Package 'SAVE'

July 2, 2014

Type Package

Title Bayesian emulation, calibration and validation of computer models

Version 0.9.3.9.2

Date 2014-05-22

Imports coda, DiceKriging, methods

Suggests MASS

Description Implements Bayesian statistical methodology for the analysis of complex computer models. It allows for the emulation, calibration, and validation of computer models,following methodology described in Bayarri et al 2007, Technometrics.

License GPL-2 | GPL-3

URL http://vangogh.fcjs.urjc.es/~jesus/SAVE,http://vangogh.fcjs.urjc.es/~jesus

BugReports

Repository CRAN

Date/Publication 2014-05-28 23:49:08

Author Jesus Palomo [aut, cre],Gonzalo Garcia-Donato [aut],Rui Paulo [aut],James Berger [ctb],Maria Jesus Bayarri [ctb],Jerome Sacks [ctb]

Maintainer Jesus Palomo <jesus.palomo@urjc.es>

NeedsCompilation yes

R topics documented:

SAVE-package *Simulator Analysis and Validation Engine*

Description

This package provides a statistical framework for the analysis of complex computer models, that is, for computer models that are expensive to run. Special emphasis is placed on key aspects like emulation (predicting the output of the code at new input points), calibration ('tuning' the model so that it matches reality) and validation (answering the question of whether the model adequately represents reality).

The methodology implemented in SAVE is Bayesian and is directly based on Bayarri et al (2007) but has roots also in the papers by Craig et al (1996), Kennedy and O'Hagan (2001) and Higdon et al (2004).

Details

Package:	SAVE
Type:	Package
Version:	0.9.3.3
Date:	2013-04-05
License:	GPL-2

Index:

Author(s)

Jesus Palomo, Rui Paulo and Gonzalo Garcia-Donato

Maintainer: <jesus.palomo@urjc.es>

References

Bayarri MJ, Berger JO, Paulo R, Sacks J, Cafeo JA, Cavendish J, Lin CH, Tu J (2007). A Framework for Validation of Computer Models. Technometrics, 49, 138-154.

Craig P, Goldstein M, Seheult A, Smith J (1996). Bayes linear strategies for history matching of hydrocarbon reservoirs. In JM Bernardo, JO Berger, AP Dawid, D Heckerman, AFM Smith (eds.), Bayesian Statistics 5. Oxford University Press: London. (with discussion).

Higdon D, Kennedy MC, Cavendish J, Cafeo J, Ryne RD (2004). Combining field data and computer simulations for calibration and prediction. SIAM Journal on Scientific Computing, 26, 448-466.

Kennedy MC, O Hagan A (2001). Bayesian calibration of computer models (with discussion). Journal of the Royal Statistical Society B, 63, 425-464.

See Also

SAVE, SAVE-class

Examples

```
## Not run:

#Validate the computer model in the Spotweld example (see Bayarri et al 2007 for details)
library(SAVE)

#############
# load data
#############

data(spotweldfield,package='SAVE')
data(spotweldmodel,package='SAVE')

##############
# create the SAVE object which describes the problem and
# compute the corresponding mle estimates
##############

gfsw <- SAVE(response.name="diameter", controllable.names=c("current", "load", "thickness"),
 calibration.names="tuning", field.data=spotweldfield,
 model.data=spotweldmodel, mean.formula=~1,
 bestguess=list(tuning=4.0))

##############
# obtain the posterior distribution of the unknown parameters
##############

gfsw <- bayesfit(object=gfsw, prior=c(uniform("tuning", upper=8, lower=0.8)),
 n.iter=20000, n.burnin=100, n.thin=2)

##############
# validate the computer model at chosen set of controllable
# inputs
###############

load <- c(4.0,5.3)
curr <- seq(from=20,to=30,length=20)
```

```
g <- c(1,2)

xnew <- as.data.frame(expand.grid(curr,load,g))
names(xnew)<-c("current","load","thickness")

valsw <- validate(object=gfsw,newdesign=xnew,n.burnin=100)

# summary of validation exercise:
summary(valsw)
# plot results
plot(valsw)

## End(Not run)
```

---

bayesfit *Bayesian fit*

Description

Bayesian analysis of a computer model: obtaining the posterior distribution of the unknown parameters in the statistical model of Bayarri et al. (2007).

