Package ‘RobustGaSP’

June 5, 2019

Type Package
Title Robust Gaussian Stochastic Process Emulation
Version 0.5.7
Date/Publication 2019-06-05 18:20:03 UTC
Maintainer Mengyang Gu <mgu6@jhu.edu>
Author Mengyang Gu [aut, cre],
Jesus Palomo [aut],
James Berger [aut]
License GPL-2 | GPL-3
LazyData true
Depends R (>= 3.5.0), methods
Imports Rcpp (>= 0.12.3), nloptr (>= 1.0.4)
LinkingTo Rcpp, RcppEigen
NeedsCompilation yes
Repository CRAN
RoxygenNote 5.0.1

R topics documented:

RobustGaSP-package ......................................................... 2
findInertInputs .............................................................. 5
humanity_model ............................................................. 7
leave_one_out_rgasp ...................................................... 8
plot ................................................................. 9
ppgasp ............................................................ 10
# Description


# Details

The DESCRIPTION file:

<table>
<thead>
<tr>
<th>Package:</th>
<th>RobustGaSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Title:</td>
<td>Robust Gaussian Stochastic Process Emulation</td>
</tr>
<tr>
<td>Version:</td>
<td>0.5.7</td>
</tr>
<tr>
<td>Date/Publication:</td>
<td>2019-06-05 08:10:03 UTC</td>
</tr>
<tr>
<td>Authors@R:</td>
<td>c(person(given=&quot;Mengyang&quot;,family=&quot;Gu&quot;,role=c(&quot;aut&quot;,&quot;cre&quot;),email=&quot;<a href="mailto:mgu6@jhu.edu">mgu6@jhu.edu</a>&quot;), person(given=&quot;Jesus&quot;,family=&quot;Palomo&quot;,role=&quot;aut&quot;,email=&quot;<a href="mailto:jesus.palomo@urjc.es">jesus.palomo@urjc.es</a>&quot;), person(given=&quot;James&quot;,family=&quot;Berger&quot;,role=&quot;aut&quot;))</td>
</tr>
<tr>
<td>Maintainer:</td>
<td>Mengyang Gu <a href="mailto:mgu6@jhu.edu">mgu6@jhu.edu</a></td>
</tr>
<tr>
<td>Author:</td>
<td>Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]</td>
</tr>
<tr>
<td>License:</td>
<td>GPL-2</td>
</tr>
<tr>
<td>LazyData:</td>
<td>true</td>
</tr>
<tr>
<td>Depends:</td>
<td>R (&gt;= 3.5.0), methods</td>
</tr>
<tr>
<td>Imports:</td>
<td>Rcpp (&gt;= 0.12.3), nloptr (&gt;= 1.0.4)</td>
</tr>
<tr>
<td>LinkingTo:</td>
<td>Rcpp, RcppEigen</td>
</tr>
<tr>
<td>NeedsCompilation:</td>
<td>yes</td>
</tr>
<tr>
<td>Repository:</td>
<td>CRAN</td>
</tr>
<tr>
<td>Packaged:</td>
<td>2019-06-05 02:09:17 UTC; gumengyang</td>
</tr>
<tr>
<td>RxygeneNote:</td>
<td>5.0.1</td>
</tr>
</tbody>
</table>
Index of help topics:

RobustGaSP-package  Robust Gaussian Stochastic Process Emulation
findInertInputs  find inert inputs with the posterior mode
humanity.X  data from the humanity model
leave_one_out_rgasp  leave-one-out fitted values and standard deviation for robust GaSP model
plot  Plot for Robust GaSP model
ppgasp  Setting up the parallel partial GaSP model
ppgasp-class  PP GaSP class
predict.ppgasp  Prediction for PP GaSP model
predict.rgasp  Prediction for Robust GaSP model
predppgasp-class  Predicted PP GaSP class
predrgasp-class  Predictive robust GaSP class
rgasp  Setting up the robust GaSP model
rgasp-class  Robust GaSP class
show  Show Robust GaSP object
show.ppgasp  Show parallel partial Gaussian stochastic process (PP GaSP) object
simulate  Sample for Robust GaSP model

Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]
Maintainer: Mengyang Gu <mgu6@jhu.edu>

References


See Also

RobustGaSP
Examples

```r
#--------------------------
# a 3 dimensional example
#--------------------------
# dimensional of the inputs
dim_inputs <- 3
# number of the inputs
num_obs <- 30
# uniform samples of design
input <- matrix(runif(num_obs*dim_inputs), num_obs, dim_inputs)

# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
# input <- maximinLHS(n=num_obs, k=dim_inputs) #maximin lhd sample

###
# outputs from the 3 dim dettepepel.3.data function

output = matrix(0, num_obs, 1)
for(i in 1:num_obs){
  output[i]<-dettepepel.3.data(input[i,])
}

# use constant mean basis, with no constraint on optimization
m1<- rgasp(design = input, response = output, lower_bound=FALSE)

# the following use constraints on optimization
# m1<- rgasp(design = input, response = output, lower_bound=TRUE)

# the following use a single start on optimization
# m1<- rgasp(design = input, response = output, lower_bound=FALSE)

# number of points to be predicted
num_testing_input <- 5000
# generate points to be predicted
testing_input <- matrix(runif(num_testing_input*dim_inputs), num_testing_input, dim_inputs)
# Perform prediction
m1.predict<-predict(m1, testing_input, outasS3 = FALSE)
# Predictive mean
m1.predict@mean

# The following tests how good the prediction is
testing_output <- matrix(0, num_testing_input, 1)
for(i in 1:num_testing_input){
  testing_output[i]<-dettepepel.3.data(testing_input[i,])
}

# compute the MSE, average coverage and average length
# out of sample MSE
MSE_emulator <- sum((m1.predict@mean-testing_output)^2)/(num_testing_input)

# proportion covered by 95% posterior predictive credible interval
```
\begin{verbatim}
prop_emulator <- length(which((m1.predict@lower95<=testing_output)
 & (m1.predict@upper95>=testing_output))/num_testing_input)

# average length of posterior predictive credible interval
length_emulator <- sum(m1.predict@upper95-m1.predict@lower95)/num_testing_input

# output of prediction
MSE_emulator
prop_emulator
length_emulator
# normalized RMSE
sqrt(MSE_emulator/mean((testing_output-mean(output))^2 ))
\end{verbatim}

\section*{findInertInputs \hspace{1cm} find inert inputs with the posterior mode}

\subsection*{Description}

The function tests for inert inputs (inputs that barely affect the outputs) using the posterior mode.

