Package: tgp (via r-universe)

September 4, 2024

Title Bayesian Treed Gaussian Process Models

Version 2.4-23

Date 2024-08-22

Depends R (>= 2.14.0)

Imports maptree

Suggests MASS

Description Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian processes (GPs) with jumps to the limiting linear model (LLM). Special cases also implemented include Bayesian linear models, CART, treed linear models, stationary separable and isotropic GPs, and GP single-index models. Provides 1-d and 2-d plotting functions (with projection and slice capabilities) and tree drawing, designed for visualization of tgp-class output. Sensitivity analysis and multi-resolution models are supported. Sequential experimental design and adaptive sampling functions are also provided, including ALM, ALC, and expected improvement. The latter supports derivative-free optimization of noisy black-box functions. For details and tutorials, see Gramacy (2007) <doi:10.18637/jss.v019.i09> and Gramacy & Taddy (2010) <doi:10.18637/jss.v033.i06>.

Maintainer Robert B. Gramacy <`rbg@vt.edu`>

License LGPL

URL `https://bobby.gramacy.com/r_packages/tgp/`

NeedsCompilation yes

Author Robert B. Gramacy [aut, cre], Matt A. Taddy [aut]

Date/Publication 2024-09-03 14:30:02 UTC

Repository https://rbgramacy.r-universe.dev

RemoteUrl https://github.com/cran/tgp

RemoteRef HEAD

RemoteSha ee420b9727b856b5173680c7112b600375731e35

Contents

`tgp-package` *The Treed Gaussian Process Model Package*

Description

A Bayesian nonstationary nonparametric regression and design package implementing an array of models of varying flexibility and complexity.

Details

This package implements Bayesian nonstationary, semiparametric nonlinear regression with "treed Gaussian process models" with jumps to the limiting linear model (LLM). The package contains functions which facilitate inference for seven regression models of varying complexity using Markov chain Monte Carlo (MCMC): linear model, CART (Classification and Regression Tree), treed linear model, Gaussian process (GP), GP with jumps to the LLM, GP single-index models, treed GPs, treed GP LLMs, and treed GP single-index models. R provides an interface to the C/C++ backbone, and a serves as mechanism for graphically visualizing the results of inference and posterior predictive surfaces under the models. A Bayesian Monte Carlo based sensitivity analysis is implemented, and multi-resolution models are also supported. Sequential experimental design and adaptive sampling functions are also provided, including ALM, ALC, and expected improvement. The latter supports derivative-free optimization of noisy black-box functions.

For a fuller overview including a complete list of functions, demos and vignettes, please use `help(package="tgp")`.

Author(s)

Robert B. Gramacy, <rbg@vt.edu>, and Matt Taddy, <mataddy@amazon.com>

References

Gramacy, R. B. (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC. (See Chapter 9.) https://bobby.gramacy.com/surrogates/

Gramacy, R. B. (2007). **tgp***: An* R *Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models.* Journal of Statistical Software, **19**(9). https://www.jstatsoft.org/v19/i09 doi:10.18637/jss.v019.i09

Robert B. Gramacy, Matthew Taddy (2010). *Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with* **tgp** *Version 2, an* R *Package for Treed Gaussian Process Models.* Journal of Statistical Software, **33**(6), 1–48. https://www.jstatsoft.org/v33/i06/ doi:10.18637/jss.v033.i06

Gramacy, R. B., Lee, H. K. H. (2008). *Bayesian treed Gaussian process models with an application to computer modeling*. Journal of the American Statistical Association, 103(483), pp. 1119-1130. Also available as ArXiv article 0710.4536 https://arxiv.org/abs/0710.4536

Robert B. Gramacy, Heng Lian (2011). *Gaussian process single-index models as emulators for computer experiments*. Available as ArXiv article 1009.4241 https://arxiv.org/abs/1009.4241

Gramacy, R. B., Lee, H. K. H. (2006). *Adaptive design of supercomputer experiments.* Available as UCSC Technical Report ams2006-02.

Gramacy, R.B., Samworth, R.J., and King, R. (2007) *Importance Tempering.* ArXiV article 0707.4242 https://arxiv.org/abs/0707.4242

Gray, G.A., Martinez-Canales, M., Taddy, M.A., Lee, H.K.H., and Gramacy, R.B. (2007) *Enhancing Parallel Pattern Search Optimization with a Gaussian Process Oracle*, SAND2006-7946C, Proceedings of the NECDC

https://bobby.gramacy.com/r_packages/tgp/

---

btgp *Bayesian Nonparametric & Nonstationary Regression Models*

Description

The seven functions described below implement Bayesian regression models of varying complexity: linear model, linear CART, Gaussian process (GP), GP with jumps to the limiting linear model (LLM), treed GP, and treed GP LLM.