Usage

```
## S4 method for signature 'SAVE'
bayesfit(object, prior, mcmcMultmle=1, prob.prop=0.5,
         method=2,n.iter, nMH=20, n.burnin=0, n.thin=1, verbose=FALSE, ...)
```

Arguments

object	An object of class SAVE normally produced by a call to function SAVE.
prior	The prior distribution assumed for the calibration parameters. This should be specified concatenating, c, calls to functions uniform and/or normal. See details below.
mcmcMultmle	A factor that is used to specify the prior for lambdaF and lambdaM (the precision parameters in the field and the bias, respectively). See details below.
prob.prop	The probability of proposing from the prior in the Metropolis-Hastings algorithm. See details below.
method	Method implemented in the Gibbs sampling. See details below.
n.iter	Number of total simulations. See details below.
nMH	Number of Metropolis-Hastings steps. See details below.
n.burnin	The number of iterations at the beginning of the MCMC that are thrown away. See details below.
n.thin	The thinning to be applied to the resulting MCMC sample. See details below.
verbose	A logical value indicating the level of output as the function runs.
...	Extra arguments to be passed to the function (still not implemented).

Details

The parameters in the statistical model which are treated as unknown are lambdaB (the precision of the bias, also called discrepancy term); lambdaF (the precision of the error) and the vector of calibration inputs. The function `bayesfit` provides a sample from the posterior distribution of these parameters using a particular MCMC strategy. This function depends on two types of arguments detailed below: those defining the prior used, and those providing details for the MCMC sampling.

About the prior: the prior for lambdaB and lambdaF is the product of two independent exponential densities with the means being equal to the corresponding maximum likelihood estimates times the factor specified in `mcmcMultmle`. Notice that the prior variance increases quadratically with this factor, and the prior becomes less informative as the factor increases.

The prior for each calibration parameter can be either a uniform distribution or a truncated normal and should be specified concatenating calls to `uniform` and/or `normal`. For example, in a problem with calibration parameters named "delta1" and "shift", a uniform(0,1) prior for "delta1" and for "shift" a normal density with mean 2 and standard deviation 1 truncated to the interval (0,3), the prior should be specified as

```
prior=c(uniform(var.name="delta1", lower=0, upper=1),
normal(var.name="shift", mean=2, sd=1, lower=0, upper=3))
```

About the MCMC. The algorithm implemented is based on a Gibbs sampling scheme. If `method=2` then the emulator of the computer model and the bias are integrated out (analytically) and only the full conditionals for lambdaB, lambdaF and calibration parameters are sampled from. Else, if `method=1` the computer model and the bias are part of the sampling scheme. The calibration parameters are sampled from their full conditional distribution using a Metropolis-Hastings algorithm with candidate samples proposed from a mixture of the prior specified and a uniform centered on the last sampled value. Here, the probability of a proposal coming from the prior is set by `prob.prop`. The Metropolis-Hastings algorithm is run `nMH` times before each sample is accepted. The default and preferred method is `method=2`.

The MCMC is run a total of `n.iter` iterations, of which the first `n.burnin` are discarded. The remaining samples are thinned using the number specified in `n.thin`.

Value

`SAVE` returns a copy of the `SAVE` object used as argument to the function, but with the following slots filled (or replaced if they where not empty)

`method`**:** the value given to `method`.

`mcmcMultmle`**:** the value given to `mcmcMultmle`.

`n.iter`**:** the value given to `n.iter`.

`nMH`**:** The value given to `nMH`.

`mcmcsample`**:** A named `matrix` with the simulated samples from the posterior distribution.

`bayesfitcall`**:** The `call` to `bayesfit`.

Author(s)

Jesus Palomo, Rui Paulo and Gonzalo Garcia-Donato

References

Bayarri MJ, Berger JO, Paulo R, Sacks J, Cafeo JA, Cavendish J, Lin CH, Tu J (2007). A Framework for Validation of Computer Models. Technometrics, 49, 138-154.

Craig P, Goldstein M, Seheult A, Smith J (1996). Bayes linear strategies for history matching of hydrocarbon reservoirs. In JM Bernardo, JO Berger, AP Dawid, D Heckerman, AFM Smith (eds.), Bayesian Statistics 5. Oxford University Press: London. (with discussion).

