Package 'stilt'

April 9, 2025

Type Package

Title Separable Gaussian Process Interpolation (Emulation)

Version 1.3.1

Date 2025-04-09

Maintainer Kelsey Ruckert <datamgmt@scrim.psu.edu>

Description Functions for simplified emulation of time series computer model output in model parameter space using Gaussian processes. Stilt can be used more generally for Kriging of spatio-temporal fields. There are functions to predict at new parameter settings, to test the emulator using cross-validation (which includes information on 95% confidence interval empirical coverage), and to produce contour plots over 2D slices in model parameter space.

License GPL-3

Depends R (>= 3.5.0)

Imports fields

LazyLoad no

NeedsCompilation no

Repository CRAN

Date/Publication 2025-04-09 16:40:02 UTC

RoxygenNote 6.0.1

Author Roman Olson [aut],
Won Chang [aut],
Klaus Keller [aut],
Murali Haran [aut],
Kelsey Ruckert [cre]

Contents

`stilt-package` *Separable Gaussian Process Emulator*

Description

'Stilt' is used for interpolating multivariate data in multivariate space. It is tailored to the case of interpolating ("emulating") regularly spaced time-series of computer model output between the model input parameters, but can be also used more broadly (e.g., for output in the form of regularly-spaced spatial 1D transects, or for interpolating time-series between locations on the Earth's surface). The interpolation technique relies on a separable Gaussian Process emulator. Package consists of functions to fit the Gaussian Process emulator, to interpolate to (predict at) a given model input setting, and to validate the emulator using cross-validation. In addition, a function is provided to plot model response surface as generated by the emulator as a function of two model parameters. The package works both on one and multi-parameter ensembles, with an exception of the 'rsurface.plot' function, which by definition is restricted to 2+ parameter ensembles. The emulator is restricted to multivariate model output and is not designed to work on interpolating a single scalar.

Details

Package:	stilt
Type:	Package
Version:	1.3.1
Date:	2025-04-09
License:	GPL-3
LazyLoad:	no

Disclaimer

This package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. In addition, the authors maintain no responsibility for any possible code errors or bugs.

Summary of main functions

emulator Fits a separable Gaussian Process emulator to ensemble model output

predict.emul Predicts using a Gaussian Process emulator

rsurface.plot Plots a 2D model response surface

test.csv Cross-validates an emulator by removing one or many model runs, training the emulator on the remaining runs, and then predicting at the withheld model runs

test.all Tests an emulator using leave-one-out cross validation

sep.cov Constructs time and parameter covariance matrices

Summary of datasets

Data.1D.model and Data.1D.par Model output and parameter settings for a simple 1-parameter ensemble example

emul.1D Gaussian Process emulator that has been fit to the aforementioned model ensemble output

Data.AR1.Korea.model and Data.AR1.Korea.par Temperature variability and future change for 29 CMIP5 global climate models

Data.UVic.model and Data.UVic.par Model output and parameter settings for temperature anomaly output from 3-parameter ensemble of climate model UVic ESCM

Data.Sicopolis.model and Data.Sicopolis.par Model output and parameter settings for ice mass loss output from a 5-parameter ensemble of ice sheet model SICOPOLIS

emul.Sicopolis A Gaussian Process emulator that has been fit to the SICOPOLIS ensemble output

Limitations and Caveats

1. The emulator (like any other available software packages) will not work well for 'jagged' model response surfaces (high nugget): predictive uncertainty will be too high.
2. The emulator is restricted to output at regular intervals in time and space
3. The code has not been tested under conditions of extreme high / extreme low input parameter range, output time(space) coordinates range, and output range (an example would be model output ranging from -1E20 to 1E20, etc.). In such cases it is recommended to re-scale the time(space) coordinates vector, the input parameters, and/or the model output.
4. The emulator will not work on scalar model output – it requires multivariate data
5. The emulator assumes a separable covariance function, and stationarity of the covariance part of the Gaussian process.
6. The optimization of the emulator parameters degrades dramatically (and increases in time) as a function of number of free parameters. Hence, the emulator might be of limited use for large parameter ensembles
7. The emulator authors are not responsible for any code errors and/or bugs

Author(s)

Roman Olson, Won Chang, Klaus Keller and Murali Haran

Maintainer: Kelsey Ruckert <datamgmt@scrim.psu.edu>

References

R. Olson and W. Chang (2013): Mathematical framework for a separable Gaussian Process Emulator. Tech. Rep., available from www.scrim.psu.edu/resources/stilt/Olson_and_Chang_2013_Stilt_Emulator_Technical_Report.pdf.

C. E. Rasmussen and C. K. I. Williams (2006): Gaussian Processes for machine learning, the MIT press, available from www.gaussianprocess.org/gpml/

`Data.1D.model` *Synthetic model output for a simple 1-parameter ensemble example*

Description

Synthetic model output for a simple 1-parameter ensemble example

Usage

```
data(Data.1D.model)
```

Format

The format is a list containing five elements

$ t : 11-element time vector: (0, 1, ..., 10)

$ tunits : Time units="Year"

$ out : [1:11, 1:21] matrix of model response, [row,col] = [time index, parameter index]. The corresponding vector of parameters for columns is in the $par component of Data.1D.par.Rd ...

