

Package ‘RobustCalibration’

September 18, 2022

Type Package

Title Robust Calibration of Imperfect Mathematical Models

Version 0.5.3

Date 2022-09-18

Maintainer Mengyang Gu <mengyang@pstat.ucsb.edu>

Author Mengyang Gu [aut, cre]

Description Implements full Bayesian analysis for calibrating mathematical models with new methodology for modeling the discrepancy function. It allows for emulation, calibration and prediction using complex mathematical model outputs and experimental data. See the reference: Mengyang Gu and Long Wang (2018) <[arXiv:1707.08215](#)>, Mengyang Gu, Fangzheng Xie and Long Wang (2021) <[arXiv:2109.00000](#)>, and Erika McPhillips (2021) <[arXiv:1810.11664](#)>.

License GPL (>= 2)

Depends methods

Imports Rcpp (>= 0.12.3), RobustGaSP (>= 0.6.4), nloptr (>= 1.0.4)

LinkingTo Rcpp, RcppEigen

NeedsCompilation yes

Repository CRAN

RoxygenNote 5.0.1

Date/Publication 2022-09-18 19:56:06 UTC

R topics documented:

RobustCalibration-package	2
predict	5
predictobj.rcalibration-class	11
predictobj.rcalibration_MS-class	12
predict_MS	13
rcalibration	16
rcalibration-class	21
rcalibration_MS	23
rcalibration_MS-class	28
show	30

 RobustCalibration-package

Robust Calibration of Imperfect Mathematical Models

Description

Implements full Bayesian analysis for calibrating mathematical models with new methodology for modeling the discrepancy function. It allows for emulation, calibration and prediction using complex mathematical model outputs and experimental data. See the reference: Mengyang Gu and Long Wang (2018) <arXiv:1707.08215>, Mengyang Gu, Fangzheng Xie and Long Wang (2021) <arXiv:1807.03829>, Mengyang Gu, Kyle Anderson and Erika McPhillips (2021) <arXiv:1810.11664>.

Details

The DESCRIPTION file:

```

Package:      RobustCalibration
Type:        Package
Title:       Robust Calibration of Imperfect Mathematical Models
Version:     0.5.3
Date:       2022-09-18
Authors@R:  c(person(given="Mengyang",family="Gu",role=c("aut","cre"),email="mengyang@pstat.ucsb.edu"))
Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>
Author:     Mengyang Gu [aut, cre]
Description: Implements full Bayesian analysis for calibrating mathematical models with new methodology for modeling the discrepancy function.
License:    GPL (>= 2)
Depends:    methods
Imports:    Rcpp (>= 0.12.3), RobustGaSP (>= 0.6.4), nloptr (>= 1.0.4)
LinkingTo:  Rcpp, RcppEigen
NeedsCompilation: yes
Repository: CRAN
Packaged:   2018-10-07 20:06:35 UTC; gumengyang
RoxygenNote: 5.0.1
Date/Publication: 2018-05-14 04:26:37 UTC
  
```

Index of help topics:

```

RobustCalibration-package
    Robust Calibration of Imperfect Mathematical
    Models
predict      Prediction for the robust calibration model
predict_MS  Prediction for the robust calibration model for
            multiple sources
predictobj.rcalibration-class
    Predictive results for the Robust Calibration
  
```

```

class
predictobj.rcalibration_MS-class Predictive results for the Robust Calibration
class
rcalibration Setting up the robust Calibration model
rcalibration-class Robust Calibration class
rcalibration_MS Setting up the robust Calibration model for
multiple sources data
rcalibration_MS-class Robust Calibration for multiple sources class
show Show an Robust Calibration object.

```

Robust calibration of imperfect mathematical models and prediction using experimental data

Author(s)

Mengyang Gu [aut, cre]
 Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

A. O'Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.

Bayarri, Maria J and Berger, James O and Paulo, Rui and Sacks, Jerry and Cafeo, John A and Cavendish, James and Lin, Chin-Hsu and Tu, Jian (2007) *A framework for validation of computer models*. *Technometrics*. **49**, 138–154.

M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.

M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.

M. Gu (2018) *Jointly Robust Prior for Gaussian Stochastic Process in Emulation, Calibration and Variable Selection*. arXiv preprint arXiv:1804.09329.

See Also

[RobustGaSP](#)

Examples

```

##-----
##A simple example where the math model is not biased
##-----
## the reality
test_funct_eg1<-function(x){
  sin(pi/2*x)
}

```

```

## obtain 25 data from the reality plus a noise
set.seed(1)
## 10 data points are very small, one may want to add more data
n=15
input=seq(0,4,4/(n-1))
input=as.matrix(input)

output=test_funct_eg1(input)+rnorm(length(input),mean=0,sd=0.2)

## plot input and output
#plot(input,output)
#num_obs=n=length(output)

## the math model
math_model_eg1<-function(x,theta){
  sin(theta*x)
}

##fit the S-GaSP model for the discrepancy
##one can choose the discrepancy_type to GaSP, S-GaSP or no discrepancy
##p_theta is the number of parameters to calibrate and user needs to specifiy
##one may also want to change the number of posterior samples by change S and S_0
##one may change sd_proposal for the standard derivation of the proposal distribution
## one may also add a mean by setting X=... and have_trend=TRUE
model_sgasp=rcalibration(design=input, observations=output, p_theta=1,simul_type=1,
                        math_model=math_model_eg1,theta_range=matrix(c(0,3),1,2)
                        ,S=10000,S_0=2000,discrepancy_type='S-GaSP')

##posterior samples of calibration parameter and value
## the value is
plot(model_sgasp$post_sample[,1],type='l',xlab='num',ylab=expression(theta))
plot(model_sgasp$post_value,type='l',xlab='num',ylab='posterior value')

show(model_sgasp)

#-----
# Example: an example used in Susie Bayarri et. al. 2007 Technometrics paper
#-----

##reality
test_funct_eg1<-function(x){
  3.5*exp(-1.7*x)+1.5
}

##math model
math_model_eg1<-function(x,theta){

```

```

    5*exp(-x*theta)
  }

  ## noise observations (sampled from reality + independent Gaussian noises)
  ## each has 3 replicates
  input=c(rep(.110,3),rep(.432,3),rep(.754,3),rep(1.077,3),rep(1.399,3),rep(1.721,3),
          rep(2.043,3),rep(2.366,3),rep(2.688,3),rep(3.010,3))
  output=c(4.730,4.720,4.234,3.177,2.966,3.653,1.970,2.267,2.084,2.079,2.409,2.371,1.908,1.665,1.685,
          1.773,1.603,1.922,1.370,1.661,1.757,1.868,1.505,1.638,1.390,1.275,1.679,1.461,1.157,1.530)

  n_stack=length(output)/3
  output_stack=rep(0,n_stack)
  input_stack=rep(0,n_stack)
  for(j in 1:n_stack){
    output_stack[j]=mean(output[ ((j-1)*3+1):(3*j)])
    input_stack[j]=mean(input[ ((j-1)*3+1):(3*j)])
  }
  output_stack=as.matrix(output_stack)
  input_stack=as.matrix(input_stack)
  ## plot the output and stack
  #plot(input,output,pch=16,col='red')
  #lines(input_stack,output_stack,pch=16,col='blue',type='p')

  ## fit the model with S-GaSP for the discrepancy
  model_sgasp=rcalibration(design=input_stack, observations=output_stack, p_theta=1,simul_type=1,
                          math_model=math_model_eg1,theta_range=matrix(c(0,10),1,2),S=10000,
                          S_0=2000,discrepancy_type='S-GaSP')

  #posterior
  plot(model_sgasp$post_sample[,1],type='l',xlab='num',ylab=expression(theta))
  show(model_sgasp)

```

 predict

Prediction for the robust calibration model

Description

Function to make prediction on Robust Calibration models after the rcalibration class has been constructed.

