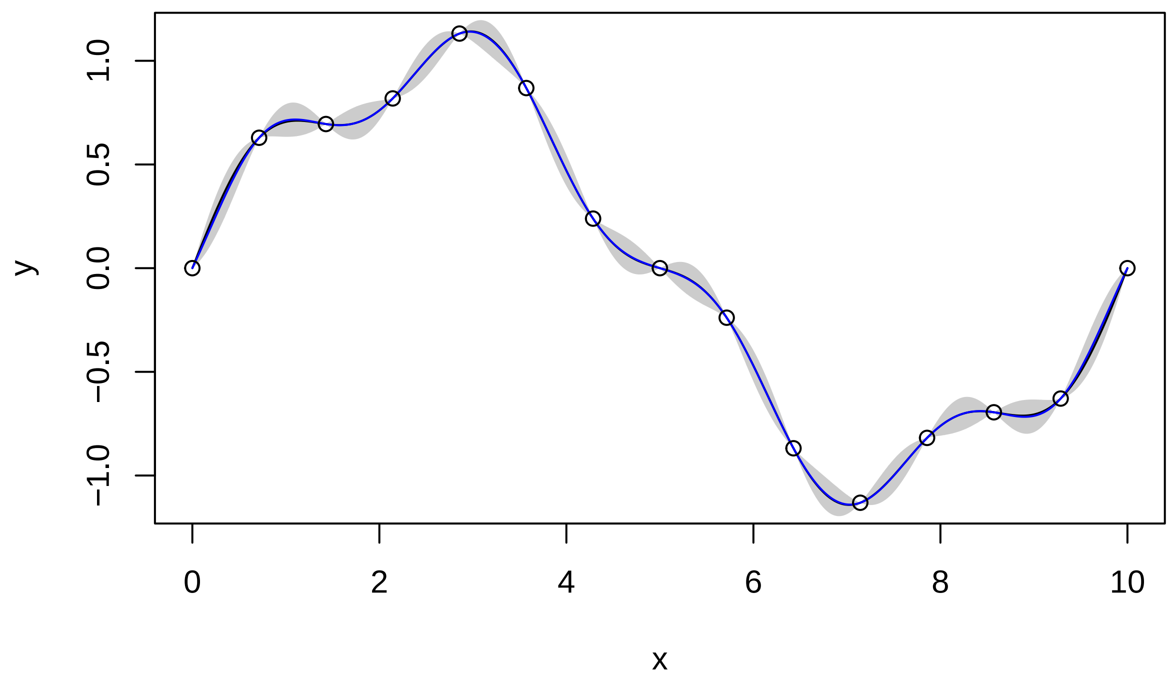
```
#plot(model)
# help(rgasp)
```

- *design*: a matrix of inputs.
- *response*: a matrix of outputs.
- *trend*: the mean/trend matrix of inputs. The default value is a vector of ones.
- *zero.mean*: it has zero mean or not. The default value is NO meaning the mean is not zero.
- *nugget*: numerical value of the nugget variance ratio. If nugget is equal to 0, it means there is either no nugget or the nugget is estimated. If the nugget is not equal to 0, it means a fixed nugget. The default value is 0.
- nugget.est
- *kernel_type*: *matern_3_2*, *matern_5_2*, *pow_exp* (alpha).
- *isotropic*: The default choice is the separable kernel.

```
testing_input = as.matrix(seq(0,10,1/100))
model.predict<-predict(model,testing_input)
names(model.predict)
```

```
## [1] "mean"    "lower95" "upper95" "sd"
```

```
testing_output=higdon.1.data(testing_input)
plot(testing_input,model.predict$mean,type='l',col='blue',
     xlab='x',ylab='y')
polygon( c(testing_input,rev(testing_input)),c(model.predict$lower95,
                                               rev(model.predict$upper95)),col = "grey80", border = F)
lines(testing_input, testing_output)
lines(testing_input,model.predict$mean,type='l',col='blue')
lines(input, output,type='p')
```