$ outname : Output name="Sample Output"

$ outunits : Output units="Unitless"

Details

Model response Y is a function of time and parameter setting Θ as follows: $Y = sin(\Theta)(1+2t+t^2)$

Examples

```
# Fit an emulator to the 1-parameter ensemble data
data(Data.1D.model)
data(Data.1D.par)
emulator(Data.1D.par, Data.1D.model, FALSE, FALSE, 1, 1)
```

Data.1D.par — *Parameter settings for 1-parameter ensemble model output in Data.1D.model*

Description

Parameter settings for 1-parameter ensemble model output in Data.1D.model

Usage

```
data(Data.1D.par)
```

Format

The format is list containing three elements:

$par [1, 1:21] matrix of parameter settings. The columns correspond to columns of $out in Data.1D.model. The parameter settings are (0, 1, ..., 20)

$parnames Parameter name = "Parameter 1"

$parunits Parameter units = "Unitless"

Examples

```
# Fit an emulator to the 1-parameter ensemble data
data(Data.1D.model)
data(Data.1D.par)
emulator(Data.1D.par, Data.1D.model, FALSE, FALSE, 1, 1)
```

Data.AR1Korea.model — *Korean modelled temperature variability and future change*

Description

Future 2081-2100 RCP8.5 emissions scenario change in summer mean maximum temperature compared to present (1973-2005) over Korea as a function of present-day (1973-2005) variability (innovation standard deviation and autocorrelation from CMIP5 GCMs)

Usage

```
data(Data.AR1Korea.model)
```

Format

The format is a list containing five elements:

$t Time vector for model output: (2081, 2082, ... 2100)

$tunits Time units = "Year"

$out [1:20, 1:29] matrix of GCM temperature change output. [row, col] = [time, model index]. The variability parameters corresponding to columns of $out are in the $par element of Data.AR1Korea.par

$outname Output name = "Temp Change 1973-2005 to 2081-2100"

$outunits Output units = "K"

Details

The CMIP5 GCM output was analysed by Jong-Soo Shin and Roman Olson at Yonsei University. The variability properties were found by maximum likelihood.

Examples

```
# Fit an emulator to the CMIP5 GCM Korean temperature variability and change
# data with innovation standard deviation and time as covariates
data(Data.AR1Korea.model)
data(Data.AR1Korea.par)
emulator(Data.AR1Korea.par, Data.AR1Korea.model, c(TRUE, FALSE), TRUE, 3, 3)
```

Data.AR1Korea.par *Korean temperature variability parameters*

Description

Korean summer mean maximum temperature 1973-2005 interannual variability parameters (innovation standard deviation and autocorrelation) for 29 CMIP5 GCMs.

Usage

```
data(Data.AR1Korea.par)
```

Format

The format is a list containing three variables

$par Variability settings for 29 CMIP5 GCMs. [1:2, 1:29] matrix. Rows are: sigma (innovation std) and rho (autocorrelation). The columns of $par correspond to columns of $out in Data.AR1Korea.model.

$parnames Parameter names for rows of $par

$parunits Parameter units for rows of $par

Details

The CMIP5 GCM output was analysed by Jong-Soo Shin and Roman Olson at Yonsei University. The variability properties were found by maximum likelihood.

Examples

```
# Fit an emulator to the CMIP5 GCM Korean temperature variability and change
# data with innovation standard deviation and time as covariates
data(Data.AR1Korea.model)
data(Data.AR1Korea.par)
emulator(Data.AR1Korea.par, Data.AR1Korea.model, c(TRUE, FALSE), TRUE, 3, 3)
```

Data.Sicopolis.model *SICOPOLIS model ensemble output*

Description

Ice mass loss output from a 5-parameter ensemble of SICOPOLIS ice sheet model

Usage

```
data(Data.Sicopolis.model)
```

Format

The format is a list containing five elements

$t 661-element time vector for model output: (1840, 1841, ..., 2500)

$tunits Time units = "Year"

$out [1:661, 1:100] matrix of Greenland Ice Sheet mass anomaly with respect to year 2003. [row, col] = [time index, parameter index]. The vector of parameter values for columns of $out is in the $par element of Data.Sicopolis.par.

$outname Output name = "GIS Mass Anomaly wrt 2003"

$outunits Output units "Gt"

Details

The ensemble was generated by Patrick Applegate. The ensemble varies five important ice sheet model parameters: Flow Enhancement Factor, Basal Sliding Factor, Geothermal Heat Flux, Snow PDD Factor, and Ice PDD Factor

Source

Applegate, P. J., Kirchner, N., Stone, E. J., Keller, K., and Greve, R., 2012, An assessment of key model parametric uncertainties in projections of Greenland Ice Sheet behavior: The Cryosphere 6, 589-606.