\subsection*{Usage}

\texttt{findInertInputs(object,threshold=0.1)}

\subsection*{Arguments}

- \texttt{object} \hspace{1cm} an object of class \texttt{rgasp} or the \texttt{ppgasp}.
- \texttt{threshold} \hspace{1cm} a threshold between 0 to 1. If the normalized inverse parameter of an input is smaller this value, it is classified as inert inputs.

\subsection*{Details}

This function utilizes the following quantity
\texttt{object@p*object@beta_hat*object@CL/sum(object@beta_hat*object@CL)}

for each input to identify the inert outputs. The average estimated normalized inverse parameters will be 1. If the estimated normalized inverse range parameters of an input is close to 0, it means this input might be an inert input.

In this method, a prior that has shrinkage effects is suggested to use, e.g. the jointly robust prior (i.e. one should set \texttt{prior_choice=’ref_approx’} in \texttt{rgasp()} to obtain the use codegasp object before using this function). Moreover, one may not add a lower bound of the range parameters to perform this method (i.e. one should set \texttt{lower_bound=F} in \texttt{rgasp()}). For more details see Chapter 4 in Mengyang Gu. (2016). Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output. Ph.D. thesis. Duke University.

\subsection*{Value}

A vector that has the same dimension of the number of inputs indicating how likely the inputs are inerts. The average value is 1. When a value is very close to zero, it tends to be an inert inputs.
Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]
Maintainer: Mengyang Gu <mgu6@jhu.edu>

References


Examples

```r
# test for inert inputs in the Borehole function
# dimensional of the inputs
dim_inputs <- 8
# number of the inputs
num_obs <- 40

# uniform samples of design
set.seed(0)
input <- matrix(runif(num_obs*dim_inputs), num_obs, dim_inputs)
# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
# input <- maximin lhs(n=num_obs, k=dim_inputs) # maximin lhd sample

# rescale the design to the domain
input[,1]<-0.05+(0.15-0.05)*input[,1];
input[,2]<-100+(50000-100)*input[,2];
input[,3]<-63070+(115600-63070)*input[,3];
input[,4]<-900+(1110-900)*input[,4];
input[,5]<-63.1+(116-63.1)*input[,5];
input[,6]<-700+(820-700)*input[,6];
input[,7]<-1120+(1680-1120)*input[,7];
input[,8]<-9855+(12045-9855)*input[,8];

# outputs from the 8 dim Borehole function
output=matrix(0,num_obs,1)
for(i in 1:num_obs){
  output[i]=borehole(input[i,])
}

# use constant mean basis with trend, with no constraint on optimization
m3<- rgasp(design = input, response = output, lower_bound=FALSE)
P=findInertInputs(m3)
```
**humanity_model**

**data from the humanity model**

---

**Description**

This data set provides the training data and testing data from the 'diplomatic and military operations in a non-warfighting domain' (DIAMOND) simulator. It produces the number of casualties during the second day to sixth day after the earthquake and volcanic eruption in Giarre and Catania. See (Overstall and Woods (2016)) for details.

**Usage**

```r
data(humanity_model)
```

**Format**

Four data frame with observations on the following variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>humanity$X$</td>
<td>A matrix of the training inputs.</td>
</tr>
<tr>
<td>humanity$Y$</td>
<td>A matrix of the output of the casualties from the second to sixth day after the earthquake and volcanic eruption for each set of training inputs.</td>
</tr>
<tr>
<td>humanity$Xt$</td>
<td>A matrix of the test inputs.</td>
</tr>
<tr>
<td>humanity$Yt$</td>
<td>A matrix of the test output of the casualties.</td>
</tr>
</tbody>
</table>

**References**


Description

A function to calculate leave-one-out fitted values and the standard deviation of the prediction on robust GaSP models after the robust GaSP model has been constructed.

Usage

leave_one_out_rgasp(object)

Arguments

object an object of class rgasp.

Value

A list of 2 elements with

mean leave one out fitted values.

sd standard deviation of each prediction.

Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]
Maintainer: Mengyang Gu <mgu6@jhu.edu>

References


See Also

rgasp

Examples

library(RobustGaSP)
# ---------------
# a 3 dimensional example
# ---------------
# dimensional of the inputs
dim_inputs <- 3
# number of the inputs
num_obs <- 30
# uniform samples of design
plot

Description

Function to make plots on Robust GaSP models after the Robust GaSP model has been constructed.

Usage

```
## S4 method for signature 'rgasp'
plot(x, y, ...)
```

Arguments

- **x**: an object of class `rgasp`.
- **y**: not used.
- **...**: additional arguments not implemented yet.