Usage

```
## S4 method for signature 'rcalibration'
```

```
predict(object, testing_input, X_testing=NULL,
        n_thinning=10,
        testing_output_weights=NULL,
        interval_est=NULL, interval_data=F,
        math_model=NULL, test_loc_index_emulator=NULL, ...)
```

Arguments

object an object of class `rcalibration`.

testing_input a matrix containing the inputs where the predict is to perform prediction.

X_testing a matrix of mean/trend for prediction.

n_thinning number of points further thinning the MCMC posterior samples.

testing_output_weights the weight of testing outputs.

interval_est a vector for the the posterior credible interval. If `interval_est` is `NULL`, we do not compute the posterior credible interval. It can be specified as a vector of values ranging from zero to one. E.g. if `interval_est=c(0.025, 0.975)`, the 95 posterior credible interval will be computed.

interval_data a bool value to decide whether the experimental noise is included for computing the posterior credible interval.

math_model a function for the math model to be calibrated.

test_loc_index_emulator a vector of the location index from the ppgasp emulator to output. Only useful for vectorized output computer model emulated by the ppgasp emulator.

... extra arguments to be passed to the function (not implemented yet).

Value

The returned value is a S4 CClass `predictobj.rcalibration`.

Author(s)

Mengyang Gu [aut, cre]

Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

A. O'Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.

Bayarri, Maria J and Berger, James O and Paulo, Rui and Sacks, Jerry and Cafeo, John A and Cavendish, James and Lin, Chin-Hsu and Tu, Jian (2007) *A framework for validation of computer models*. *Technometrics*. **49**, 138–154.

M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.

M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.

M. Gu (2018) *Jointly Robust Prior for Gaussian Stochastic Process in Emulation, Calibration and Variable Selection*. arXiv preprint arXiv:1804.09329.

Examples

```
#-----
# Example: an example used in Susie Bayarri et. al. 2007 Technometrics paper
#-----

##reality
test_funct_eg1<-function(x){
  3.5*exp(-1.7*x)+1.5
}

##math model
math_model_eg1<-function(x,theta){
  5*exp(-x*theta)
}

## noise observations (sampled from reality + independent Gaussian noises)
## each has 3 replicates
input=c(rep(.110,3),rep(.432,3),rep(.754,3),rep(1.077,3),rep(1.399,3),rep(1.721,3),
        rep(2.043,3),rep(2.366,3),rep(2.688,3),rep(3.010,3))
output=c(4.730,4.720,4.234,3.177,2.966,3.653,1.970,2.267,2.084,2.079,2.409,2.371,1.908,1.665,1.685,
        1.773,1.603,1.922,1.370,1.661,1.757,1.868,1.505,1.638,1.390,1.275,1.679,1.461,1.157,1.530)

## calculating the average or the stack data
n_stack=length(output)/3
output_stack=rep(0,n_stack)
input_stack=rep(0,n_stack)
for(j in 1:n_stack){
  output_stack[j]=mean(output[ ((j-1)*3+1):(3*j)])
  input_stack[j]=mean(input[ ((j-1)*3+1):(3*j)])
}
output_stack=as.matrix(output_stack)
input_stack=as.matrix(input_stack)
## plot the output and stack output
#plot(input,output,pch=16,col='red')
#lines(input_stack,output_stack,pch=16,col='blue',type='p')

## fit model using S-GaSP for the discrepancy
## one can change S and S_0 for the number of posterior and burn-in samples
## Normally you may need a larger number of posterior sample
## you can set S=50000 and S_0=5000
## one may also change the sd of the proposal distribution using sd_proposal
model_sgasp=rcalibration(design=input_stack, observations=output_stack, p_theta=1,simul_type=1,
                        math_model=math_model_eg1,theta_range=matrix(c(0,10),1,2),
```

```

S=10000,S_0=2000,discrepancy_type='S-GaSP')

# one can fit the GaSP model for discrepancy function by discrepancy_type='GaSP'
# one can fit a model without the discrepancy function by discrepancy_type='no-discrepancy'

## posterior of the calibration parameter
#plot(model_sgasp@post_sample[,1],type='l',xlab='num',ylab=expression(theta))
show(model_sgasp)

##

## test data set
testing_input=as.matrix(seq(0,6,0.02))

##perform prediction
prediction_sgasp=predict(model_sgasp,testing_input,math_model=math_model_eg1,
                        interval_est=c(0.025,0.975),interval_data=TRUE,
                        n_thinning =20 )

##real test output
testing_output=test_funct_eg1(testing_input)

##the prediction by S-GaSP
min_val=min(prediction_sgasp@mean,prediction_sgasp@interval,output,testing_output)
max_val=max(prediction_sgasp@mean,prediction_sgasp@interval,output,testing_output)

plot(testing_input,prediction_sgasp@mean,type='l',col='blue',xlab='x',ylab='y',
      ylim=c(min_val,max_val) )
lines(testing_input,prediction_sgasp@interval[,1],col='blue',lty=2)
lines(testing_input,prediction_sgasp@interval[,2],col='blue',lty=2)

lines(input,output,type='p')
lines(testing_input,prediction_sgasp@math_model_mean,col='blue',lty=3)

lines(testing_input,testing_output,type='l')

legend("topright", legend=c("reality", "predictive mean", "95 percent posterior credible interval",
                            "predictive mean of the math model"),
      col=c("black", "blue", "blue", "blue"), lty=c(1,1,2,3),cex=.6)

## MSE if the math model and discrepancy are used for prediction
mean((testing_output-prediction_sgasp@mean)^2)

## MSE if the math model is used for prediction
mean((testing_output-prediction_sgasp@math_model_mean)^2)

#####
#the example with a mean structure
#####