Higdon D, Kennedy MC, Cavendish J, Cafeo J, Ryne RD (2004). Combining field data and computer simulations for calibration and prediction. SIAM Journal on Scientific Computing, 26, 448-466.

Kennedy MC, O Hagan A (2001). Bayesian calibration of computer models (with discussion). Journal of the Royal Statistical Society B, 63, 425-464.

Roustant O., Ginsbourger D. and Deville Y. (2012). DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. Journal of Statistical Software, 51(1), 1-55.

See Also

plot, predictreality, validate

Examples

```
## Not run:
library(SAVE)

#############
# load data
#############

data(spotweldfield,package='SAVE')
data(spotweldmodel,package='SAVE')

##############
# create the SAVE object which describes the problem and
# compute the corresponding mle estimates
##############

gfsw <- SAVE(response.name="diameter", controllable.names=c("current", "load", "thickness"),
 calibration.names="tuning", field.data=spotweldfield,
 model.data=spotweldmodel, mean.formula=~1,
 bestguess=list(tuning=4.0))

##############
# obtain the posterior distribution of the unknown parameters
##############

gfsw <- bayesfit(object=gfsw, prior=c(uniform("tuning", upper=8, lower=0.8)),
 n.iter=20000, n.burnin=100, n.thin=2)

# summary of the results
```

```
summary(gfsw)

## End(Not run)
```

`normal,uniform` *Auxiliary functions.*

Description

Auxiliary functions.

Usage

```
uniform(var.name, lower, upper)

normal(var.name, mean, sd, lower, upper)
```

Arguments

`var.name`	The name of the variable whose prior distribution is being specified.
`lower`	The lower prior bound for `var.name`.
`upper`	The upper prior bound for `var.name`.
`mean`	The prior mean for `var.name`.
`sd`	The prior standard deviation for `var.name`.

Details

These functions are intended as a user-friendly way of specifying the type of priors assumed for the calibration parameters in the function `bayesfit`.

For example, in a problem with calibration parameters named "delta1" and "shift", a uniform(0,1) prior for "delta1" and for "shift" a normal density with mean 2 and standard deviation 1 truncated to the interval (0,3), the prior should be specified as

```
prior=c(uniform(var.name="delta1", lower=0, upper=1),
normal(var.name="shift", mean=2, sd=1, lower=0, upper=3))
```

plot *Plots for an object of class* SAVE

Description

Plots are provided to summarize graphically the Bayesian analysis of a computer model.

Usage

```
## S4 method for signature 'SAVE'
plot(x, option = "trace", ...)
```

Arguments

x	An object of class SAVE
option	One of "trace", "calibration" or "precision"(see details)
...	Additional graphical parameters to be passed

Details

Three different plots are implemented. If option="trace" this function returns the trace plots of the simulated chains. This plot is useful for assessing the convergence of the sampling method. If option="calibration" this function plots an histogram of the sample obtained from the posterior distribution of the calibration parameters and a line representing the prior assumed. If option="precision" the histograms and priors correspond to the precision parameters.

Author(s)

Jesus Palomo, Rui Paulo and Gonzalo Garcia-Donato.

Examples

```
## Not run:
library(SAVE)

#############
# load data
#############

data(spotweldfield,package='SAVE')
data(spotweldmodel,package='SAVE')

##############
# create the SAVE object which describes the problem and
# compute the corresponding mle estimates
##############

gfsw <- SAVE(response.name="diameter", controllable.names=c("current", "load", "thickness"),
```

```
 calibration.names="tuning", field.data=spotweldfield,
 model.data=spotweldmodel, mean.formula=~1,
 bestguess=list(tuning=4.0))

##############
# obtain the posterior distribution of the unknown parameters
##############

gfsw <- bayesfit(object=gfsw, prior=c(uniform("tuning", upper=8, lower=0.8)),
 n.iter=20000, n.burnin=100, n.thin=2)

#A trace plot of the chains
plot(gfsw, option="trace")
#The histogram of the posterior density of calibration parameters
plot(gfsw, option="calibration")
#The histogram of the posterior density of precision parameters
plot(gfsw, option="precision")


## End(Not run)
```

`plot.predictreality.SAVE` *A function for plotting summaries of an object of class* `predictreality.SAVE`.

Description

Two different plots to summarize graphically the results in an object of class `predictreality.SAVE`.