Examples

```
# Fit an emulator to the SICOPOLIS ensemble data
data(Data.Sicopolis.par)
data(Data.Sicopolis.model)
## Not run: emulator(Data.Sicopolis.par, Data.Sicopolis.model, c(FALSE, FALSE,
FALSE, FALSE, FALSE), FALSE, 200000, 20000)
## End(Not run)
```

`Data.Sicopolis.par` *SICOPOLIS model ensemble parameter settings*

Description

Parameter settings corresponding to SICOPOLIS ensemble model output in Data.Sicopolis.model.

Usage

```
data(Data.Sicopolis.par)
```

Format

The format is a list containing three elements

$par [1:5, 1:100] matrix of parameter settings for the SICOPOLIS model ensemble. [row, col] = [parameter number, ensemble run number]. The columns match the columns of $out element of Data.Sicopolis.model

$par Parameter names for rows of $par

$parunits Parameter units for rows of $par

Details

The ensemble was generated by Patrick Applegate. The ensemble varies five important ice sheet model parameters: Flow Enhancement Factor, Basal Sliding Factor, Geothermal Heat Flux, Snow PDD Factor, and Ice PDD Factor

Source

Applegate, P. J., Kirchner, N., Stone, E. J., Keller, K., and Greve, R., 2012, An assessment of key model parametric uncertainties in projections of Greenland Ice Sheet behavior: The Cryosphere 6, 589-606.

Examples

```
# Fit an emulator to the SICOPOLIS ensemble data
data(Data.Sicopolis.par)
data(Data.Sicopolis.model)
## Not run: emulator(Data.Sicopolis.par, Data.Sicopolis.model, c(FALSE, FALSE,
FALSE, FALSE, FALSE), FALSE, 200000, 20000)
## End(Not run)
```

Data.UVic.model *UVic ESCM climate model ensemble output*

Description

Temperature anomaly output from UVic ESCM climate model ensemble

Usage

```
data(Data.UVic.model)
```

Format

The format is a list containing five elements:

$t Time vector for model output: (1850, 1851, ... 2009)

$tunits Time units = "Year"

$out [1:160, 1:250] matrix of UVic ESCM temperature output. [row, col] = [time, run index]. The parameter settings corresponding to runs in columns of $out are in the $par element of Data.UVic.par

$outname Output name = "Temperature Anomaly from 1850-1899"

$outunits Output units = "[K]"

Details

The ensemble was run by Roman Olson at Penn State University. The ensemble varies key parameters that affect decadal- and century-scale climate response to external climate forcings: vertical ocean diffusivity, climate sensitivity, and anthropogenic sulfate aerosol scaling factor.

Source

Olson, R., R. Sriver, M. Goes, N. M. Urban, H. D. Matthews, M. Haran and K. Keller (2012): A climate sensitivity estimate using Bayesian fusion of instrumental observations and an Earth System model. Journal of Geosphysical Research - Atmospheres, 117(D04103), doi:10.1029/2011JD016620

References

Olson, R., R. Sriver, W. Chang, M. Haran, N. M. Urban and K. Keller (2013): What is the effect of unresolved internal climate variability on climate sensitivity estimates? Journal of Geophysical Research - Atmospheres, 118, doi:10.1002/jgrd.50390

Examples

```
# Fit an emulator to the UVic ESCM ensemble data using all parameter and
#time covariates
data(Data.UVic.model)
data(Data.UVic.par)
## Not run: emulator(Data.UVic.par, Data.UVic.model, c(TRUE, TRUE, TRUE), TRUE, 1, 1)
```

Data.UVic.par *UVic ESCM climate model ensemble parameter settings*

Description

Parameter settings for the UVic ESCM climate model ensemble output in Data.UVic.model

Usage

```
data(Data.UVic.par)
```

Format

The format is a list containing three variables

$par Parameter settings for the temperature output from the UVic ESCM ensemble in Data.UVic.model. [1:3, 1:250] matrix. [row, col] = [parameter number, run number]. The columns of $par correspond to columns of $out in Data.UVic.model.

$parnames Parameter names for rows of $par

$parunits Parameter units for rows of $par

Details

The ensemble was run by Roman Olson at Penn State University. The ensemble varies key parameters that affect decadal- and century-scale climate response to external climate forcings: vertical ocean diffusivity, climate sensitivity, and anthropogenic sulfate aerosol scaling factor.