```



```

##now let's fit model with mean
model_sgasp_with_mean=rcalibration(design=input_stack, observations=output_stack,
                                   p_theta=1,X=matrix(1,dim(input_stack)[1],1),
                                   have_trend=TRUE,simul_type=1,
                                   math_model=math_model_eg1,
                                   theta_range=matrix(c(0,10),1,2),
                                   S=10000,S_0=2000,
                                   discrepancy_type='S-GaSP')

#posterior
#plot(model_sgasp_with_mean@post_sample[,1],type='l',xlab='num',ylab=expression(theta))
show(model_sgasp_with_mean)

## test data set
testing_input=as.matrix(seq(0,6,0.02))

prediction_sgasp_with_mean=predict(model_sgasp_with_mean,testing_input, X_testing=matrix(1,dim
(testing_input)[1],1),
math_model=math_model_eg1,n_thinning = 50,
interval_est=c(0.025,0.975),interval_data=TRUE)

##plot for the S-GaSP
##for this example, with a mean structure, it fits much better
min_val=min(prediction_sgasp_with_mean@mean,output,testing_output,
prediction_sgasp_with_mean@interval[,1])
max_val=max(prediction_sgasp_with_mean@mean,output,testing_output,
prediction_sgasp_with_mean@interval[,2])

plot(testing_input,prediction_sgasp_with_mean@mean,type='l',col='blue',xlab='x',
      ylab='y',ylim=c(min_val,max_val) )
#lines(testing_input,prediction_sgasp_with_mean@interval[,1],col='blue',lty=2)
#lines(testing_input,prediction_sgasp_with_mean@interval[,2],col='blue',lty=2)

lines(input,output,type='p')
lines(testing_input,prediction_sgasp_with_mean@math_model_mean,col='blue',lty=3)
lines(testing_input,prediction_sgasp_with_mean@interval[,1],col='blue',lty=2)
lines(testing_input,prediction_sgasp_with_mean@interval[,2],col='blue',lty=2)

lines(testing_input,testing_output,type='l')

legend("topright", legend=c("reality", "predictive mean", "predictive mean of the math model"),
      col=c("black", "blue","blue"), lty=c(1,1,3),cex=.6)

## MSE if the math model and discrepancy are used for prediction
mean((testing_output-prediction_sgasp_with_mean@mean)^2)

## MSE if the math model is used for prediction
mean((testing_output-prediction_sgasp_with_mean@math_model_mean)^2)

```

```

## Not run:
#-----
#the example with the emulator
#-----

n_design=80

design_simul=matrix(runif(n_design*2),n_design,2)
#library(lhs)
#design_simul=maximinLHS(n=n_design,k=2)

design_simul[,1]=6*design_simul[,1]  ##the first one is the observed input x
design_simul[,2]=10*design_simul[,2]  ##the second one is the calibration parameter \theta

output_simul=math_model_eg1(design_simul[,1],design_simul[,2])

##this is a little slow compared with the previous model

model_sgasp_with_mean_emulator=rcalibration(design=input_stack, observations=output_stack,
      p_theta=1,simul_type=0,
      have_trend=T,X=matrix(1,dim(input_stack)[1],1),
      input_simul=design_simul, output_simul=output_simul,
      theta_range=matrix(c(0,10),1,2),
      S=10000,S_0=2000,discrepancy_type='S-GaSP')

##now the output is a list
show(model_sgasp_with_mean_emulator)

##here is the plot
plot(model_sgasp_with_mean_emulator@post_sample[,4],type='l',xlab='num',ylab=expression(theta))
plot(model_sgasp_with_mean_emulator@post_value,type='l',xlab='num',ylab='posterior value')

prediction_sgasp_with_mean_emulator=predict(model_sgasp_with_mean_emulator,testing_input,
      X_testing=matrix(1,dim(testing_input)[1],1),
      interval_est=c(0.025,0.975),
      interval_data=TRUE)

##for this example, with a mean structure, it fits much better
min_val=min(prediction_sgasp_with_mean_emulator@mean,output,testing_output,
      prediction_sgasp_with_mean_emulator@math_model_mean)
max_val=max(prediction_sgasp_with_mean_emulator@mean,output,testing_output,
      prediction_sgasp_with_mean_emulator@math_model_mean)

plot(testing_input,prediction_sgasp_with_mean_emulator@mean,type='l',col='blue',xlab='x',
      ylab='y',ylim=c(min_val,max_val) )
#lines(testing_input,prediction_sgasp_with_mean@interval[,1],col='blue',lty=2)
#lines(testing_input,prediction_sgasp_with_mean@interval[,2],col='blue',lty=2)

lines(input,output,type='p')
lines(testing_input,prediction_sgasp_with_mean_emulator@math_model_mean,col='blue',lty=3)

```

```

lines(testing_input,testing_output,type='l')

legend("topright", legend=c("reality", "predictive mean", "predictive mean of the math model"),
      col=c("black", "blue","blue"), lty=c(1,1,3),cex=.6)

## MSE if the math model and discrepancy are used for prediction
mean((testing_output-prediction_sgasp_with_mean_emulator@mean)^2)

## MSE if the math model is used for prediction
mean((testing_output-prediction_sgasp_with_mean_emulator@math_model_mean)^2)

## End(Not run)

```

predictobj.rcalibration-class

Predictive results for the Robust Calibration class

Description

S4 class for prediction after Robust recalibration with or without the specification of the discrepancy model.

Objects from the Class

Objects of this class are created and initialized with the function `predict` that computes the prediction and the uncertainty quantification.

Slots

`mean`: object of class vector. The predictive mean at testing inputs combining the mathematical model and discrepancy function.

`math_model_mean`: object of class vector. The predictive mean at testing inputs using only the mathematical model (and the trend if specified).

`math_model_mean_no_trend`: object of class vector. The predictive mean at testing inputs using only the mathematical model without the trend.

`interval`: object of class matrix. The upper and lower predictive credible interval. If `interval_data` is TRUE in the `predict.rcalibration`, the experimental noise is included for computing the predictive credible interval.

Author(s)

Mengyang Gu [aut, cre]

Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

- A. O’Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.
- M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.
- M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.

See Also

[predict.rcalibration](#) for more details about how to do prediction for a rcalibration object.

predictobj.rcalibration_MS-class

Predictive results for the Robust Calibration class

Description

S4 class for prediction after Robust rcalibration for multiple sources.

Objects from the Class

Objects of this class are created and initialized with the function [predict_MS](#) that computes the prediction and the uncertainty quantification.

Slots

- mean:** object of class `list`. Each element is a vector of the predictive mean at testing inputs combining the mathematical model and discrepancy function for each source.
- math_model_mean:** object of class `list`. Each element is a vector of the predictive mean at testing inputs using only the mathematical model (and the trend if specified).
- math_model_mean_no_trend:** object of class `list`. Each element is a vector of the predictive mean at testing inputs using only the mathematical model without the trend for each source.
- interval:** object of class `list`. Each element is a matrix of the upper and lower predictive credible interval. If `interval_data` is `TRUE` in the [predict_MS](#), the experimental noise is included for computing the predictive credible interval.

Author(s)

Mengyang Gu [aut, cre]

Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

- A. O'Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.
- M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.
- M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.

See Also

[predict_MS](#) for more details about how to do prediction for a rcalibration_MS object.

predict_MS	<i>Prediction for the robust calibration model for multiple sources</i>
------------	---

Description

Function to make prediction on Robust Calibration models after the rcalibration class has been constructed for multiple sources.