Value

Three plots: the leave-one-out fitted values versus exact values, standardized residuals and QQ plot.
Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]

Maintainer: Mengyang Gu <mgu6@jhu.edu>

References


Examples

library(RobustGaSP)
#-------------
# a 3 dimensional example
#-------------
# dimensional of the inputs
dim_inputs <- 3
# number of the inputs
num_obs <- 30
# uniform samples of design
input <- matrix(runif(num_obs*dim_inputs), nrow=num_obs, ncol=dim_inputs)

# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
# input <- maximinLHS(n=num_obs, k=dim_inputs) ##maximin lhd sample

# outputs from the 3 dim dettepepel.3.data function
output = matrix(0, nrow=num_obs, ncol=1)
for(i in 1:nrow(input)){
  output[i] <- dettepepel.3.data(input[i,])
}

# use constant mean basis, with no constraint on optimization
m1 <- rgasp(design = input, response = output, lower_bound=FALSE)

# plot
plot(m1)

------

ppgasp  Setting up the parallel partial GaSP model

Description

Setting up the parallel partial GaSP model for estimating the parameters (if the parameters are not given).
Usage

ppgasp.design.response,trend=matrix(1,dim(response)[1],1),zero.mean="No",nugget=0, nugget.est=F,range.par=NA,prior_choice='ref_approx',a=0.2, b=1/(length(response))^{(1/dim(as.matrix(response)))[2]}*(a+dim(as.matrix(response))[2]), kernel_type='matern_5_2', alpha=rep(1.9,dim(as.matrix(response))[2]),lower_bound=T, max_eval=max(30,20+5*dim(response)[2]),initial_values=NA,num_initial_values=2)

Arguments

design a matrix of inputs.
response a matrix of outputs where each row is one runs of the computer model output.
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 FALSE meaning the mean is not zero. TRUE means the mean is 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 boolean value. T means nugget should be estimated and F means nugget is fixed or not estimated. The default value is F F.
range.par either NA or a vector. If it is NA, it means range parameters are estimated; otherwise range parameters are given. The default value is NA.
prior_choice the choice of prior for range parameters and noise-variance parameters. ref_xi and ref_gamma means the reference prior with reference prior with the log of inverse range parameterization ξ or range parameterization γ. ref_approx uses the jointly robust prior to approximate the reference prior. The default choice is ref_approx.
a prior parameters in the jointly robust prior. The default value is 0.2.
b prior parameters in the jointly robust prior. The default value is n^{(M-1)/(p)}(a+p) where n is the number of runs and p is the dimension of the input vector.
kernel_type A vector specifying the type of kernels of each coordinate of the input. matern_3_2 and matern_5_2 are Matern correlation with roughness parameter 3/2 and 5/2 respectively. pow_exp is power exponential correlation with roughness parameter alpha. If pow_exp is to be used, one needs to specify its roughness parameter alpha. The default choice is matern_5_2.
alpha roughness parameters in the pow_exp correlation functions. The default choice is a vector with each entry being 1.9.
lower_bound boolean value. T means the default lower bounds of the inverse range parameters are used to constrained the optimization and F means the optimization is unconstrained. The default value is T and we also suggest to use F in various scenarios.
max_eval the maximum number of steps to estimate the range and nugget parameters.
initial_values a matrix of initial values of the kernel parameters to be optimized numerically, where each row of the matrix contains a set of the log inverse range parameters and the log nugget parameter.

num_initial_values the number of initial values of the kernel parameters in optimization.

Value

ppgasp returns a S4 object of class ppgasp (see ppgasp-class).

Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]

Maintainer: Mengyang Gu <mgu6@jhu.edu>

References


Examples

library(RobustGaSP)

###PP GaSP model for the humanity model
data(humanity_model)
###pp gasp
m.ppgasp=ppgasp(design=humanity.X,response=humanity.Y,nugget.est= TRUE)
show(m.ppgasp)

###make predictions
m_pred=predict(m.ppgasp,humanity.Xt)
sqrt(mean((m_pred$mean-humanity.Yt)^2))
mean(m_pred$upper95>humanity.Yt & humanity.Yt>m_pred$lower95)
mean(m_pred$upper95-m_pred$lower95)
sqrt( mean( (mean(humanity.Y)-humanity.Yt)^2 ) )

###with a linear trend on the selected input performs better
## Not run:
###PP GaSP Emulation with a linear trend for the humanity model
data(humanity_model)
###pp gasp with trend
n<dim(humanity.Y)[1]
n_testing<dim(humanity.Yt)[1]
H=cbind(matrix(1,n,1),humanity.X$foodC)
ppgasp-class

Description

S4 class for PP GaSP model if the range and noise-variance ratio parameters are given and/or have been estimated.

Objects from the Class

Objects of this class are created and initialized with the function `ppgasp` that computes the calculations needed for setting up the analysis.

Slots

- **p**: Object of class `integer`. The dimensions of the inputs.
- **num_obs**: Object of class `integer`. The number of observations.
- **k**: Object of class `integer`. The number of outputs in each computer model run.
- **input**: Object of class `matrix` with dimension n x p. The design of experiments.
- **output**: Object of class `matrix` with dimension n x k. Each row denotes a output vector in each run of the computer model.
- **X**: Object of class `matrix` of with dimension n x q. The mean basis function, i.e. the trend function.
- **zero_mean**: A character to specify whether the mean is zero or not. "Yes" means it has zero mean and "No" means the mean is not zero.
- **q**: Object of class `integer`. The number of mean basis.
- **LB**: Object of class `vector` with dimension p x 1. The lower bound for inverse range parameters beta.
- **beta_initial**: Object of class `vector` with the initial values of inverse range parameters p x 1.
- **beta_hat**: Object of class `vector` with dimension p x 1. The inverse-range parameters.
- **log_post**: Object of class `numeric` with the logarithm of marginal posterior.
predict.ppgasp

R0: Object of class list of matrices where the j-th matrix is an absolute difference matrix of the j-th input vector.

theta_hat: Object of class vector with dimension q x 1. The the mean (trend) parameter.

L: Object of class matrix with dimension n x n. The Cholesky decomposition of the correlation matrix R, i.e.

\[ L \% * t(L) = R \]

sigma2_hat: Object of the class matrix. The estimated variance parameter of each output.