Source

Olson, R., R. Sriver, M. Goes, N. M. Urban, H. D. Matthews, M. Haran and K. Keller (2012): A climate sensitivity estimate using Bayesian fusion of instrumental observations and an Earth System model. Journal of Geosphysical Research - Atmospheres, 117(D04103), doi:10.1029/2011JD016620

References

Olson, R., R. Sriver, W. Chang, M. Haran, N. M. Urban and K. Keller (2013): What is the effect of unresolved internal climate variability on climate sensitivity estimates? Journal of Geophysical Research - Atmospheres, 118, doi:10.1002/jgrd.50390

Examples

```
# Fit an emulator to the UVic ESCM ensemble data using all parameter and
# time covariates
data(Data.UVic.par)
data(Data.UVic.model)
## Not run: emulator(Data.UVic.par, Data.UVic.model, c(TRUE, TRUE, TRUE), TRUE, 1, 1)
```

emul.1D — *Emulator of model for the 1-parameter example*

Description

Gaussian Process emulator object fitted to the 1-parameter example given in Data.1D.par and Data.1D.model

Usage

```
data(emul.1D)
```

Format

This is a standard emulator list object, containing standard emulator components as described in 'emulator' function

Details

The emulator was fit with both the parameter and time covariates, with a relative tolerance of 1E-9, with two optimization attempts. The regression coefficients were estimated during the optimization process.

Examples

```
# Cross-validate the 1-parameter model emulator and plot the results at
# time index 5
data(emul.1D)
test.all(emul.1D, 5)
```

emul.predict — *To predict using an emulator (Deprecated)*

Description

To predict using an emulator. This function is deprecated. Use `predict.emul` instead.

Usage

```
emul.predict(emul, theta.star)
```

Arguments

`emul`	Standard emulator object, as output, for example, by the 'emulator' function.
`theta.star`	Parameter setting at which to predict. Must have the same number of elements as there are columns in the `emul$Theta.mat`. Vector (or, for a 1 parameter ensemble, a single numeric value).

Details

Emulator prediction follows standard formulation in Gaussian Process theory. For more details, see References.

Value

Prediction at the parameter setting `theta.star` for all times specified by `emul$t.vec`. List with components

`mean`	Posterior mean. n*1 matrix. Rows correspond to the times of `emul$t.vec`.
`covariance`	Posterior covariance matrix. n*n matrix. Diagonal elements represent variances at each time in `emul$t.vec`.

Note

Prediction outside the emulator range ("extrapolation") is currently not allowed.

References

R. Olson and W. Chang (2013): Mathematical framework for a separable Gaussian Process Emulator. Tech. Rep., available from
www.scrim.psu.edu/resources/stilt/Olson_and_Chang_2013_Stilt_Emulator_Technical_Report.pdf.

See Also

`rsurface.plot`, `test.csv`, `emulator`, `predict.emul`

Examples

```
## Not run:
# Predict using the SICOPOLIS model at a mid-range parameter setting, and plot
# the prediction and associated uncertainty
data(emul.Sicopolis)
pred <- emul.predict(emul.Sicopolis, c(3, 10, 50, 3, 12.5))
plot.default(NA, xlim=range(emul.Sicopolis$t.vec),
     ylim=range(pred$mean), xlab="Year",
     ylab="Ice Mass Loss relative to year 2003")
lines(emul.Sicopolis$t.vec, pred$mean, col="brown", lwd=3)
std <- sqrt(diag(pred$covariance))
lines(emul.Sicopolis$t.vec, pred$mean + std, col="brown", lty=2)
lines(emul.Sicopolis$t.vec, pred$mean - std, col="brown", lty=2)


# Fit an emulator to the 1-parameter test ensemble data, predict at
# Theta*=8, and plot the prediction
data(Data.1D.par)
data(Data.1D.model)
emul.1D <- emulator(Data.1D.par, Data.1D.model, TRUE, TRUE, 100, 0.1)
pred.1D <- emul.predict(emul.1D, 8)
plot(emul.1D$t.vec, pred.1D$mean, xlab="Year",
     ylab="Sample Model Output at Theta*=8")
```

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

`emul.Sicopolis` *SICOPOLIS ice sheet model emulator*

Description

Sample emulator for ice mass loss from the 5-parameter SICOPOLIS ice sheet model ensemble output

Usage

```
data(emul.Sicopolis)
```

Format

This is a standard emulator list object, containing standard emulator components as described in 'emulator' function. The emulated quantity is Greenland ice mass loss relative to year 2003 (Gt).

Details

The emulator was fit using all five parameter covariates, and the time covariate. The beta parameters are the multiple linear regression estimates. The model ensemble was generated by Patrick Applegate. The ensemble varies five important ice sheet model parameters: Flow Enhancement Factor, Basal Sliding Factor, Geothermal Heat Flux, Snow PDD Factor, and Ice PDD Factor.

Source

Applegate, P. J., Kirchner, N., Stone, E. J., Keller, K., and Greve, R., 2012, An assessment of key model parametric uncertainties in projections of Greenland Ice Sheet behavior: The Cryosphere 6, 589-606.