Usage

```
## S4 method for signature 'rcalibration_MS'
predict_MS(object, testing_input,
           X_testing=as.list(rep(0,object@num_sources)),
           testing_output_weights=NULL,
           n_thinning=10,
           interval_est=NULL,
           interval_data=rep(F,length(testing_input)),
           math_model=NULL,...)
```

Arguments

object	an object of class rcalibration_MS.
testing_input	a list of matrices containing the inputs where the predict_MS is to perform prediction.
X_testing	a list of matrices of mean/trend for prediction if specified.
testing_output_weights	a list of vecots for the weight of testing outputs for multiple sources.
n_thinning	number of points further thinning the MCMC posterior samples.
interval_est	a list of vectors for the posterior predictive credible interval for multiple sources. If interval_est is NULL, we do not compute the posterior credible interval. It can be specified as a vector of values ranging from zero to one. E.g.
interval_data	a vector of bool values to decide whether the experimental noise is included for computing the posterior credible interval.

math_model a list of functions for the math model to be calibrated for multiple sources.
 ... extra arguments to be passed to the function (not implemented yet).

Value

The returned value is a S4 CClass `predictobj.rcalibration`.

Author(s)

Mengyang Gu [aut, cre]

Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

A. O'Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.

K. R. Anderson and M. P. Poland (2016), *Bayesian estimation of magma supply, storage, and eruption rates using a multiphysical volcano model: Kilauea volcano, 2000-2012.. Earth and Planetary Science Letters*, **447**, 161-171.

K. R. Anderson and M. P. Poland (2017), *Abundant carbon in the mantle beneath Hawaii. Nature Geoscience*, **10**, 704-708.

Bayarri, Maria J and Berger, James O and Paulo, Rui and Sacks, Jerry and Cafeo, John A and Cavendish, James and Lin, Chin-Hsu and Tu, Jian (2007) *A framework for validation of computer models. Technometrics*. **49**, 138-154.

M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.

M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.

M. Gu (2018) *Jointly Robust Prior for Gaussian Stochastic Process in Emulation, Calibration and Variable Selection*. arXiv preprint arXiv:1804.09329.

Examples

```
#-----
# An example for calibrating and predicting mathematical models for data from multiple sources
#-----

library(RobustCalibration)

##reality
test_funct<-function(x){
  sin(pi*x/2)+2*cos(pi*x/2)
}
```

```

##math model from two sources
math_model_source_1<-function(x,theta){
  sin(theta*x)
}

math_model_source_2<-function(x,theta){
  cos(theta*x)
}

input1=seq(0,2,2/(10-1))
input2=seq(0,3,3/(10-1))
##
output1=test_func(input1)+rnorm(length(input1), sd=0.01)
output2=test_func(input2)+rnorm(length(input2), sd=0.02)

plot(input1, output1)
plot(input2, output2)

design=list()
design[[1]]=as.matrix(input1)
design[[2]]=as.matrix(input2)

observations=list()
observations[[1]]=output1
observations[[2]]=output2

p_theta=1

theta_range=matrix(0,p_theta,2)
theta_range[1,]=c(0, 8)
simul_type=c(1,1)

math_model=list()

math_model[[1]]=math_model_source_1
math_model[[2]]=math_model_source_2

## calibrating two mathematical models for these two sources
model_sgasp=rcalibration_MS(design=design, observations=observations, p_theta=1,
  simul_type=simul_type,math_model=math_model,
  theta_range=theta_range,
  S=10000,S_0=2000,
  discrepancy_type=rep('S-GaSP',length(design)))

#plot(model_sgasp@post_theta[,1],type='l')
mean(model_sgasp@post_theta[,1])

testing_input1=seq(0,2,2/(25-1))

```

```

testing_input2=seq(0,3,3/(25-1))

testing_input=list()
testing_input[[1]]=as.matrix(testing_input1)
testing_input[[2]]=as.matrix(testing_input2)

predict_sgasp=predict_MS(model_sgasp, testing_input, math_model=math_model)

testing_output1=test_funct(testing_input1)
testing_output2=test_funct(testing_input2)

plot(predict_sgasp@mean[[1]])
lines(testing_output1)

plot(predict_sgasp@mean[[2]])
lines(testing_output2)

```

rcalibration

Setting up the robust Calibration model

Description

Setting up the Calibration model for estimating the parameters via MCMC with or without a discrepancy function.

Usage

```

rcalibration(design, observations, p_theta=NULL,
X=matrix(0,dim(as.matrix(design))[1],1),
have_trend=FALSE, simul_type=1, input_simul=NULL,output_simul=NULL,simul_nug=FALSE,
loc_index_emulator=NULL,math_model=NULL, theta_range=NULL,
sd_proposal=c(rep(0.05,dim(as.matrix(theta_range) ) [1]),
rep(0.25,dim(as.matrix(design))[2]),0.25),
S=10000,S_0=2000,thinning=1, discrepancy_type='S-GaSP',
kernel_type='matern_5_2', lambda_z=NA, a=1/2-dim(as.matrix(design))[2], b=1,
alpha=rep(1.9,dim(as.matrix(design))[2]),
output_weights=rep(1,dim(as.matrix(design))[1]),method='post_sample',
initial_values=NULL,num_initial_values=3)

```


Arguments

design	a matrix of observed inputs.
observations	a vector of experimental data.
p_theta	an integer about the number of parameters, which should be specified by the user.
X	a matrix of the mean/trend discrepancy between the reality and math model.
have_trend	a bool value meaning whether we assume a mean/trend discrepancy function.
simul_type	an integer about the math model/simulator. If the simul_type is 0, it means we use RobustGaSP R package to build an emulator for emulation. If the simul_type is 1, it means the function of the math model is given by the user. When simul_type is 2 or 3, the mathematical model is the geophysical model for Kilauea Volcano. If the simul_type is 2, it means it is for the ascending mode InSAR data; if the simul_type is 3, it means it is for the descending mode InSAR data.
input_simul	an $D \times (p_x + p_{\theta})$ matrix of design for emulating the math model. It is only useful if simul_type is 0, meaning that we emulate the output of the math model.
output_simul	a D dimensional vector of the math model runs on the design (input_simul). It is only useful if simul_type is 0, meaning that we emulate the output of the math model.
simul_nug	a bool value meaning whether we have a nugget for emulating the math model/simulator. If the math model is stochastic, we often need a nugget. If simul_Nug is TRUE, it means we have a nugget for the emulator. If simul_Nug is FALSE, it means we do not have a nugget for the emulator.
loc_index_emulator	a vector of the location index from the ppgasp emulator to output. Only useful for vectorized output computer model emulated by the ppgasp emulator.
math_model	a function of the math model provided by the user. It is only useful if simul_type is 1, meaning that we know the math model and it can be computed fast. If the evaluation the math model is computationally slow, one should set simul_type to be 0 to emulate the math model.
theta_range	a $p_{\theta} \times 2$ matrix of the range of the calibration parameters. The first column is the lower bound and the second column is the upper bound. It should be specified by the user if the simul_type is 0.
sd_proposal	a vector of the standard deviation of the proposal distribution in MCMC.
S	number of posterior samples to run.
S_0	number of burn-in samples.
thinning	number of posterior samples to record.
discrepancy_type	characters about the type of the discrepancy. If it is 'no-discrepancy', it means no discrepancy function. If it is 'GaSP', it means the GaSP model for the discrepancy function. If it is 'S-GaSP', it means the S-GaSP model for the discrepancy function.