Usage

```
## S4 method for signature 'predictreality.SAVE'
plot(x, option = "trace", ...)
```

Arguments

`x`	An object of class `predictreality.SAVE`
`option`	One of "biascorr" or "biasfun" (see details)
`...`	Additional graphical parameters to be passed

Details

If `option`="biascorr" this function returns a plot with point predictions and 95% tolerance bounds of reality at the given set of controllable inputs. If `option`="biasfun" the plot represents the estimated bias and 95% credible bounds.

Author(s)

Jesus Palomo, Rui Paulo and Gonzalo Garcia-Donato.

Examples

```
## Not run:
library(SAVE)

#############
# load data
#############

data(spotweldfield,package='SAVE')
data(spotweldmodel,package='SAVE')

##############
# create the SAVE object which describes the problem and
# compute the corresponding mle estimates
##############

gfsw <- SAVE(response.name="diameter", controllable.names=c("current", "load", "thickness"),
 calibration.names="tuning", field.data=spotweldfield,
 model.data=spotweldmodel, mean.formula=~1,
 bestguess=list(tuning=4.0))

##############
# obtain the posterior distribution of the unknown parameters
##############

gfsw <- bayesfit(object=gfsw, prior=c(uniform("tuning", upper=8, lower=0.8)),
 n.iter=20000, n.burnin=100, n.thin=2)

#########
# bias-corrected prediction at a set of inputs
# using predictreality
##########

load <- c(4.0,5.3)
curr <- seq(from=20,to=30,length=20)
g <- c(1,2)

xnew<- expand.grid(current = curr, load = load, thickness=g)

# Obtain samples
prsw <- predictreality(object=gfsw, newdesign=xnew, tol=1.E-12)

#Plot the results:
#Represent reality and tolerance bounds:
plot(prsw, option="biascorr")
#Represent bias and tolerance bounds:
plot(prsw, option="biasfun")
```

```
## End(Not run)
```

---

predictcode *Predict values of the computer model at new input points*

Description

The emulator of the computer model fitted by SAVE is used to predict values of the model at new input points.

Usage

```
## S4 method for signature 'SAVE'
predictcode(object, newdesign, n.iter=1000, sampledraws=T, tol=1e-10, verbose=FALSE)

## S4 method for signature 'predictcode.SAVE'
summary(object)

## S4 method for signature 'summary.predictcode.SAVE'
show(object)

## S4 method for signature 'predictcode.SAVE'
plot(x, ...)
```

Arguments

object	An object of the corresponding signature.
newdesign	A named matrix containing the points (calibration and controllable inputs) where predictions are to be performed. Column names should contain both the object@controllablenames and object@calibrationnames
n.iter	The number of simulations that are to be drawn from the emulator (see details below)
sampledraws	If TRUE a sample of size n.iter is obtained from the emulator. If FALSE only the covariance matrix and the mean of the emulator are returned.
tol	The tolerance in the Cholesky decomposition
verbose	A logical value indicating the level of output as the function runs.
...	Extra arguments to be passed to the function (still not implemented).
x	An object of class predictcode.SAVE

Details

The emulator of the computer model fitted by SAVE evaluated at the new input points specified in newdesign is a multivariate normal. Then predictcode computes the mean, the covariance matrix and, if sampledraws=TRUE, a simulated sample of size n.iter from this multivariate normal. A pivotal Cholesky decomposition algorithm is used in the simulation of the samples and tol is a tolerance parameter in this algorithm.

The object created can be explored with the functions plot and summary. The first function plots a graphic with the mean and 95% tolerance bounds of the emulator at each of the new input points. Furthermore, summary prints a matrix with the mean of the emulator at each new input point, the associated standard deviation, and 95% tolerance bounds.

Value

Returns an S4 object of the class predictcode.SAVE that contains the following slots:

newdesign	A copy of the design.
samples	The matrix that contains the simulations (see details).
mle	A copy of the maximum likelihood estimate object@mle.
predictcodecall	The call to this function.
modelmean	The mean of the emulator (see details) at the new design newdesign.
covmat	The covariance matrix of the emulator (see details) at the new design newdesign.

Author(s)

Jesus Palomo, Rui Paulo and Gonzalo Garcia-Donato.