LX: Object of the class matrix with dimension q x q. The Cholesky decomposition of the correlation matrix

\[ t(X) \% * R^{-1} \% * X \]

Cl: Object of the class vector used for the lower bound and the prior.

nugget: A numeric object used for the noise-variance ratio parameter.

nugget.est: A logical object of whether the nugget is estimated (T) or fixed (F).

kernel_type: A vector of character to specify the type of kernel to use.

alpha: Object of class vector with dimension p x 1 for the roughness parameters in the kernel.

call: The call to ppgasp function to create the object.

Methods

show Prints the main slots of the object.
predict See predict.

Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]
Maintainer: Mengyang Gu <mgu6@jhu.edu>

See Also

RobustGaSP for more details about how to create a RobustGaSP object.

predict.ppgasp Prediction for PP GaSP model

Description

Function to make prediction on the PP GaSP model after the PP GaSP model has been constructed.

Usage

```r
## S4 method for signature 'ppgasp'
predict(object, testing_input,
    testing_trend= matrix(1,dim(testing_input)[1],1),outasS3 = T, ...)
```
predict.ppgasp

Arguments

object       an object of class ppgasp.
testing_input  a matrix containing the inputs where the rgasp is to perform prediction.
testing_trend  a matrix of mean/trend for prediction.
outassS3      a boolean parameter indicating whether the output of the function should be as an S3 object.
...           Extra arguments to be passed to the function (not implemented yet).

Value

If the parameter outassS3=F, then the returned value is a S4 object of class predppgasp-class with

call:       call to predict.ppgasp function where the returned object has been created.
mean:       predictive mean for the testing inputs.
lower95:    lower bound of the 95% posterior credible interval.
upper95:    upper bound of the 95% posterior credible interval.
sd:         standard deviation of each testing_input.

If the parameter outassS3=T, then the returned value is a list with

mean        predictive mean for the testing inputs.
lower95     lower bound of the 95% posterior credible interval.
upper95     upper bound of the 95% posterior credible interval.
sd          standard deviation of each testing_input.

Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]
Maintainer: Mengyang Gu <mgu6@jhu.edu>

References


Examples

library(RobustGaSP)
#---------------------------------
# an example of environmental model
#---------------------------------

set.seed(1)
n=100
p=4

# using the latin hypercube will be better
library(lhs)
input_samples=maximinLHS(n,p)
input_samples=matrix(runif(n*p),n,p)
input=matrix(0,n,p)
input[,]=7+input_samples[,]*6
input[,]=0.02+input_samples[,]*1
input[,]=0.01+input_samples[,]*2.99
input[,]=30.01+input_samples[,]*0.285

k=400
output=matrix(0,n,k)
environ.4.data is an environmental model on a spatial-time vector
for(i in 1:n){
  output[i,]=environ.4.data(input[i,],s=seq(0.15,3,0.15),t=seq(3,60,3) )
}

# samples some test inputs
n_star=1000
sample_unif=matrix(runif(n_star*p),n_star,p)

testing_input=matrix(0,n_star,p)
testing_input[,]=7+sample_unif[,]*6
testing_input[,]=0.02+sample_unif[,]*1
testing_input[,]=0.01+sample_unif[,]*2.99
testing_input[,]=30.01+sample_unif[,]*0.285

testing_output=matrix(0,n_star,k)
for(i in 1:n_star){
  testing_output[i,]=environ.4.data(testing_input[i,],s=seq(0.15,3,0.15)
    ,t=seq(3,60,3) )
}

# we do a transformation of the output
# one can change the number of initial values to test
log_output=matrix(log(output+1),n,p)
m.ppgasp=ppgasp(design=input,response=log_output[,],kernel_type
  ='pow_exp',num_initial_values=2)
m_pred.ppgasp=predict(m.ppgasp,testing_input)

# we transform back for the prediction
m_pred_ppgasp_mean=exp(m_pred.ppgasp$mean)-1
mean_squared_error
mean( (m_pred_ppgasp_mean-testing_output)^2)

# variance of the testing outputs
var(as.numeric(testing_output))

# makes plots for the testing
par(mfrow=c(1,2))
predict.rgasp

**Prediction for Robust GaSP model**

**Description**

Function to make prediction on the robust GaSP model after the robust GaSP model has been constructed.

**Usage**

```r
## S4 method for signature 'rgasp'
predict(object, testing_input, testing_trend= matrix(1,dim(testing_input)[1],1), outassS = T,...)
```

**Arguments**

- `object` an object of class `rgasp`.
- `testing_input` a matrix containing the inputs where the `rgasp` is to perform prediction.
- `testing_trend` a matrix of mean/trend for prediction.
- `outassS` a boolean parameter indicating whether the output of the function should be as an `S3` object.
- `...` Extra arguments to be passed to the function (not implemented yet).
Value

If the parameter `outassS3=F`, then the returned value is a S4 object of class `predrgasp-class` with

- **call**: call to `predict.rgasp` function where the returned object has been created.
- **mean**: predictive mean for the testing inputs.
- **lower95**: lower bound of the 95% posterior credible interval.
- **upper95**: upper bound of the 95% posterior credible interval.
- **sd**: standard deviation of each testing input.

If the parameter `outassS3=T`, then the returned value is a list with

- **mean**: predictive mean for the testing inputs.
- **lower95**: lower bound of the 95% posterior credible interval.
- **upper95**: upper bound of the 95% posterior credible interval.
- **sd**: standard deviation of each testing input.

Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]
Maintainer: Mengyang Gu <mgu6@jhu.edu>

References


Examples

```r
# a 3 dimensional example
# dimensional of the inputs
dim_inputs <- 3
# number of the inputs
num_obs <- 30
# uniform samples of design
input <- matrix(runif(num_obs*dim_inputs), num_obs, dim_inputs)

# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
```
predict.rgasp

# input <- maximinLHS(n=num_obs, k=dim_inputs)  ## maximin lhd sample
# outputs from the 3 dim dettepepel3.data function

output = matrix(0,num_obs,1)
for(i in 1:num_obs){
  output[i]<-dettepepel3.data (input[i,])
}

# use constant mean basis, with no constraint on optimization
m1 <- rgasp(design = input, response = output, lower_bound=FALSE)

# the following use constraints on optimization
# m1 <- rgasp(design = input, response = output, lower_bound=TRUE)
# the following use a single start on optimization
# m1 <- rgasp(design = input, response = output, lower_bound=FALSE)

# number of points to be predicted
num_testing_input <- 5000
# generate points to be predicted
testing_input <- matrix(runif(num_testing_input*dim_inputs),num_testing_input,dim_inputs)
# Perform prediction
m1.predict <- predict(m1, testing_input)
# Predictive mean
m1.predict$mean

# The following tests how good the prediction is
testing_output <- matrix(0,num_testing_input,1)
for(i in 1:num_testing_input){
  testing_output[i]<-dettepepel3.data(testing_input[i,])
}

# compute the MSE, average coverage and average length
# out of sample MSE
MSE_emulator <- sum((m1.predict$mean-testing_output)^2)/(num_testing_input)

# proportion covered by 95% posterior predictive credible interval
prop_emulator <- length(which((m1.predict$lower95<=testing_output) & (m1.predict$upper95>=testing_output)))/num_testing_input

# average length of posterior predictive credible interval
length_emulator <- sum(m1.predict$upper95-m1.predict$lower95)/num_testing_input

# output of prediction
MSE_emulator
prop_emulator
length_emulator
# normalized RMSE
sqrt(MSE_emulator/mean(((testing_output-mean(output))^2)))

#-------------------------------------------------------------
# a 2 dimensional example with trend
# dimensional of the inputs
dim_inputs <- 2
# number of the inputs
num_obs <- 20

# uniform samples of design
input <- matrix(runif(num_obs*dim_inputs), num_obs, dim_inputs)
# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
# input <- maximinLHS(n=num_obs, k=dim_inputs) ##maximin lhd sample

# outputs from the 2 dim Brainin function
output <- matrix(0, num_obs, 1)
for(i in 1:num_obs){
  output[i]<-limetal.2.data (input[i,])
}

# mean basis (trend)
X<-cbind(rep(1, num_obs), input)

# use constant mean basis with trend, with no constraint on optimization
m2<- rgasp(design = input, response = output, trend =X, lower_bound=FALSE)

# number of points to be predicted
num_testing_input <- 5000
# generate points to be predicted
testing_input <- matrix(runif(num_testing_input*dim_inputs), num_testing_input, dim_inputs)

# trend of testing
testing_X<-cbind(rep(1, num_testing_input), testing_input)

# Perform prediction
m2.predict<-predict(m2, testing_input,testing_trend=testing_X)
# Predictive mean
m2.predict$mean

# The following tests how good the prediction is
testing_output <- matrix(0, num_testing_input, 1)
for(i in 1:num_testing_input)
  testing_output[i]<-limetal.2.data(testing_input[i,])

# compute the MSE, average coverage and average length
# out of sample MSE
MSE_emulator <- sum((m2.predict$mean-testing_output)^2)/(num_testing_input)
# proportion covered by 95% posterior predictive credible interval
prop_emulator <- length(which(m2.predict$lower95<=testing_output) & (m2.predict$upper95>=testing_output)) / num_testing_input

# average length of posterior predictive credible interval
length_emulator <- sum(m2.predict$upper95-m2.predict$lower95) / num_testing_input

# output of prediction
MSE_emulator = prop_emulator
length_emulator
# normalized RMSE
sqrt(MSE_emulator/mean((testing_output-mean(output))^2 ))

# an 8 dimensional example using only a subset inputs and a noise with unknown variance
# dimensional of the inputs
dim_inputs <- 8
# number of the inputs
num_obs <- 30

# uniform samples of design
input <- matrix(runif(num_obs*dim_inputs), num_obs, dim_inputs)
# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
# input <- maximinLHS(n=num_obs, k=dim_inputs) # maximin lhd sample

# rescale the design to the domain
input[,1]<-0.85*(0.15-0.05)*input[,1];
input[,2]<-100+(50000-100)*input[,2];
input[,3]<-63070+(115600-63070)*input[,3];
input[,4]<-990+(1110-990)*input[,4];
input[,5]<-63.1*(116-63.1)*input[,5];
input[,6]<-700+(820-700)*input[,6];
input[,7]<-1120+(1680-1120)*input[,7];
input[,8]<-9855+(12045-9855)*input[,8];

# outputs from the 8 dim Borehole function
output=matrix(0,num_obs,1)
for(i in 1:num_obs){
  output[i]=borehole(input[i,])
}

# use constant mean basis with trend, with no constraint on optimization
m3 <- rgasp(design = input[,c(1,4,6,7,8)], response = output,
            nugget.est=TRUE, lower_bound=FALSE)
predppgasp-class

# number of points to be predicted
num_testing_input <- 5000  
# generate points to be predicted
testing_input <- matrix(runif(num_testing_input*dim_inputs),num_testing_input,dim_inputs)

# rescale the points to the region to be predict
testing_input[,1]<-.05+(.15-.05)*testing_input[,1];
testing_input[,2]<-100+(5000-100)*testing_input[,2];
testing_input[,3]<-63070+(115600-63070)*testing_input[,3];
testing_input[,4]<-990+(1110-990)*testing_input[,4];
testing_input[,5]<-63.1+(116-63.1)*testing_input[,5];
testing_input[,6]<-7000+(820-7000)*testing_input[,6];
testing_input[,7]<-1120+(1680-1120)*testing_input[,7];
testing_input[,8]<-9855+(12045-9855)*testing_input[,8];