kernel_type	characters about the type of the discrepancy.type of kernel. matern_3_2 and matern_5_2 are Matern kernel with roughness parameter 3/2 and 5/2 respectively. pow_exp is power exponential kernel with roughness parameter alpha. If pow_exp is to be used, one needs to specify its roughness parameter alpha.
lambda_z	a vector value about how close the math model to the reality in squared distance when the S-GaSP model is used for modeling the discrepancy.
a	a scalar of the prior parameter.
b	a scalar of the prior parameter.
alpha	a numeric parameter for the roughness in the kernel.
output_weights	a vector of the weights of the outputs.
method	characters for method of parameter estimation. If it is 'post_sample', the posterior sampling will be used. If it is 'mle', the maximum likelihood estimator will be used.
initial_values	either a vector or a matrix of initial values of parameters. If posterior sampling method is used, it needs to be vector of the initial values of the calibration parameters. If an optimization method is used, it can be a matrix of the calibration parameters and kernel parameters (log inverse range parameters and the log nugget parameter) to be optimized numerically, where each row of the matrix contains a set of initial values.
num_initial_values	the number of initial values of the kernel parameters in optimization.

Value

rcalibration returns an S4 object of class rcalibration (see rcalibration-class).

Author(s)

Mengyang Gu [aut, cre]

Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

- A. O'Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.
- K. R. Anderson and M. P. Poland (2016), *Bayesian estimation of magma supply, storage, and eruption rates using a multiphysical volcano model: Kilauea volcano, 2000-2012.. Earth and Planetary Science Letters*, **447**, 161-171.
- K. R. Anderson and M. P. Poland (2017), *Abundant carbon in the mantle beneath Hawaii*. *Nature Geoscience*, **10**, 704-708.
- M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.
- M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.
- M. Gu (2018) *Jointly Robust Prior for Gaussian Stochastic Process in Emulation, Calibration and Variable Selection*. arXiv preprint arXiv:1804.09329.

Examples

```

library(RobustCalibration)

#-----
# an example with multiple local maximum of minimum in L2 loss
#-----

## the reality
test_funcn_eg1<-function(x){
  x*cos(3/2*x)+x
}

## obtain 25 data from the reality plus a noise
set.seed(1)
## 10 data points are very small, one may want to add more data
n=15
input=seq(0,5,5/(n-1))
input=as.matrix(input)

output=test_funcn_eg1(input)+rnorm(length(input),mean=0,sd=0.1)

num_obs=n=length(output)

## the math model
math_model_eg1<-function(x,theta){
  sin(theta*x)+x
}

##fit the S-GaSP model for the discrepancy
##one can choose the discrepancy_type to GaSP, S-GaSP or no discrepancy
##p_theta is the number of parameters to calibrate and user needs to specify
##one may also want to change the number of posterior samples by change S and S_0
p_theta=1
model_sgasp=rcalibration(design=input, observations=output, p_theta=p_theta,simul_type=1,
                        math_model=math_model_eg1,theta_range=matrix(c(0,3),1,2)
                        ,S=10000,S_0=2000,discrepancy_type='S-GaSP')

##if the acceptance rate is too low or too high, one can adjust sd_proposal, e.g.
#model_sgasp=rcalibration(design=input, observations=output, p_theta=1,simul_type=1,
#                          # sd_proposal=c(rep(0.02,p_theta),rep(0.2,dim(input)[2]),0.2)
#                          # math_model=math_model_eg1,theta_range=matrix(c(0,3),1,2)
#                          # ,S=10000,S_0=2000,discrepancy_type='S-GaSP')

##posterior samples of calibration parameter and value
plot(model_sgasp@post_sample[,1],type='l',xlab='num',ylab=expression(theta))
plot(model_sgasp@post_value,type='l',xlab='num',ylab='posterior value')

```

```

show(model_sgasp)

##one may want to fit a a model with an estimated baseline mean discrepancy by setting
##X=matrix(1,dim(input_stack)[1],1),have_trend=TRUE

model_sgasp_with_mean=rcalibration(design=input, observations=output, p_theta=1,simul_type=1,
                                  X=matrix(1,dim(input)[1],1),have_trend=TRUE,
                                  math_model=math_model_eg1,theta_range=matrix(c(0,3),1,2),
                                  S=10000,S_0=2000,discrepancy_type='S-GaSP')

show(model_sgasp_with_mean)

##posterior samples of calibration parameter and value
plot(model_sgasp_with_mean@post_sample[,1],type='l',xlab='num',ylab=expression(theta))
plot(model_sgasp_with_mean@post_value,type='l',xlab='num',ylab='posterior value')

## Not run:
#-----
# an example with multiple local maximum of minimum in L2 loss
# for combing the emulator
#-----

## the reality
test_funct_eg1<-function(x){
  x*cos(3/2*x)+x
}

## obtain 20 data from the reality plus a noise
set.seed(1)
n=20
input=seq(0,5,5/(n-1))
input=as.matrix(input)

output=test_funct_eg1(input)+rnorm(length(input),mean=0,sd=0.05)

num_obs=n=length(output)

## the math model
math_model_eg1<-function(x,theta){
  sin(theta*x)+x
}

##let's build an emulator for the case if the math model is too slow

# let's say we can only run the math model n_design times
n_design=80

design_simul=matrix(runif(n_design*2),n_design,2)
design_simul[,1]=5*design_simul[,1]  ##the first one is the observed input x

```

```

design_simul[,2]=3*design_simul[,2]  ##the second one is the calibration parameter

output_simul=math_model_eg1(design_simul[,1],design_simul[,2])

##this is a little slow compared with the previous model
model_sgasp_emulator=rcalibration(design=input, observations=output, p_theta=1,simul_type=0,
                                  input_simul=design_simul, output_simul=output_simul,
                                  theta_range=matrix(c(0,3),1,2),
                                  S=10000,S_0=2000,discrepancy_type='S-GaSP')

##now the output is a list
show(model_sgasp_emulator)

##here is the plot
plot(model_sgasp_emulator@post_sample[,1],type='l',xlab='num',ylab=expression(theta))
plot(model_sgasp_emulator@post_value,type='l',xlab='num',ylab='posterior value')

## End(Not run)

```

rcalibration-class *Robust Calibration class*

Description

S4 class for Robust rcalibration with or without the specification of the discrepancy model.

Objects from the Class

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

Slots

p_x: Object of class integer. The dimension of the observed inputs.
p_theta: Object of class integer. The calibration parameters.
num_obs: Object of class integer. The number of experimental observations.
input: Object of class matrix with dimension $n \times p_x$. The design of experiments.
output: Object of class vector with dimension $n \times 1$. The vector of the experimental observations.
X: Object of class matrix of with dimension $n \times q$. The mean/trend discrepancy basis function.
have_trend: Object of class bool to specify whether the mean/trend discrepancy is zero. "TRUE" means it has zero mean discrepancy and "FALSE" means the mean discrepancy is not zero.
q: Object of class integer. The number of basis functions of the mean/trend discrepancy.