Examples

```
# Plot the response surface as a function of snow PDD factor, and ice
# PDD factor at year 2500 (time index 661), while keeping other parameters
# at mid-range values
data(emul.Sicopolis)
## Not run:
  rsurface.plot(emul=emul.Sicopolis, parind=c(4,5), parvals=c(3, 10, 50, NA, NA),
      tind=661, n1=5, n2=5)

## End(Not run)
```

`emulator` *Function to fit an emulator to ensemble model output*

Description

Fits a Gaussian Process emulator to regularly spaced time-series (or 1D spatial) model output. This emulator can later be used to interpolate the model output across model parameter settings. The emulator is fit by maximizing the emulator likelihood using a PORT local optimization routine. The emulator has a flexible structure which can be chosen by the user (see Details).

Usage

```
emulator(mpars, moutput, par.reg, time.reg, kappa0, zeta0, myrel.tol =
NULL, twice = FALSE, fix.betas = TRUE)
```

Arguments

`mpars`	Model ensemble parameter settings. A list with components: **$par** Matrix of ensemble parameter settings. [row, column] = [parameter index, run index]. Columns should correspond to columns of the model output matrix `moutput$out`. **$parnames** Parameter names for rows of $par (character vector) **$parunits** Units for parameters in rows of $par (character vector)
`moutput`	Model output. A list with components: **$t** Time vector for rows of $out (vector) **$tunits** Time units (character) **$out** Model output matrix [row, col] = [time, run index]. The run indices should match with the `mpars$par` argument. **$outname** name (character) **$outunits** units (character
`par.reg`	Logical vector specifying which parameters to include into the regression matrix. Elements refer to rows of `mpars$par`. Note that if nothing is included into the regression matrix, the mean is still optimized.
`time.reg`	Logical specifying whether to include time into the regression matrix.
`kappa0`	Initial guess for the parameter covariance scaling factor. Larger values lead to higher parameter covariance.
`zeta0`	Initial guess for the nugget.
`myrel.tol`	Relative tolerance for optimization. If you want to use the defaults, assign it to NULL. The optimization stops if it thinks the true likelihood is within this fraction of current likelihood. The higher the number the sooner the optimization stops, but the emulator is 'worse'.
`twice`	Logical. If `TRUE`, the optimization is done twice with different initial parameter values, and then the best of the two is chosen as final emulator. Default is `FALSE`.
`fix.betas`	Logical. If `TRUE`, beta regression coefficients are fixed, and if `FALSE`, they are estimated. Default is `TRUE`.

Details

The emulator parameters are optimized using a local optimization method, to within the specified relative tolerance. The optimization maximizes emulator likelihood. Optionally, if `twice=TRUE`, the optimization is done twice with different initial parameter settings, and the emulator with the highest likelihood is selected. Depending on the value of `fix.betas`, the regression beta parameters can either be fixed at their multiple linear regression estimates, or estimated together with other emulator parameters. Mean structure is flexible, and user can choose which parameter and time dimensions to use as regressors via the `time.reg` and `par.reg` parameters

Value

Emulator returns an object of `class` "emul" and is a list with components (see also Reference in the References Section).

`$Theta.mat`	Theta matrix. [row, col] = [run number, parameter number]
`$t.vec`	Time vector
`$Y.mat`	Data matrix Y
`$X.mat`	Regression matrix
`$beta.vec`	A vector of regression parameters
`$kappa`	Parameter covariance scaling factor
`$phi.vec`	A vector of range parameters phi
`$zeta`	Nugget
`$n`	Length of time dimension
`$rho`	Time AR(1) parameter
`$p`	Number of model runs in the ensemble
`$vecC`	Data matrix Y minus the mean vector
`$par.reg`	Logical vector specifying which parameters to include in the regression matrix
`$time.reg`	Logical, specifies whether to include time into the regression matrix
`$Sigma.mats`	List of two precomputed matrices that may be useful: `$Sigma.t.mat` and `$Sigma.theta.mat` (see References)
`$Sigma.theta.inv.mat`	Inverse of `$Sigma.mats$Sigma.theta.mat`

Note

Evaluation of this function (especially for large datasets and many parameters) might take a long time.

Limitations and Caveats

1. The emulator will not work well for 'jagged' model response surfaces (high nugget)
2. The emulator is restricted to output at regular intervals in time and space

3. The code has not been tested under conditions of extreme high / extreme low input parameter range, output time(space) coordinates range, and output range (an example would be model output ranging from -1E20 to 1E20, etc.). In such cases it is recommended to re-scale the time(space) coordinates vector, the input parameters, and/or the model output.
4. The emulator will not work on scalar model output – it requires multivariate data
5. The emulator assumes a separable covariance function, and stationarity of the covariance part of the Gaussian process.
6. The optimization of the emulator parameters degrades dramatically (and increases in time) as a function of number of free parameters. Hence, the emulator might be of limited use for large parameter ensembles
7. The emulator authors are not responsible for any code errors and/or bugs

Note

If the final emulator performs poorly, try adjusting the settings, especially the initial `kappa0` and `zeta0` values.