# Perform prediction
m3$predict<-predict(m3, testing_input[,c(1,4,6,7,8)])
# Predictive mean
m3$predict$mean

# The following tests how good the prediction is
testing_output <- matrix(0,num_testing_input,1)
for(i in 1:num_testing_input){
   testing_output[i]<-borehole(testing_input[i,])
}

# compute the MSE, average coverage and average length
# out of sample MSE
MSE_emulator <- sum((m3$predict$mean-testing_output)^2)/(num_testing_input)

# proportion covered by 95% posterior predictive credible interval
prop_emulator <- length(which((m3$predict$lower95<=testing_output) &
   (m3$predict$upper95>=testing_output)))/num_testing_input

# average length of posterior predictive credible interval
length_emulator <- sum(m3$predict$upper95-m3$predict$lower95)/num_testing_input

# output of sample prediction
MSE_emulator
prop_emulator
length_emulator
# normalized RMSE
sqrt(MSE_emulator/mean((testing_output-mean(output))^2 ))

predppgasp-class Predicted PP GaSP class
Description
S4 class for the prediction of a PP GaSP model

Objects from the Class
Objects of this class are created and initialized with the function `predict.ppgasp` that computes the prediction on the PP GaSP model after the PP GaSP model has been constructed.

Slots
- `call`: call to `predict.ppgasp` function where the returned object has been created.
- `mean`: predictive mean for the testing inputs.
- `lower95`: lower bound of the 95% posterior credible interval.
- `upper95`: upper bound of the 95% posterior credible interval.
- `sd`: standard deviation of each testing input.

Author(s)
Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]
Maintainer: Mengyang Gu <mgu6@jhu.edu>

See Also
- `predict.ppgasp` for more details about how to make predictions based on a ppgasp object.

---

predrgasp-class

Description
S4 class for the prediction of a Robust GaSP

Objects from the Class
Objects of this class are created and initialized with the function `predict.rgasp` that computes the prediction on Robust GaSP models after the Robust GaSP model has been constructed.

Slots
- `call`: call to `predict.rgasp` function where the returned object has been created.
- `mean`: predictive mean for the testing inputs.
- `lower95`: lower bound of the 95% posterior credible interval.
- `upper95`: upper bound of the 95% posterior credible interval.
- `sd`: standard deviation of each testing input.
Author(s)
Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]
Maintainer: Mengyang Gu <mgu6@jhu.edu>

See Also
predict.rgasp for more details about how to make predictions based on a rgasp object.

rgasp

Setting up the robust GaSP model

Description
Setting up the robust GaSP model for estimating the parameters (if the parameters are not given).

Usage
rgasp(design, response, trend=matrix(1,length(response),1), zero.mean="No", nugget=0, nugget.est=F, range.par=NA, prior_choice='ref_approx', a=0.2, b=1/(length(response))^((1/dim(as.matrix(design))[2]))*(1+dim(as.matrix(design))[2]), kernel_type=’matern_5_2’, alpha=rep(1.9,dim(as.matrix(design))[2]), lower_bound=T, max_eval=max(30,20+5*dim(design)[2]), initial_values=NA, num_initial_values=2)

Arguments
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 FALSE meaning the mean is not zero. TRUE means the mean is 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 boolean value. T means nugget should be estimated and F means nugget is fixed or not estimated. The default value is F F.
range.par either NA or a vector. If it is NA, it means range parameters are estimated; otherwise range parameters are given. The default value is NA.
prior_choice the choice of prior for range parameters and noise-variance parameters. ref_xi and ref_gamma means the reference prior with reference prior with the log of inverse range parameterization ξ or range parameterization γ. ref_approx uses the jointly robust prior to approximate the reference prior. The default choice is ref_approx.
a  prior parameters in the jointly robust prior. The default value is 0.2.

b  prior parameters in the jointly robust prior. The default value is $n^{-(-1/p)(a+p)}$
where $n$ is the number of runs and $p$ is the dimension of the input vector.

kernel_type  A vector specifying the type of kernels of each coordinate of the input. matern_3_2
and matern_5_2 are Matern correlation with roughness parameter 3/2 and 5/2 respectively.
pow_exp is power exponential correlation with roughness parameter alpha. If pow_exp is to be used, one needs to specify its roughness
parameter alpha. The default choice is matern_5_2.

alpha  roughness parameters in the pow_exp correlation functions. The default choice
is a vector with each entry being 1.9.

lower_bound  boolean value. T means the default lower bounds of the inverse range param-
eters are used to constrained the optimization and F means the optimization is
unconstrained. The default value is T and we also suggest to use F in various
scenarios.

max_eval  the maximum number of steps to estimate the range and nugget parameters.

initial_values  a matrix of initial values of the kernel parameters to be optimized numerically,
where each row of the matrix contains a set of the log inverse range parameters
and the log nugget parameter.

num_initial_values  the number of initial values of the kernel parameters in optimization.

Value

rgasp returns a S4 object of class rgasp (see rgasp-class).

Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]

Maintainer: Mengyang Gu <mgu6@jhu.edu>

References

M. Gu, X. Wang and J.O. Berger (2018), Robust Gaussian stochastic process emulation, Annals of
Statistics, 46(6A), 3038-3066.