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

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

alpha: Object of class `vector`. Each element is the parameter for the roughness for each input coordinate in the kernel.

theta_range: A matrix for the range of the calibration parameters.

lambda_z: Object of class `vector` about how close the math model to the reality in squared distance when the S-GaSP model is used for modeling the discrepancy.

S: Object of class `integer` about how many posterior samples to run.

S_0: Object of class `integer` about the number of burn-in samples.

prior_par: Object of class `vector` about prior parameters.

output_weights: Object of class `vector` about the weights of the experimental data.

sd_proposal: Object of class `vector` about the standard deviation of the proposal distribution.

discrepancy_type: Object of class `character` about the discrepancy. If it is 'no-discrepancy', it means no discrepancy function. If it is 'GaSP', it means the GaSP model for the discrepancy function. If it is 'S-GaSP', it means the S-GaSP model for the discrepancy function.

simul_type: Object of class `integer` about the math model/simulator. If the `simul_type` is 0, it means we use RobustGaSP R package to build an emulator for emulation. If the `simul_type` is 1, it means the function of the math model is given by the user. When `simul_type` is 2 or 3, the mathematical model is the geophysical model for Kilauea Volcano. If the `simul_type` is 2, it means it is for the ascending mode InSAR data; if the `simul_type` is 3, it means it is for the descending mode InSAR data.

emulator_rgasp: An S4 class of `rgasp` from the RobustGaSP package.

emulator_ppgasp: An S4 class of `ppgasp` from the RobustGaSP package.

post_sample: Object of class `matrix` for the posterior samples after burn-in.

post_value: Object of class `vector` for the posterior values after burn-in.

accept_S: Object of class `vector` for the number of proposed samples of the calibration parameters are accepted in MCMC. The first value is the number of proposed calibration parameters are accepted in MCMC. The second value is the number of proposed range and nugget parameters are accepted.

count_boundary: Object of class `vector` for the number of proposed samples of the calibration parameters are outside the range and they are rejected directly.

have_replicates: Object of class `bool` for having repeated experiments (replicates) or not.

num_replicates: Object of class `vector` for the number of replicates at each observable input.

thinning: Object of class `integer` for the ratio between the number of posterior samples and the number of samples to be recorded.

S_2_f: Object of class `numeric` for the variance of the field observations.

num_obs_all: Object of class `integer` for the total number of field observations.

method: Object of class `character` for posterior sampling or maximum likelihood estimation.

initial_values: Object of class `matrix` for initial starts of kernel parameters in maximum likelihood estimation.

param_est: Object of class vector for estimated range and nugget parameter in parameter estimation.

opt_value: Object of class numeric for optimized likelihood or loss function.

emulator_type: Object of class character for the type of emulator. 'rgasp' means scalar-valued emulator and 'ppgasp' means vectorized emulator.

loc_index_emulator: Object of class vector for location index to output in the ppgasp emulator for computer models with vectorized output.

Methods

show Prints the main slots of the object.

predict See [predict](#).

predict_separable_2dim See [predict_separable_2dim](#).

Author(s)

Mengyang Gu [aut, cre]
Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

A. O'Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.

M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.

M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.

M. Gu (2018) *Jointly Robust Prior for Gaussian Stochastic Process in Emulation, Calibration and Variable Selection*. arXiv preprint arXiv:1804.09329.

See Also

[rcalibration](#) for more details about how to create a rcalibration object.

rcalibration_MS

Setting up the robust Calibration model for multiple sources data

Description

Setting up the Calibration model for estimating the parameters via MCMC for multiple sources.

Usage

```
rcalibration_MS(design, observations, p_theta=NULL, index_theta=NULL,
  X=as.list(rep(0,length(design))),
  have_trend=rep(FALSE,length(design)),
  simul_type=rep(1, length(design)),
  input_simul=NULL, output_simul=NULL,
  simul_nug=rep(FALSE,length(design)),loc_index_emulator=NULL,
  math_model=NULL,
  theta_range=NULL, sd_proposal_theta=rep(0.05,p_theta),
  sd_proposal_cov_par=NULL,
  S=10000,S_0=2000, thinning=1,measurement_bias=FALSE,
  shared_design=NULL,have_measurement_bias_recorded=F,
  shared_X=0,have_shared_trend=FALSE,
  discrepancy_type=rep('S-GaSP',length(design)+measurement_bias),
  kernel_type=rep('matern_5_2',length(design)+measurement_bias),
  lambda_z=as.list(rep(NA,length(design)+measurement_bias)),
  a=NULL,b=NULL,alpha=NULL,
  output_weights=NULL)
```

Arguments

design	a list of observed inputs from multiple sources.
observations	a list of experimental data from multiple sources.
index_theta	a list of vectors for the index of calibration parameter contained in each source.
p_theta	an integer about the number of parameters, which should be specified by the user.
X	a list of matrices of the mean/trend discrepancy between the reality and math model for multiple sources.
have_trend	a vector of bool value meaning whether we assume a mean/trend discrepancy function.
simul_type	a vector of integer about the math model/simulator for multiple sources. If the simul_type is 0, it means we use RobustGaSP R package to build an emulator for emulation. If the simul_type is 1, it means the function of the math model is given by the user. When simul_type is 2 or 3, the mathematical model is the geophysical model for Kilauea Volcano. If the simul_type is 2, it means it is for the ascending mode InSAR data; if the simul_type is 3, it means it is for the descending mode InSAR data.
input_simul	a list of matrices, each having dimension $D \times (p_x+p_\theta)$ being the design for emulating the math model. It is only useful if the i th value of simul_type is 0 for the i th source, meaning that we emulate the output of the math model.
output_simul	a list of vectors, each having dimension $D \times 1$ being the math model outputs on the design (input_simul). It is only useful if the i th value of simul_type is 0 for the i th source, meaning that we emulate the output of the math model.