See Also

SAVE

Examples

```
## Not run:
library(SAVE)

#############
# load data
#############

data(spotweldfield,package='SAVE')
data(spotweldmodel,package='SAVE')

##############
# create the SAVE object which describes the problem and
# compute the corresponding mle estimates
##############

gfsw <- SAVE(response.name="diameter", controllable.names=c("current", "load", "thickness"),
```

```
 calibration.names="tuning", field.data=spotweldfield,
 model.data=spotweldmodel, mean.formula=~1,
 bestguess=list(tuning=4.0))

##########
# emulate the output of the model using predictcode
##########

# construct design at which to emulate the model
u <- 3.2
load <- c(4.0,5.3)
curr <- seq(from=20,to=30,length=20)
g <- c(1,2)

xnewpure <- expand.grid(curr,load,g)
xnewpure <- cbind(xnewpure,rep(u,dim(xnewpure)[1]))
names(xnewpure) <- c("current","load","thickness","tuning")
xnewpure <- as.data.frame(xnewpure)

pcsw<- predictcode(object=gfsw, newdesign=xnewpure, n.iter=20000, tol=1.E-12)

#A summary of the emulation:
summary(pcsw)

#A plot of the emulation
plot(pcsw)

## End(Not run)
```

`predictreality` *Predict values of reality at new input points*

Description

The emulator of the computer model and the Bayesian fit are used to produce samples from the posterior predictive distribution of the computer model and bias function evaluated at the new input points. Then, bias-corrected predictions of the response (reality) are produced by adding these two samples (model+bias).

Usage

```
## S4 method for signature 'SAVE'
predictreality(object, newdesign, n.burnin=0, n.thin=1, tol=1E-10, verbose=FALSE, ...)

## S4 method for signature 'predictreality.SAVE'
summary(object)

## S4 method for signature 'summary.predictreality.SAVE'
show(object)
```

Arguments

`object`	An object of the corresponding signature.
`newdesign`	A named matrix containing the points (controllable inputs) where predictions are to be performed. Column names should contain the `object@controllablenames`. This parameter should be set to NULL in the situation with constant controllable inputs.
`n.burnin`	The burnin to be applied (see details below).
`n.thin`	The thinin to be applied (see details below).
`tol`	The tolerance in the Cholesky decomposition.
`verbose`	A `logical` value indicating the level of output as the function runs.
`...`	Extra arguments to be passed to the function (still not implemented).

Details

Draws from the posterior predictive distribution of the computer model and bias at a given set of controllable inputs are simulated using the MCMC sample from the posterior distribution of the parameters of the model stored in `object@mcmcsample`. This sample can be thinned by `n.thin` and/or the first `n.burnin` draws can be discarded.

A preliminary analysis of the resulting sample can be performed with `summary` which provides point estimates and tolerance bounds of the predictions.

Value

Returns an S4 object of class `predictreality.SAVE` with the following slots:

`modelpred`	A list with the simulations from the posterior distribution of the computer model output evaluated at the new design
`biaspred`	A matrix with the simulations from the posterior distribution of the bias function evaluated at the new design.
`newdesign`	A copy of the design given as argument.
`predictrealitycall`	The call to the function.

Author(s)

Jesus Palomo, Rui Paulo and Gonzalo Garcia-Donato.

See Also

`validate`

Examples

```
## Not run:
library(SAVE)

#############
# load data
#############

data(spotweldfield,package='SAVE')
data(spotweldmodel,package='SAVE')

##############
# create the SAVE object which describes the problem and
# compute the corresponding mle estimates
##############

gfsw <- SAVE(response.name="diameter", controllable.names=c("current", "load", "thickness"),
 calibration.names="tuning", field.data=spotweldfield,
 model.data=spotweldmodel, mean.formula=~1,
 bestguess=list(tuning=4.0))

# summary of the results

summary(gfsw)

##############
# obtain the posterior distribution of the unknown parameters
##############

gfsw <- bayesfit(object=gfsw, prior=c(uniform("tuning", upper=8, lower=0.8)),
 n.iter=20000, n.burnin=100, n.thin=2)

#########
# bias-corrected prediction at a set of inputs
# using predictreality
##########

load <- c(4.0,5.3)
curr <- seq(from=20,to=30,length=20)
g <- c(1,2)

xnew<- expand.grid(current = curr, load = load, thickness=g)

# Obtain samples
prsw <- predictreality(object=gfsw, newdesign=xnew, tol=1.E-12)

#Summarize the results:
summary(prsw)


## End(Not run)
```

SAVE *Setting up the analysis of a computer model*

Description

Setting up the SAVE methodology: construction of an emulator of the computer model and estimation of the parameters of the Gaussian process for the bias function.