Test a 2D function:

```
##test a 2D function
##this is a 2D Branin function from SFU
branin <- function(xx, a=1, b=5.1/(4*pi^2), c=5/pi, r=6, s=10, t=1/(8*pi))
{
  x1 <- xx[1]
  x2 <- xx[2]

  term1 <- a * (x2 - b*x1^2 + c*x1 - r)^2
  term2 <- s*(1-t)*cos(x1)

  y <- term1 + term2 + s
  return(y)
}

##lattice
```

```
library(lhs)
N=36
input_ori=maximinLHS(n=36, k=2) ##this is from a maximin latin hypercube
input=input_ori
# n=sqrt(N)
# input_ori=matrix(NA,N,2)
# input=matrix(NA,N,2)
# input_ori[,1]=rep(as.numeric(seq(0,1,1/(n-1))),n)
# input_ori[,2]=as.vector(t(matrix(as.numeric(seq(0,1,1/(n-1))),n,n )))
input[,1]=-5+input_ori[,1]*20
input[,2]=0+input_ori[,2]*15

num_obs=dim(input)[1]
p=dim(input)[2]
output=matrix(0,num_obs,1)
for(j in 1:num_obs){
  output[j]=branin(input[j,])
}

model<- rgasp(design = input, response = output)
```

```
## The upper bounds of the range parameters are 836.6575 632.0497
## The initial values of range parameters are 16.73315 12.64099
## Start of the optimization  1  :
## The number of iterations is  29
##  The value of the  marginal posterior  function is  -131.115
##  Optimized range parameters are 42.6368 139.4829
##  Optimized nugget parameter is 0
##  Convergence:  FALSE
## The initial values of range parameters are 2.160851 1.632407
## Start of the optimization  2  :
## The number of iterations is  16
##  The value of the  marginal posterior  function is  -131.115
##  Optimized range parameters are 42.63612 139.4821
##  Optimized nugget parameter is 0
##  Convergence:  TRUE
```

```
library(fields)
```

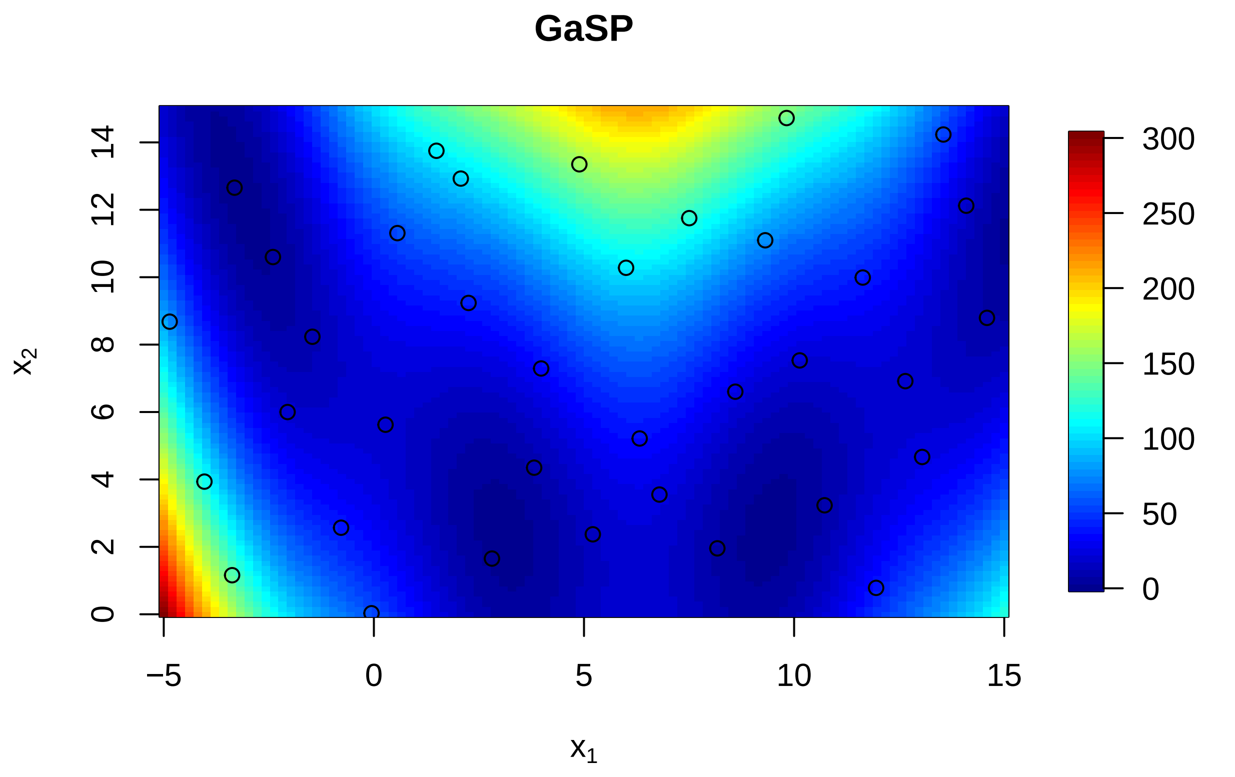
```
## Loading required package: spam
```

```
## Warning: package 'spam' was built under R version 4.1.2
```

```
## Spam version 2.8-0 (2022-01-05) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: viridis

## Loading required package: viridisLite

##
## Try help(fields) to get started.
```

```
num_testing=10000
grid.list=list(x=seq(0,1,1/99), y=seq(0,1,1/99))## use field package
testing_input_ori=make.surface.grid(grid.list)

testing_input=testing_input_ori
testing_input[,1]=-5+testing_input_ori[,1]*20
testing_input[,2]=0+testing_input_ori[,2]*15

model.predict<-predict(model,testing_input)
quilt.plot(x=(testing_input[,1]), y=(testing_input[,2]), z=model.predict$mean,
           nrow = 100, ncol = 100,main='GaSP',xlab=expression(x[1]), ylab=expression(x[2]))
lines(input[,1], input[,2],type='p',pch=1)
```



```
testing_output=matrix(0,num_testing,1)
for(j in 1:num_testing){
  testing_output[j]=branin(testing_input[j,])
}

sqrt(mean((model.predict$mean-testing_output)^2))/sd(testing_output)
```