simul_nug	a vectors of bool values meaning whether we have a nugget for emulating the math model/simulator for this source. If the math model is stochastic, we often need a nugget. If simul_Nug is TRUE, it means we have a nugget for the emulator. If simul_Nug is FALSE, it means we do not have a nugget for the emulator.
loc_index_emulator	a list for location index to output in the ppgasp emulator for computer models with vectorized output.
math_model	a list of functions of the math models provided by the user for multiple sources. It is only useful if simul_type is 1, meaning that we know the math model and it can be computed fast. If the evaluation the math model is computationally slow, one should set simul_type to be 0 to emulate the math model.
theta_range	a p_theta x 2 matrix of the range of the calibration parameters. The first column is the lower bound and the second column is the upper bound. It should be specified by the user if the simul_type is 0.
sd_proposal_theta	a vector of the standard deviation of the proposal distribution for the calibration parameters in MCMC.
sd_proposal_cov_par	a list of vectors of the standard deviation of the proposal distribution for range and nugget parameters in MCMC for each source.
S	an integer about about how many posterior samples to run.
S_0	an integer about about the number of burn-in samples.
thinning	the ratio between the number of posterior samples and the number of recorded samples.
measurement_bias	containing measurement bias or not.
shared_design	A matrix for shared design across different sources of data used when measurement bias exists.
have_measurement_bias_recorded	A bool value whether we record measurement bias or not.
shared_X	A matrix of shared trend when measurement bias exists.
have_shared_trend	A bool value whether we have shared trend when measurement bias exist.
discrepancy_type	a vector of characters about the type of the discrepancy for each source. If it is 'no-discrepancy', it means no discrepancy function. If it is 'GaSP', it means the GaSP model for the discrepancy function. If it is 'S-GaSP', it means the S-GaSP model for the discrepancy function.
kernel_type	a vector of characters about the type of the discrepancy.type of kernel for each source. matern_3_2 and matern_5_2 are Matern kernel with roughness parameter 3/2 and 5/2 respectively. pow_exp is power exponential kernel with roughness parameter alpha. If pow_exp is to be used, one needs to specify its roughness parameter alpha.

lambda_z	a vector numeric values about how close the math model to the reality in squared distance when the S-GaSP model is used for modeling the discrepancy for each source.
a	a vector of the prior parameter for multiple sources.
b	a vector of the prior parameter for multiple sources.
alpha	a list of vectors of roughness parameters in the kernel for multiple sources.
output_weights	a list of vectors of the weights of the outputs for multiple sources.

Value

rcalibration_MS returns an S4 object of class rcalibration_MS (see rcalibration_MS-class).

Author(s)

Mengyang Gu [aut, cre]

Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

A. O’Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.

K. R. Anderson and M. P. Poland (2016), *Bayesian estimation of magma supply, storage, and eruption rates using a multiphysical volcano model: Kilauea volcano, 2000-2012.. Earth and Planetary Science Letters*, **447**, 161-171.

K. R. Anderson and M. P. Poland (2017), *Abundant carbon in the mantle beneath Hawaii. Nature Geoscience*, **10**, 704-708.

M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.

M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.

M. Gu (2018) *Jointly Robust Prior for Gaussian Stochastic Process in Emulation, Calibration and Variable Selection* . arXiv preprint arXiv:1804.09329.

Examples

```
#-----
# An example for calibrating mathematical models for data from multiple sources
#-----

library(RobustCalibration)

##reality
test_funct<-function(x){
  sin(pi*x/2)+2*cos(pi*x/2)
}
```

```
##math model from two sources
math_model_source_1<-function(x,theta){
  sin(theta*x)
}

math_model_source_2<-function(x,theta){
  cos(theta*x)
}

input1=seq(0,2,2/(10-1))
input2=seq(0,3,3/(15-1))
##
output1=test_func(input1)+rnorm(length(input1), sd=0.01)
output2=test_func(input2)+rnorm(length(input2), sd=0.02)

plot(input1, output1)
plot(input2, output2)

design=list()
design[[1]]=as.matrix(input1)
design[[2]]=as.matrix(input2)

observations=list()
observations[[1]]=output1
observations[[2]]=output2

p_theta=1

theta_range=matrix(0,p_theta,2)
theta_range[1,]=c(0, 8)
simul_type=c(1,1)

math_model=list()

math_model[[1]]=math_model_source_1
math_model[[2]]=math_model_source_2

## calibrating two mathematical models for these two sources
model_sgasp=rcalibration_MS(design=design, observations=observations, p_theta=1,
  simul_type=simul_type,math_model=math_model,
  theta_range=theta_range,
  S=10000,S_0=2000,
  discrepancy_type=rep('S-GaSP',length(design)))

plot(model_sgasp$post_theta[,1],type='l')
mean(model_sgasp$post_theta[,1])
```

rcalibration_MS-class *Robust Calibration for multiple sources class*

Description

S4 class for multiple sources Robust rcalibration with or without the specification of the discrepancy model.

Objects from the Class

Objects of this class are created and initialized with the function `rcalibration_MS` that computes the prediction after calibrating the mathematical models from multiple sources.

Slots

`num_sources`: Object of class integer. The number of sources.

`p_x`: Object of class vector. Each element is the dimension of the observed inputs in each source.

`p_theta`: Object of class integer. The number of calibration parameters.

`num_obs`: Object of class vector. Each element is the number of experimental observations of each source.

`index_theta`: Object of class list. The each element is a vector of the index of calibration parameters (theta) contained in each source.

`input`: Object of class list. Each element is a matrix of the design of experiments in each source with dimension $n_i \times p_{x,i}$, for $i=1, \dots, \text{num_sources}$.

`output`: Object of class list. Each element is a vector of the experimental observations in each source with dimension $n_i \times 1$, for $i=1, \dots, \text{num_sources}$.

`X`: Object of class list. Each element is a matrix of the mean/trend discrepancy basis function in each source with dimension $n_i \times q_i$, for $i=1, \dots, \text{num_sources}$.

`have_trend`: Object of class vector. Each element is a bool to specify whether the mean/trend discrepancy is zero in each source. "TRUE" means it has zero mean discrepancy and "FALSE" means the mean discrepancy is not zero.

`q`: Object of class vector. Each element is integer of the number of basis functions of the mean/trend discrepancy in each source.

`R0`: Object of class list. Each element is a list of matrices where the j -th matrix is an absolute difference matrix of the j -th input vector in each source.

`kernel_type`: Object of class vector. Each element is a character to specify the type of kernel to use in each source.

`alpha`: Object of class list. Each element is a vector of parameters for the roughness parameters in the kernel in each source.

`theta_range`: A matrix for the range of the calibration parameters.

- lambda_z:** Object of class vector. Each element is a numeric value about how close the math model to the reality in squared distance when the S-GaSP model is used for modeling the discrepancy in each source.
- S:** Object of class integer about how many posterior samples to run.
- S_0:** Object of class integer about the number of burn-in samples.
- prior_par:** Object of class list. Each element is a vector about prior parameters.
- output_weights:** Object of class list. Each element is a vector about the weights of the experimental data.
- sd_proposal_theta:** Object of class vector about the standard deviation of the proposal distribution for the calibration parameters.
- sd_proposal_cov_par:** Object of class list. Each element is a vector about the standard deviation of the proposal distribution for the calibration parameters in each source.
- discrepancy_type:** Object of class vector. Each element is a character about the type of the discrepancy in each source. If it is 'no-discrepancy', it means no discrepancy function. If it is 'GaSP', it means the GaSP model for the discrepancy function. If it is 'S-GaSP', it means the S-GaSP model for the discrepancy function.
- simul_type:** Object of class vector. Each element is an integer about the math model/simulator. If the simul_type is 0, it means we use RobustGaSP R package to build an emulator for emulation. If the simul_type is 1, it means the function of the math model is given by the user. When simul_type is 2 or 3, the mathematical model is the geophysical model for Kilauea Volcano. If the simul_type is 2, it means it is for the ascending mode InSAR data; if the simul_type is 3, it means it is for the descending mode InSAR data.
- emulator_rgasps:** Object of class list. Each element is an S4 class of rgasps from the Robust-GaSP package in each source.
- emulator_ppgasps:** Object of class list. Each element is an S4 class of ppgasps from the Robust-GaSP package in each source.
- post_theta:** Object of class matrix for the posterior samples of the calibration parameters after burn-in.
- post_individual_par:** Object of class list. Each element is a matrix for the posterior samples after burn-in in each source.
- post_value:** Object of class vector for the posterior values after burn-in.
- accept_S_theta:** Object of class numerical for the number of proposed samples of the calibration parameters are accepted in MCMC.
- accept_S_beta:** Object of class vector for the number of proposed samples of the range and nugget parameters in each source are accepted in MCMC.
- count_boundary:** Object of class vector for the number of proposed samples of the calibration parameters are outside the range and they are rejected directly.
- have_measurement_bias_recorded:** Object of class bool for whether measurement bias will be recorded or not.
- measurement_bias:** Object of class bool for whether measurement bias exists or not.
- post_delta:** Object of class matrix of samples of model discrepancy.
- post_measurement_bias:** Object of class list of samples of measurement_bias if measurement bias is chosen to be recorded.