Usage

```
SAVE(response.name=NULL, controllable.names=NULL, calibration.names=NULL,
field.data=NULL, model.data=NULL, mean.formula=~1, bestguess=NULL,
kriging.controls=SAVE.controls(), verbose=FALSE)

## S4 method for signature 'SAVE'
show(object)

## S4 method for signature 'SAVE'
summary(object)

## S4 method for signature 'summary.SAVE'
show(object)
```

Arguments

`response.name`	A `character` object with the name of the response variable.
`controllable.names`	Either a `character` object with the names of the controllable inputs or set to NULL if in the field experiment the controllable inputs (if any) have not been varied (the analysis has constant controllable inputs).
`calibration.names`	A `character` object with the names of the calibration inputs. Use NULL if the model does not depend on any calibration inputs.
`field.data`	A `data.frame` with the observations coming from the field experiment. The column names of this `data.frame` should contain the `controllable.names` and the `response.name`.
`model.data`	A `data.frame` with the observations coming from runs of the computer model. The column names of this `data.frame` should contain the `controllable.names`, the `calibration.names` and the `response.name`.
`mean.formula`	A `formula` specifying the mean function of the Gaussian Process approximation to the output of the computer model (the emulator). Can only involve terms in `controllable.names`.
`bestguess`	A named `list` specifying the best guess for the calibration parameters.

`kriging.controls`
: A named `list` specifying the parameters to be passed to the kriging process at the stage I parameters (the ones invoved in the construction of the emulator) estimation step.

`verbose`
: A `logical` value indicating the level of output as the function runs.

`object`
: An object of the corresponding signature.

Details

Based on computer model runs, `SAVE` fits an approximation of the model output, usually called an emulator. The emulator is constructed using the Gaussian process response technique (GASP), described in more detail in `SAVE-class`. Further, at this stage an estimation of the parameters of the Gaussian process specifying the bias function (difference between field observations and computer model outputs) is also performed. Some of the calculations are done using the package `DiceKriging`

Value

`SAVE` returns an S4 object of class `SAVE` (see `SAVE-class`).

Author(s)

Jesus Palomo, Rui Paulo and Gonzalo Garcia-Donato

References

Bayarri MJ, Berger JO, Paulo R, Sacks J, Cafeo JA, Cavendish J, Lin CH, Tu J (2007). A Framework for Validation of Computer Models. Technometrics, 49, 138-154.

Craig P, Goldstein M, Seheult A, Smith J (1996). Bayes linear strategies for history matching of hydrocarbon reservoirs. In JM Bernardo, JO Berger, AP Dawid, D Heckerman, AFM Smith (eds.), Bayesian Statistics 5. Oxford University Press: London. (with discussion).

Higdon D, Kennedy MC, Cavendish J, Cafeo J, Ryne RD (2004). Combining field data and computer simulations for calibration and prediction. SIAM Journal on Scientific Computing, 26, 448-466.

Kennedy MC, O Hagan A (2001). Bayesian calibration of computer models (with discussion). Journal of the Royal Statistical Society B, 63, 425-464.

Roustant O., Ginsbourger D. and Deville Y. (2012). DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. Journal of Statistical Software, 51(1), 1-55.

See Also

`predictcode`, `bayesfit`

Examples

```
## Not run:
library(SAVE)

#############
# load data
#############

data(spotweldfield,package='SAVE')
data(spotweldmodel,package='SAVE')

##############
# create the SAVE object which describes the problem and
# compute the corresponding mle estimates
##############

gfsw <- SAVE(response.name="diameter", controllable.names=c("current", "load", "thickness"),
 calibration.names="tuning", field.data=spotweldfield,
 model.data=spotweldmodel, mean.formula=~1,
 bestguess=list(tuning=4.0))

# summary of the results

summary(gfsw)

## End(Not run)
```

SAVE-class *SAVE class*

Description

S4 class for Statistical Analysis and Validation Engine.