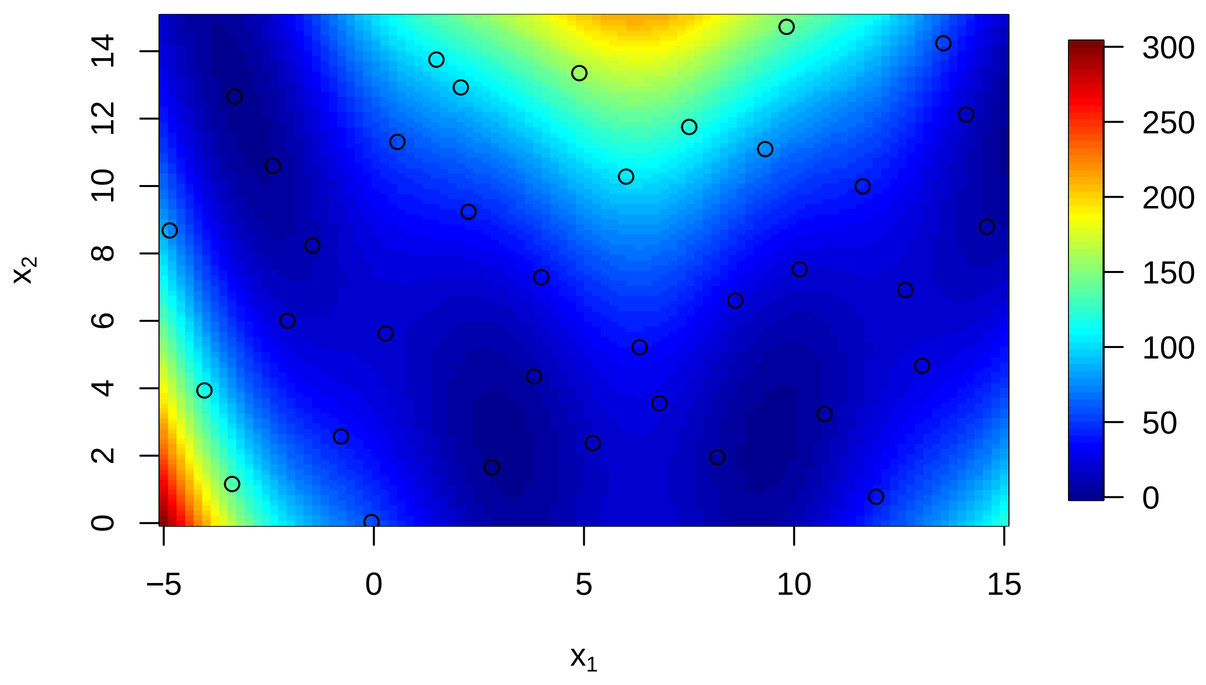
```
## [1] 0.01345911
```

```
##see wjat they are if we have noise in the data
output=matrix(0,num_obs,1)
for(j in 1:num_obs){
  output[j]=branin(input[j,])+rnorm(1,mean=0,sd=1)
}

##nugget
model_tilde<- rgasp(design = input, response = output,nugget.est=T)
```

```
## The upper bounds of the range parameters are 836.6575 632.0497 Inf
## The initial values of range parameters are 16.73315 12.64099
## Start of the optimization  1  :
## The number of iterations is  31
##  The value of the  marginal posterior  function is  -141.7305
##  Optimized range parameters are 25.95761 85.97853
##  Optimized nugget parameter is 1.289297e-07
##  Convergence:  FALSE
## The initial values of range parameters are 2.160851 1.632407
## Start of the optimization  2  :
## The number of iterations is  28
##  The value of the  marginal posterior  function is  -141.7305
##  Optimized range parameters are 25.95778 85.97914
##  Optimized nugget parameter is 1.289261e-07
##  Convergence:  TRUE
```

```
model_tilde.predict<-predict(model_tilde,testing_input)
quilt.plot(x=(testing_input[,1]), y=(testing_input[,2]), z=model.predict$mean,
           nrow = 100, ncol = 100,main='GaSP with noisy observations',xlab=expression(x[1]),
lines(input[,1], input[,2],type='p',pch=1)
```

GaSP with noisy observations



```
##RMSE
sqrt(mean((model_tilde.predict$mean-testing_output)^2))/sd(testing_output)
```

```
## [1] 0.04797479
```