M. Gu (2018), Jointly robust prior for Gaussian stochastic process in emulation, calibration and

M. Gu. (2016). Robust uncertainty quantification and scalable computation for computer models

M. Gu. and J.O. Berger (2016). Parallel partial Gaussian process emulation for computer models

Wolpert (2014), Automating emulator construction for geophysical hazard maps. SIAM/ASA Journal
on Uncertainty Quantification, 2(1), 126-152.
Examples

```r
library(RobustGaSP)
#-------------------------
# a 3 dimensional example
#-------------------------
# dimensional of the inputs
dim_inputs <- 3
# number of the inputs
num_obs <- 30
# uniform samples of design
input <- matrix(runif(num_obs*dim_inputs), num_obs, dim_inputs)

# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
# input <- maximinLHS(n=num_obs, k=dim_inputs) ##maximin lhd sample

###
# outputs from the 3 dim dettepepel.3.data function

output = matrix(0,num_obs,1)
for(i in 1:num_obs){
  output[i]<-dettepepel.3.data(input[i,])
}

# use constant mean basis, with no constraint on optimization
m1<- rgasp(design = input, response = output, lower_bound=FALSE)

###
# a 1 dimensional example with zero mean
#-------------------------

input=10*seq(0,1,1/14)
output<higdon.1.data(input)
#the following code fit a GaSP with zero mean by setting zero.mean="Yes"
model<- rgasp(design = input, response = output, zero.mean="Yes")
model

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

########plot predictive distribution
testing_output=higdon.1.data(testing_input)
plot(testing_input,model.predict$mean,type='l',col='blue',
  xlab='input',ylab='output')
polygon( c(testing_input,rev(testing_input)),c(model.predict$lower95, rev(model.predict$upper95)),col = "grey80", border = FALSE)
lines(testing_input, testing_output)
lines(testing_input,model.predict$mean,type='l',col='blue')
```
rgasp

lines(input, output,type='p')

## mean square erros
mean((model.predict$mean-testing_output)^2)

#--------------------------------------------------
# a 2 dimensional example with trend
#--------------------------------------------------
# dimensional of the inputs
dim_inputs <- 2
# number of the inputs
num_obs <- 20

# uniform samples of design
input <-matrix(runif(num_obs*dim_inputs), num_obs,dim_inputs)
# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
# input <- maximinLHS(n=num_obs, k=dim_inputs) # maximin lhd sample

# outputs from a 2 dim function
output <- matrix(0,num_obs,1)
for(i in 1:num_obs){
  output[i]<-limetal2.data (input[i,])
}

####trend or mean basis
X<-cbind(rep(1,num_obs), input )

# use constant mean basis with trend, with no constraint on optimization
m2<- rgasp(design = input, response = output,trend =X, lower_bound=FALSE)

show(m2)   # show this rgasp object
m2@beta_hat  # estimated inverse range parameters
m2@theta_hat

#--------------------------------------------------
# an 8 dimensional example using only a subset inputs and a noise with unknown variance
#--------------------------------------------------
# dimensional of the inputs
dim_inputs <- 8
# number of the inputs
num_obs <- 30

# uniform samples of design
input <-matrix(runif(num_obs*dim_inputs), num_obs,dim_inputs)
# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
# input <- maximinLHS(n=num_obs, k=dim_inputs) # maximin lhd sample
# rescale the design to the domain
input[,1]<-0.05*(0.15-0.05)*input[,1];
input[,2]<-100*(50000-100)*input[,2];
input[,3]<-63070*(115600-63070)*input[,3];
input[,4]<-990*(1110-990)*input[,4];
input[,5]<-63.1*(116-63.1)*input[,5];
input[,6]<-700*(820-700)*input[,6];
input[,7]<-1120*(1680-1120)*input[,7];
input[,8]<-9855*(12045-9855)*input[,8];

# outputs from the 8 dim Borehole function
output=matrix(0,num_obs,1)
for(i in 1:num_obs){
    output[i]=borehole(input[i,])
}

# use constant mean basis with trend, with no constraint on optimization
m3<-rgasp(design = input[,c(1,4,6,7,8)], response = output,
          nugget.est=TRUE, lower_bound=FALSE)

m3@beta_hat  # estimated inverse range parameters
m3@nugget

---

**rgasp-class**

*Robust GaSP class*

---

**Description**

S4 class for Robust GaSP if the range and noise-variance ratio parameters are given and/or have been estimated.

**Objects from the Class**

Objects of this class are created and initialized with the function `rgasp` that computes the calculations needed for setting up the analysis.

**Slots**

`p`: Object of class `integer`. The dimensions of the inputs.

`num_obs`: Object of class `integer`. The number of observations.
Input: Object of class matrix with dimension n x p. The design of experiments.

Output: Object of class matrix with dimension n x 1. The Observations or output vector.

x: Object of class matrix of with dimension n x q. The mean basis function, i.e. the trend function.

zero_mean: A character to specify whether the mean is zero or not. "Yes" means it has zero mean and "No" means the mean is not zero.

q: Object of class integer. The number of mean basis.

LB: Object of class vector with dimension p x 1. The lower bound for inverse range parameters beta.

beta_initial: Object of class vector with the initial values of inverse range parameters p x 1.

beta_hat: Object of class vector with dimension p x 1. The inverse-range parameters.

log_post: Object of class numeric with the logarithm of marginal posterior.

R: Object of class list of matrices where the j-th matrix is an absolute difference matrix of the j-th input vector.

theta_hat: Object of class vector with dimension q x 1. The the mean (trend) parameter.

L: Object of class matrix with dimension n x n. The Cholesky decomposition of the correlation matrix R, i.e.

\[ L^\% * t(L) = R \]

sigma2_hat: Object of the class numeric. The estimated variance parameter.

LX: Object of the class matrix with dimension q x q. The Cholesky decomposition of the correlation matrix

\[ t(X)^\% * R^{-1} * X \]

CL: Object of the class vector used for the lower bound and the prior.

nugget: A numeric object used for the noise-variance ratio parameter.

nugget.est: A logical object of whether the nugget is estimated (T) or fixed (F).

kernel_type: A vector of character to specify the type of kernel to use.

alpha: Object of class vector with dimension p x 1 for the roughness parameters in the kernel.

call: The call to rgasp function to create the object.

Methods

show Prints the main slots of the object.

predict See predict.

Note

The response output must have one dimension. The number of observations in input must be equal to the number of experiments output.

Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]

Maintainer: Mengyang Gu <mgu6@jhu.edu>
See Also

RobustGaSP for more details about how to create a RobustGaSP object.

---

show

Show Robust GaSP object

Description

Function to print Robust GaSP models after the Robust GaSP model has been constructed.