Objects from the Class

Objects of this class are created and initialized with the function SAVE that computes the calculations needed for setting up the analysis. These can be completed with the function bayesfit that performs the Bayesian analysis in the SAVE methodology

Slots

responsename: Object of class character. The response name.

controllablenames: Object of class character. The names of the controllable inputs.

calibrationnames: Object of class character. The names of the calibration inputs.

constant.controllables: Object of class logical. Controls whether or not the analysis has constant controllable inputs.

`df`: Object of class `matrix`. The field design once the replicates (if any) have been removed.

`dm`: Object of class `matrix`. The model design.

`ym`: Object of class `numeric`. Model response associated with dm.

`yf`: Object of class `numeric`. The field observations.

`meanformula`: Object of class `formula`. The formula that specifies the mean function of the emulator of the computer model.

`mle`: The maximum likelihood estimates. This is a `list` with three components

- `thetaM`: A `numeric` vector containing the estimate of the parameters specifying the covariance structure of the emulator of the computer model. This covariance function has precision lambdaM and a separable correlation function with k(x,y)=exp(-betaM*h^alphaM) where h=abs(x-y). The vector `thetaM` is organized as follows: (lambdaM, betaM, alphaM), where betaM and alphaM are named vectors.
- `thetaL`: The `numeric` vector of regression coefficients associated with the mean function of the emulator of the computer model
- `thetaF`: A `numeric` vector organized as (lambdaB, betaB, alphaB, lambdaF) containing the estimates of lambdaF, the precision of the field measurement error, and of the parameters specifying the Gaussian process prior of the bias function. The covariance function and the parameters follow the same structure as that described for `thetaM`

`bestguess`: A `numeric` vector containing the best guess (provided in the call) for the calibration inputs.

`xm`: The model matrix corresponding to the evaluation of the `meanformula` at `dm`.

`xf`: The model matrix corresponding to the evaluation of the `meanformula` at `df`.

`prior`: Description of the prior used (empty if if `bayesfit` is not run).

`method`: A `numeric` object with possible values 1 and 2. Two different MCMC methods have been implemented. If `method=2` then the computer model and bias are integrated out (analytically) before sampling the calibration parameters. If `method=1` then the calibration parameters is sampled from the full conditional. (Empty if if `bayesfit` is not run).

`mcmcMultmle`: A positive `numeric` object. Priors for the precisions (lambdaM and lambdaB) are exponential distributions centered at the corresponding mle multiplied by `mcmcMultmle`. (Empty if if `bayesfit` is not run).

`mcmcsample`: A `matrix` with the result of the MCMC sampling after the burnin and thinin has been applied. (Empty if if `bayesfit` is not run).

`wd`: A `character` with the name of the working directory.

`call`: The `call` to `SAVE` function to create the object.

`bayesfitcall`: The `call` to `bayesfit`. (Empty if if `bayesfit` is not run).

Methods

summary A summary of the object created.

show Prints the summary of the object.

plot See `plot`.

predictcode See `predictcode`.

bayesfit See `bayesfit`.

predictreality See `predictreality`.

validate See `validate`.

Author(s)

J. Palomo, R. Paulo and G. Garcia-Donato

See Also

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

Spotweld *Spotweld data*

Description

These data relate to an engineering application where two metal sheets are compressed by water-cooled copper electrodes under an applied load L and the resistance spot welding is measure. Two data sets are provided, one containing observations from the real experiment (spotweldfield) and another one containing observations from the simulator (i.e. the computer model, spotweldmodel).

Details about this data set can be found in Bayarri et al (2007).

Usage

```
data(spotweldfield)

data(spotweldmodel)
```

Format

Two data frame with observations on the following variables.

load Controllable input (Load)

current Controllable input (The direct current of magnitude supplied to the sheets)

thickness Controllable input

diameter Response variable (Resistance)

tuning Calibration input (hence only in spotweldmodel)

References

Bayarri MJ, Berger JO, Paulo R, Sacks J, Cafeo JA, Cavendish J, Lin CH, Tu J (2007). A Framework for Validation of Computer Models. Technometrics, 49, 138-154.

Synthetic *Synthetic data*

Description

Synthetic data from a real experiment (`synthfield`) and another one containing observations from the simulator (i.e. the computer model, `synthmodel`).

Usage

```
data(synthfield)

data(synthmodel)
```

Format

Two data frame with observations on the following variables.