References

R. Olson and W. Chang (2013): Mathematical framework for a separable Gaussian Process Emulator. Tech. Rep., available from
www.scrim.psu.edu/resources/stilt/Olson_and_Chang_2013_Stilt_Emulator_Technical_Report.pdf.

See Also

`emul.lik`, `optimize.emul`

Examples

```
# Fit an emulator to the example 1D parameter ensemble data
# Do not use any covariates, use standard settings
data(Data.1D.model)
data(Data.1D.par)
emul.1D <- emulator(Data.1D.par, Data.1D.model, FALSE, FALSE, 1, 1)

# Take a look at the range and regression parameters
cat("Range parameters are:", emul.1D$phi.vec, "\n")
cat("Regression parameters are:", emul.1D$beta.vec, "\n")

# Predict using the emulator at Theta*=3
pred.1D <- predict(emul.1D, 3)

# Plot the prediction
plot(emul.1D$t.vec, pred.1D$mean, xlab="Year", ylab="Sample Output")


## Not run:
    # Fit an emulator to the UVic ESCM 3-parameter ensemble temperature
    # output Use time and aerosol scaling covariates (parameter 3), run
    # the optimization twice with relative tolerance of 0.1, keep
```

```
    # regression parameters at their multiple linear regression
    # estimates data(Data.UVic.model) data(Data.UVic.par) UVic.emul <-
    # emulator(mpars=Data.UVic.par, moutput=Data.UVic.model,
    # par.reg=c(FALSE, FALSE, TRUE), time.reg=TRUE, kappa0=1, zeta0=1,
    # myrel.tol=0.1, twice=TRUE, fix.betas=TRUE)

## End(Not run)
```

`predict.emul` *Predict Method for an Emulator*

Description

To predict using an emulator

Usage

```
## S3 method for class 'emul'
predict(object, theta.star, ...)
```

Arguments

`object`	Standard emulator object of class "emul", as output, for example, by the `emulator` function.
`theta.star`	Parameter setting at which to predict. Must have the same number of elements as there are columns in the `object$Theta.mat`. Vector (or, for a 1 parameter ensemble, a single numeric value).
`...`	Additional optional arguments. At present no optional arguments are used.

Details

Emulator prediction follows standard formulation in Gaussian Process theory. For more details, see References.

Value

Prediction at the parameter setting `theta.star` for all times specified by `object$t.vec`. List with components

`mean`	Posterior mean. n*1 matrix. Rows correspond to the times of `object$t.vec`.
`covariance`	Posterior covariance matrix. n*n matrix. Diagonal elements represent variances at each time in `object$t.vec`.

Note

Prediction outside the emulator range ("extrapolation") is currently not allowed.

References

R. Olson and W. Chang (2013): Mathematical framework for a separable Gaussian Process Emulator. Tech. Rep., available from
www.scrim.psu.edu/resources/stilt/Olson_and_Chang_2013_Stilt_Emulator_Technical_Report.pdf.

See Also

rsurface.plot, test.csv, emulator

Examples

```
# Predict using the SICOPOLIS model at a mid-range parameter setting, and plot
# the prediction and associated uncertainty
data(emul.Sicopolis)
pred <- predict(emul.Sicopolis, c(3, 10, 50, 3, 12.5))
plot.default(NA, xlim=range(emul.Sicopolis$t.vec),
     ylim=range(pred$mean), xlab="Year",
     ylab="Ice Mass Loss relative to year 2003")
lines(emul.Sicopolis$t.vec, pred$mean, col="brown", lwd=3)
std <- sqrt(diag(pred$covariance))
lines(emul.Sicopolis$t.vec, pred$mean + std, col="brown", lty=2)
lines(emul.Sicopolis$t.vec, pred$mean - std, col="brown", lty=2)


# Fit an emulator to the 1-parameter test ensemble data, predict at
# Theta*=8, and plot the prediction
data(Data.1D.par)
data(Data.1D.model)
emul.1D <- emulator(Data.1D.par, Data.1D.model, TRUE, TRUE, 100, 0.1)
pred.1D <- predict(emul.1D, 8)
plot(emul.1D$t.vec, pred.1D$mean, xlab="Year",
     ylab="Sample Model Output at Theta*=8")
```

---

rsurface.plot *To produce a response surface plot of the emulator*

Description

To produce a response surface plot of the emulator

Usage

```
rsurface.plot(emul, parind, parvals, tind, n1, n2, zlim = NULL)
```

Arguments

emul	A standard emulator object, as output, for example, by the 'emulator' function
parind	Vector of parameter indices (x and y) (columns of emul$Theta.mat) for which the response surface is desired. Should contain only two elements. parind[2] can be less or more than parind[1].
parvals	A vector of parameter values to use for the rest of the parameters (that are kept at constant values). Must be the same length as the number of columns in emul$Theta.mat. The columns corresponding to parind should be NA.
tind	Time index at which to predict
n1	X direction grid size
n2	Y direction grid size
zlim	Vector of z-limits for the filled.contour function. Default is the range of data plotted.