thinning: Object of class integer for the ratio between the number of posterior samples and the number of samples to be recorded.

emulator_type: Object of class vector for the type of emulator for each source of data. 'rgasp' means scalar-valued emulator and 'ppgasp' means vectorized emulator.

loc_index_emulator: Object of class list for location index to output in the ppgasp emulator for computer models with vectorized output.

Methods

predict_MS See [predict_MS](#).

Author(s)

Mengyang Gu [aut, cre]

Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

A. O'Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.

M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.

M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.

M. Gu (2018) *Jointly Robust Prior for Gaussian Stochastic Process in Emulation, Calibration and Variable Selection*. arXiv preprint arXiv:1804.09329.

See Also

[rcalibration_MS](#) for more details about how to create a rcalibration_MS object.

show

Show an Robust Calibration object.

Description

Function to print the Robust Calibration model after the rcalibration class has been constructed.

Usage

```
## S4 method for signature 'rcalibration'
show(object)
```

Arguments

object an object of class rcalibration.

Author(s)

Mengyang Gu [aut, cre]

Maintainer: Mengyang Gu <mengyang@pstat.ucsb.edu>

References

A. O'Hagan and M. C. Kennedy (2001), *Bayesian calibration of computer models*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425-464.

M. Gu (2016), *Robust Uncertainty Quantification and Scalable Computation for Computer Models with Massive Output*, Ph.D. thesis., Duke University.

M. Gu and L. Wang (2017) *Scaled Gaussian Stochastic Process for Computer Model Calibration and Prediction*. arXiv preprint arXiv:1707.08215.

M. Gu (2018) *Jointly Robust Prior for Gaussian Stochastic Process in Emulation, Calibration and Variable Selection*. arXiv preprint arXiv:1804.09329.

Examples

```
##-----
##A simple example where the math model is not biased
##-----
## the reality
test_funct_eg1<-function(x){
  sin(pi/2*x)
}

## obtain 15 data from the reality plus a noise
set.seed(1)
## 10 data points are very small, one may want to add more data
n=15
input=seq(0,4,4/(n-1))
input=as.matrix(input)

output=test_funct_eg1(input)+rnorm(length(input),mean=0,sd=0.2)

## plot input and output
#plot(input,output)
#num_obs=n=length(output)

## the math model
math_model_eg1<-function(x,theta){
  sin(theta*x)
}

##fit the S-GaSP model for the discrepancy
##one can choose the discrepancy_type to GaSP, S-GaSP or no discrepancy
```

```

##p_theta is the number of parameters to calibrate and user needs to specify
##one may also want to change the number of posterior samples by change S and S_0
##one may change sd_proposal for the standard derivation of the proposal distribution
## one may also add a mean by setting X=... and have_trend=TRUE
model_sgasp=rcalibration(design=input, observations=output, p_theta=1,simul_type=1,
                        math_model=math_model_eg1,theta_range=matrix(c(0,3),1,2)
                        ,S=10000,S_0=2000,discrepancy_type='S-GaSP')

##posterior samples of calibration parameter and value
## the value is
plot(model_sgasp$post_sample[,1],type='l',xlab='num',ylab=expression(theta))
plot(model_sgasp$post_value,type='l',xlab='num',ylab='posterior value')

show(model_sgasp)

#-----
# an example with multiple local maximum of minimum in L2 loss
#-----

## the reality
test_funct_eg1<-function(x){
  x*cos(3/2*x)+x
}

## obtain 15 data from the reality plus a noise
set.seed(1)
n=15
input=seq(0,5,5/(n-1))
input=as.matrix(input)

output=test_funct_eg1(input)+rnorm(length(input),mean=0,sd=0.05)

num_obs=n=length(output)

## the math model
math_model_eg1<-function(x,theta){
  sin(theta*x)+x
}

## fit the S-GaSP model for the discrepancy

model_sgasp=rcalibration(design=input, observations=output, p_theta=1,simul_type=1,
                        math_model=math_model_eg1,theta_range=matrix(c(0,3),1,2),
                        discrepancy_type='S-GaSP')

## posterior samples

```



```
plot(model_sgasppost_sample[,1], type='l', xlab='num', ylab=expression(theta))  
show(model_sgasppost)
```

Index

- * **calibration**
 - RobustCalibration-package, 2
 - * **classes**
 - predictobj.rcalibration-class, 11
 - predictobj.rcalibration_MS-class, 12
 - rcalibration-class, 21
 - rcalibration_MS-class, 28
 - * **emulation**
 - RobustCalibration-package, 2
 - * **inverse problem**
 - RobustCalibration-package, 2
 - * **model misspecification**
 - RobustCalibration-package, 2
 - * **prediction**
 - RobustCalibration-package, 2
- predict, 5, 11, 23
- predict,rcalibration-method (predict), 5
- predict.rcalibration, 11, 12
- predict.rcalibration (predict), 5
- predict_MS, 12, 13, 13, 30
- predict_MS,rcalibration_MS-method (predict_MS), 13
- predict_MS.rcalibration_MS (predict_MS), 13
- predict_separable_2dim, 23
- predictobj.rcalibration (predictobj.rcalibration-class), 11
- predictobj.rcalibration-class, 11
- predictobj.rcalibration_MS (predictobj.rcalibration_MS-class), 12
- predictobj.rcalibration_MS-class, 12
- rcalibration, 16, 21, 23
- rcalibration-class, 21
- rcalibration-method (rcalibration), 16
- rcalibration_MS, 23, 28, 30
- rcalibration_MS-class, 28
- rcalibration_MS-method (rcalibration_MS), 23
- RobustCalibration (RobustCalibration-package), 2
- RobustCalibration-package, 2
- RobustGaSP, 3
- show, 30
- show,rcalibration-method (show), 30
- show.rcalibration (show), 30
- show.rcalibration-class (show), 30