Usage

```r
## S4 method for signature 'rgasp'
show(object)
```

Arguments

- `object`: an object of class `rgasp`.

Author(s)

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]

Maintainer: Mengyang Gu <mgu6@jhu.edu>

Examples

```
# a 3 dimensional example
#--------------------------
# dimensional of the inputs
dim_inputs <- 3
# number of the inputs
num_obs <- 30
# uniform samples of design
input <- matrix(runif(num_obs*dim_inputs), num_obs, dim_inputs)

# Following codes use maximin Latin Hypercube Design, which is typically better than uniform
# library(lhs)
# input <- maximinLHS(n=num_obs, k=dim_inputs) ##maximin lhd sample

####
# outputs from the 3 dim dettepapel.3.data function

output = matrix(0, num_obs, 1)
for(i in 1:num_obs){
  output[i]<-dettepapel.3.data (input[i,])
}

# use constant mean basis, with no constraint on optimization
```
show.ppgasp

m1 <- rgasp(design = input, response = output, lower_bound = FALSE)

# the following use constraints on optimization
# m1 <- rgasp(design = input, response = output, lower_bound = TRUE)

# the following use a single start on optimization
# m1 <- rgasp(design = input, response = output, lower_bound = FALSE)

show(m1)

---

**show.ppgasp**  
*Show parallel partial Gaussian stochastic process (PP GaSP) object*

**Description**

Function to print the PP GaSP model after the PP GaSP model has been constructed.

**Usage**

```r
## S4 method for signature 'ppgasp'
show(object)
```

**Arguments**

- `object` an object of class `ppgasp`.

**Author(s)**

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]

Maintainer: Mengyang Gu <mgu6@jhu.edu>

**Examples**

```r
library(RobustGaSP)

## PP GaSP model for the humanity model
data(humanity_model)
## pp gasp
m.ppgasp <- ppgasp(design = humanity.X, response = humanity.Y, nugget.est = TRUE)
show(m.ppgasp)
```
simulate  

*Sample for Robust GaSP model*

**Description**

Function to sample Robust GaSP after the Robust GaSP model has been constructed.

**Usage**

```r
## S4 method for signature 'rgasp'
simulate(object, testing_input, num_sample=1,
           testing_trend= matrix(1, dim(testing_input)[1], 1), ...)
```

**Arguments**

- `object`: an object of class `rgasp`.
- `testing_input`: a matrix containing the inputs where the `rgasp` is to sample.
- `num_sample`: number of samples one wants.
- `testing_trend`: a matrix of mean/trend for prediction.
- `...`: Extra arguments to be passed to the function (not implemented yet).

**Value**

The returned value is a matrix where each column is a sample on the prespecified inputs.

**Author(s)**

Mengyang Gu [aut, cre], Jesus Palomo [aut], James Berger [aut]

Maintainer: Mengyang Gu <mgu6@jhu.edu>

**References**


**Examples**

```r
# a 1 dimensional example

###dim higdon.1.data
p1 = 1     ###dimensional of the inputs
dim_inputs1 <- p1
n1 = 15    ###sample size or number of training computer runs you have
num_obs1  <- n1
input1 = 10*matrix(runif(num_obs1*dim_inputs1), num_obs1, dim_inputs1) #uniform
```
### lhs is better
#library(lhs)
#input1 = l0+maximinLHS(n=num_obs1, k=dim_inputs1)  ##maximin lhd sample
output1 = matrix(0,num_obs1,1)
for(i in 1:num_obs1){
  output1[i]=higdon.t.data(input1[i])
}

m1<- rgasp(design = input1, response = output1, lower_bound=FALSE)

### locations to samples
testing_input1 = seq(0,1,1/50)
testing_input1=as.matrix(testing_input1)
### draw 10 samples
m1_sample=simulate(m1,testing_input1,num_sample=10)

### plot these samples
matplot(testing_input1,m1_sample, type='l',xlab='input',ylab='output')
lines(input1,output1,type='p')
Index

*Topic classes
  ppgasp-class, 13
  predppgasp-class, 22
  predrgasp-class, 23
  rgasp-class, 28

*Topic computer model
  RobustGaSP-package, 2

*Topic datasets
  humanity_model, 7

*Topic emulation
  RobustGaSP-package, 2

*Topic package
  RobustGaSP-package, 2

*Topic simulation
  RobustGaSP-package, 2

findInertInputs, 5

humanity.X (humanity_model), 7
humanity.Xt (humanity_model), 7
humanity.Y (humanity_model), 7
humanity.Yt (humanity_model), 7
humanity_model, 7

leave_one_out_rgasp, 8

plot, 9
plot,rgasp-method (plot), 9
plot.rgasp (plot), 9
ppgasp, 10, 13
ppgasp-class, 13
ppgasp-method (ppgasp), 10
predict, 14, 29
predict (predict.rgasp), 17
predict,ppgasp-method (predict.ppgasp), 14
predict,rgasp-method (predict.rgasp), 17
predict.ppgasp, 14, 23
predict.ppgasp-class (predict.ppgasp), 14
predict.rgasp, 17, 23, 24
predict.rgasp-class (predict.rgasp), 17
predppgasp-class, 22
predrgasp-class, 23
rgasp, 8, 24, 28
rgasp-class, 28
rgasp-method (rgasp), 24
RobustGaSP, 3, 14, 30
RobustGaSP (RobustGaSP-package), 2
RobustGaSP-package, 2

show, 30
show,ppgasp-method (show.ppgasp), 31
show,rgasp-method (show), 30
show.ppgasp, 31
show.ppgasp-class (show.ppgasp), 31
show.rgasp (show), 30
show.rgasp-class (show), 30
simulate, 32
simulate,rgasp-method (simulate), 32
simulate.rgasp (simulate), 32
simulate.rgasp-class (simulate), 32