Details

Produces a response surface plot of the emulator as a function of selected parameters, while the rest of the parameters are fixed at their values set by parvals. Use n1 and n2 to specify X and Y grid size. Optionally, plot limits can be specified through zlim. The code relies on the filled.contour function.

Value

None

Note

Evaluation of this function (especially for large datasets and grid sizes) might take a long time. This is because in current specification, the 'predict.emul' function predicts the entire time-series (or space transect) at once.

References

R. Olson and W. Chang (2013): Mathematical framework for a separable Gaussian Process Emulator. Tech. Rep., available from
www.scrim.psu.edu/resources/stilt/Olson_and_Chang_2013_Stilt_Emulator_Technical_Report.pdf.

See Also

emulator, predict.emul

Examples

```
# Plot the SICOPOLIS ice mass loss in year 2500 as a function of Snow
# PDD Factor and Ice PDD Factor, at mid-range values of other parameters
data(emul.Sicopolis)
## Not run:
  rsurface.plot(emul=emul.Sicopolis, parind=c(4,5), parvals=c(3, 10, 50, NA, NA),
```

```
      tind=661, n1=5, n2=5)

## End(Not run)
```

---

sep.cov *Construct time and parameter covariance matrices*

Description

Construct time and parameter covariance matrices. The time matrix follows standard AR(1) covariance. The parameter covariance is squared exponential, separable between the parameters, with an extra nugget term. While not directly related to other functions in this package, this function can be re-used if someone wants to build a new/different Gaussian Process emulator code.

Usage

```
sep.cov(Theta.mat, t.vec, rho, kappa, phi.vec, zeta)
```

Arguments

Theta.mat	Parameter matrix of the ensemble. [row, col] = [run number, parameter number]. See References for more information on this and other arguments.
t.vec	Evenly spaced vector of times for model output (can also be a coordinate vector for a regularly spaced spatial transect)
rho	Lag-1 time autocorrelation parameter ρ
kappa	Parameter covariance scaling factor κ
phi.vec	Vector of range parameters. Must have the same length as number of columns in Theta.mat. All elements should be positive.
zeta	Nugget ζ. Should be positive.

Details

The time covariance is an n*n matrix (where n is the length of time dimension). Its (j,k) element is specified by $\varsigma_{t,jk} = \frac{\rho^{|t_j - t_k|}}{1-\rho^2}$. The parameter covariance is a p*p matrix (where p is the number of runs in the model ensemble). Specifically, $\Sigma_{\theta,ij} = \kappa * exp(-A) + \zeta * 1(i = j)$ where $A = \sum_{k=1}^{m} \frac{(\theta_{k,i} - \theta_{k,j})^2}{\phi_k^2}$. Here k refers to parameter index. For more details on the time and parameter covariance matrices produced by this function, see References.

Value

List with components

$Sigma.t.mat	Time covariance matrix
$Sigma.theta.mat	Parameter covariance matrix

References

R. Olson and W. Chang (2013): Mathematical framework for a separable Gaussian Process Emulator. Tech. Rep., available from
www.scrim.psu.edu/resources/stilt/Olson_and_Chang_2013_Stilt_Emulator_Technical_Report.pdf.

See Also

predict.emul

Examples

```
# Construct time AR(1) and square exponential separable parameter
# covariance matrix for a simple 1D parameter ensemble output example
data(Data.1D.par)
data(Data.1D.model)

Theta.mat.1D <- t(Data.1D.par$par)
t.vec.1D     <- Data.1D.model$t

# Use lag-1 time autocorrelation of 0.9, nugget and parameter covariance
# scaling factor of 100, and range of 3.
cov <- sep.cov(Theta.mat.1D, t.vec.1D, 0.9, 100, 3, 100)

# Find the covariance between second parameter setting (Theta=1) and
# ninth parameter setting (Theta=8)
cov.2.9 <- cov$Sigma.theta.mat[2,9]
cat("Covariance between Theta=1 and Theta=8 is:", cov.2.9, "\n")

# Plot the time covariance matrix [for fun]
# Note how covariance is high between similar years, but is low for markedly
# different years. Produces a pretty plot.
filled.contour(t.vec.1D, t.vec.1D, cov$Sigma.t.mat, xlab="Year", ylab="Year")
```

---

test.all *To test an emulator using leave-one-out cross-validation*

Description

To test an emulator using the leave-one-out cross validation analysis, whereby each ensemble run is first excluded from an emulator, and the emulator is then used to predict at the parameter settings of the excluded run. The process is repeated for all runs within the ensemble. The function also makes a plot of emulator predictions vs. actual model output, and a plot of normalized prediction errors for each excluded run.