- `y` Response
- `x` Controllable input
- `v` Calibration variable (only in `synthmodel`)

validate *Validation of a computer model*

Description

Assessing the validity of a computer model at a given set of controllable inputs

Usage

```
## S4 method for signature 'SAVE'
validate(object, newdesign, calibration.value="mean", prob=0.90,
  n.burnin=0, n.thin=1, tol=1E-10, ...)

## S4 method for signature 'validate.SAVE'
summary(object)

## S4 method for signature 'summary.validate.SAVE'
show(object)

## S4 method for signature 'validate.SAVE'
plot(x, ...)
```

Arguments

object	An object of corresponding signature.
x	An object of class validate.SAVE.
newdesign	A named matrix containing the points (controllable inputs) where predictions are to be performed. Column names should contain the object@controllablenames. This parameter should be set to NULL in the situation with constant controllable inputs.
calibration.value	Either the summary of the posterior distribution of the calibration parameters which is taken as the true value for the computer model (possible values are "mean" and "median") or a named data frame with numerical values to be used for the calibration parameters.
prob	The probability used for the representation of the credible intervals.
n.burnin	The burn-in to be applied (see details below).
n.thin	The thinning to be applied (see details below).
tol	The tolerance in the Cholesky decomposition.
...	Additional arguments to be passed, still not implemented.

Details

Following the framework for the analysis of computer models by Bayarri et al (2007), validation of a computer model translates to the question: is the computer model producing results that are accurate enough for its intended use?

Answering this question implies the comparison of the responses of the computer model at the 'true' value of the calibration parameters (calibration.value) with reality. This comparison should be performed at the set of controllable inputs that are of interest for the researcher (which in turn represent the 'intended use' for the computer model) and that are passed as the argument newdesign to the function.

For this comparison to be performed, validation returns a matrix (@validate) containing (for each one of the input values in the rows of this matrix) the prediction of reality (column called "bias.corrected") jointly with the estimate of the model response, both with corresponding accompanying tolerance bounds (columns called "tau.bc" and "tau.pm" respectively). These measures should be interpreted as:

Prob(|estimate-real value|<tau)=prob

Also, the discrepancy between computer model and reality can be assessed through the estimated bias (column called "bias") with the associated 100prob% credible interval (columns called "bias.Lower" and "bias.Upper").

In the calculations, the simulated sample from the posterior distribution contained in object@mcmsample is used. It is possible to discard some of the samples in this matrix by not considering the first n.burnin and/or thinning by n.thin.

The results can be conveniently visualized with the functions summary and plot.

Value

Returns an S4 object of class `validate.SAVE` with the following slots:

`bayesfitcall`: The `call` to `bayesfit`.

`call`: The original `call` to `SAVE` function.

`newdesign`: A copy of the design given.

`validate`: A matrix with the results of the validation analysis (see details below).

`validatecall`: The call to the `validate` function.

Author(s)

Jesus Palomo, Rui Paulo and Gonzalo Garcia-Donato

References

Bayarri MJ, Berger JO, Paulo R, Sacks J, Cafeo JA, Cavendish J, Lin CH, Tu J (2007). A Framework for Validation of Computer Models. Technometrics, 49, 138-154.

Examples

```
## Not run:
library(SAVE)

#############
# load data
#############

data(spotweldfield,package='SAVE')
data(spotweldmodel,package='SAVE')

##############
# create the SAVE object which describes the problem and
# compute the corresponding mle estimates
##############

gfsw <- SAVE(response.name="diameter", controllable.names=c("current", "load", "thickness"),
 calibration.names="tuning", field.data=spotweldfield,
 model.data=spotweldmodel, mean.formula=~1,
 bestguess=list(tuning=4.0))

##############
# obtain the posterior distribution of the unknown parameters
##############

gfsw <- bayesfit(object=gfsw, prior=c(uniform("tuning", upper=8, lower=0.8)),
 n.iter=20000, n.burnin=100, n.thin=2)

##############
# validate the computer model at chosen set of controllable
# inputs
```

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

load <- c(4.0,5.3)
curr <- seq(from=20,to=30,length=20)
g <- c(1,2)

xnew <- expand.grid(current = curr, load = load, thickness=g)

valsw <- validate(object=gfsw,newdesign=xnew,n.burnin=100)

# summary of results
summary(valsw)
# plot results
plot(valsw)



## End(Not run)
```

Index