Usage

```
test.all(final.emul, t.plot)
```

Arguments

final.emul	Standard emulator object, that can be generated, for example, by 'emulator' function
t.plot	Time index of the emulator to analyze

Details

For a perfect emulator, emulator predictions vs. actual model output will fall exactly on a 1:1 line. If a withheld run is at the corner of parameter space, prediction gives a warning. Such runs are omitted from the plot.

See Also

test.csv

Examples

```
# Cross-validate the 1D-parameter ensemble emulator at the first time index
data(emul.1D)
test.all(emul.1D, 1)
```

---

test.csv *To cross-validate a Gaussian Process emulator*

Description

To test an emulator using cross-validation. Any reasonable number of runs can be excluded from the emulator for testing. Then, the emulator is used to predict at the excluded parameter settings, and the results can be plotted. Optionally, prediction standard deviation can also be plotted. The program is reproducible if the seed number is specified.

Usage

```
test.csv(final.emul, num.test, plot.std, theseed = NULL, test.runind =
NULL, make.plot = TRUE)
```

Arguments

final.emul	A standard emulator object. It can be generated by, for example, the 'emulator' function.
num.test	Number of test runs to withhold at random from the emulator. Can not be more than half of the ensemble.
plot.std	If TRUE, prediction std. is plotted around the emulator prediction
theseed	Seed for random number generator. Default is NULL.
test.runind	If not NULL, a monotonically increasing vector of run indices at which to test the emulator. Has to contain num.test number of elements. If NULL, run indices are generated at random. Default is NULL.
make.plot	If TRUE (the default) produced a cross-validation plot.

Details

The function withholds model runs from the emulator, and then uses the withheld emulator to predict at the excluded parameter settings. Any reasonable number of runs up to 1/2 of the ensemble can be excluded. If `make.plot=TRUE` the plot of model output and emulator prediction at the excluded setting is produced. Setting `plot.std=TRUE` adds the prediction standard deviation to the plot. If `plot.std=TRUE` but `make.plot=FALSE` a warning is given and nothing is plotted. The default setting is to exclude `num.test` runs at random. If `theseed` is specified the results become reproducible as the seed passed on to the R random number generator is fixed. Optionally, the excluded run indices can be specified explicitly via the vector `test.runind`. If both `theseed` and `test.runind` arguments are specified, then `theseed` is ignored.

The function prints out the excluded run indices and outputs some basic guidance. The function handles runs which are at parameter space boundaries by passing NA to the relevant rows of output matrices. If prediction at a particular withheld run gives error (most likely due to the fact that extrapolation is not allowed and the excluded run is at the corner of parameter space) that particular run is not plotted. Model output is beige, and emulator output is brown.

By 'withholding' runs from an emulator it is meant that the emulator statistical parameters are kept the same, whereas the withheld runs are eliminated from all relevant emulator components. The ensemble size `final.emul$p` is reduced accordingly.

Value

List with components

`$model.out.test` Model output at test parameter settings [row,col] = [test run index, time index]

`$emul.out.test` Emulator output at test parameter settings [row,col] = [test run index, time index]

`$emul.std.test` Emulator prediction standard deviation at test parameter settings [row,col] = [test run index, time index]

`$coverage` Empirical 95% coverage, expressed as a fraction

The time vector for columns of all matrix components is `final.emul$t.vec`

References

R. Olson and W. Chang (2013): Mathematical framework for a separable Gaussian Process Emulator. Tech. Rep., available from
www.scrim.psu.edu/resources/stilt/Olson_and_Chang_2013_Stilt_Emulator_Technical_Report.pdf.

See Also

`test.all`, `predict.emul`

Examples

```
# Test the 1D emulator by withholding the second parameter
# setting, and then predicting at the withheld run
data(emul.1D)
test.1D <- test.csv(final.emul=emul.1D, num.test=1, plot.std=FALSE,
```

```
                    test.runind=2, make.plot=FALSE)

# Create a custom plot
plot.default(NA, xlim=range(emul.1D$t.vec), ylim=range(test.1D$model.out.test),
             xlab="Year", ylab="Sample Output", main="Emulator Test at Theta=1",
             cex.lab=1.3, cex.axis=1.3, cex.main=1.3)
lines(emul.1D$t.vec, test.1D$model.out.test, lwd=4, lty=2, col="green")
# The emulator prediction is close to perfect! Yay!
lines(emul.1D$t.vec, test.1D$emul.out.test, lwd=2, col="yellow")

# Produce a legend
model.max <- max(test.1D$model.out.test)
# We already know that time ranges from 0 to 10 so figuring out x placement
# for the legend is not hard
legend(1, model.max, c("Model Output", "Emulator Predictions"),
       col=c("green", "yellow"), lty=c(2,1), lwd=c(4,2), cex=1.3)



# Test SICOPOLIS emulator at three pre-selected parameter settings
# specified by theseed=1234321
data(emul.Sicopolis)
# Plot the standard deviation in all three cases
test.csv(final.emul=emul.Sicopolis, num.test=3, plot.std=TRUE, theseed=1234321,
         make.plot=TRUE)
```

Index