Usage

```
blm(X, Z, XX = NULL, meanfn = "linear", bprior = "bflat",
        BTE = c(1000, 4000, 3), R = 1, m0r1 = TRUE, itemps = NULL,
        pred.n = TRUE, krige = TRUE, zcov = FALSE, Ds2x = FALSE,
        improv = FALSE, sens.p = NULL, trace = FALSE, verb = 1, ...)
```

```
btlm(X, Z, XX = NULL, meanfn = "linear", bprior = "bflat",
        tree = c(0.5, 2), BTE = c(2000, 7000, 2), R = 1, m0r1 = TRUE,
 itemps = NULL, pred.n = TRUE, krige = TRUE, zcov = FALSE,
        Ds2x = FALSE, improv = FALSE, sens.p = NULL, trace = FALSE,
        verb = 1, ...)
bcart(X, Z, XX = NULL, bprior = "bflat", tree = c(0.5, 2),
        BTE = c(2000, 7000, 2), R = 1, m0r1 = TRUE, itemps = NULL,
        pred.n = TRUE, krige = TRUE, zcov = FALSE, Ds2x = FALSE,
        improv=FALSE, sens.p = NULL, trace = FALSE, verb = 1, ...)
bgp(X, Z, XX = NULL, meanfn = "linear", bprior = "bflat",
        corr = "expsep", BTE = c(1000, 4000, 2), R = 1, m0r1 = TRUE,
 itemps = NULL, pred.n = TRUE, krige = TRUE, zcov = FALSE,
        Ds2x = FALSE, improv = FALSE, sens.p = NULL, nu = 1.5,
        trace = FALSE, verb = 1, ...)
bgpllm(X, Z, XX = NULL, meanfn = "linear", bprior = "bflat",
        corr = "expsep", gamma=c(10,0.2,0.7), BTE = c(1000, 4000, 2),
        R = 1, m0r1 = TRUE, itemps = NULL, pred.n = TRUE,
        krige = TRUE, zcov = FALSE, Ds2x = FALSE, improv = FALSE,
        sens.p = NULL, nu = 1.5, trace = FALSE, verb = 1, ...)
btgp(X, Z, XX = NULL, meanfn = "linear", bprior = "bflat",
        corr = "expsep", tree = c(0.5, 2), BTE = c(2000, 7000, 2),
        R = 1, m0r1 = TRUE, linburn = FALSE, itemps = NULL,
 pred.n = TRUE, krige = TRUE, zcov = FALSE, Ds2x = FALSE,
        improv = FALSE, sens.p = NULL, nu = 1.5, trace = FALSE,
        verb = 1, ...)
btgpllm(X, Z, XX = NULL, meanfn = "linear", bprior = "bflat",
        corr = "expsep", tree = c(0.5, 2), gamma=c(10,0.2,0.7),
 BTE = c(2000, 7000, 2), R = 1, m0r1 = TRUE, linburn = FALSE,
        itemps = NULL, pred.n = TRUE, krige = TRUE, zcov = FALSE,
        Ds2x = FALSE, improv = FALSE, sens.p = NULL, nu = 1.5,
        trace = FALSE, verb = 1, ...)
```

Arguments

Each of the above functions takes some subset of the following arguments...

`X`	`data.frame`, `matrix`, or vector of inputs `X`
`Z`	Vector of output responses `Z` of length equal to the leading dimension (rows) of `X`, i.e., `length(Z) == nrow(X)`
`XX`	Optional `data.frame`, `matrix`, or vector of predictive input locations with the same number of columns as `X`, i.e., `ncol(XX) == ncol(X)`
`meanfn`	A choice of mean function for the process. When `meanfn = "linear"` (default), then we have the process $$Z = (\mathbf{1} \;\; \mathbf{X})\beta + W(\mathbf{X})$$ where $W(\mathbf{X})$ represents the Gaussian process part of the model (if present). Otherwise, when `meanfn = "constant"`, then $$Z = \beta_0 + W(\mathbf{X})$$

`bprior`	Linear (beta) prior, default is `"bflat"`; alternates include `"b0"` hierarchical Normal prior, `"bmle"` empirical Bayes Normal prior, `"b0not"` Bayesian treed LM-style prior from Chipman et al. (same as `"b0"` but without `tau2`), `"bmzt"` a independent Normal prior (mean zero) with inverse-gamma variance (`tau2`), and `"bmznot"` is the same as `"bmznot"` without `tau2`. The default `"bflat"` gives an "improper" prior which can perform badly when the signal-to-noise ratio is low. In these cases the "proper" hierarchical specification `"b0"` or independent `"bmzt"` or `"bmznot"` priors may perform better
`tree`	a 2-vector containing the tree process prior parameterization `c(alpha, beta)` specifying $$p_{\text{split}}(\eta, \mathcal{T}) = \alpha * (1+\eta)^\beta$$ automatically giving zero probability to trees with partitions containing less than `min(c(10,nrow(X)+1))` data points. You may also specify a longer vector, writing over more of the components of the `$tree` output from `tgp.default.params`
`gamma`	Limiting linear model parameters `c(g, t1, t2)`, with growth parameter `g > 0` minimum parameter `t1 >= 0` and maximum parameter `t1 >= 0`, where `t1 + t2 <= 1` specifies $$p(b\|d) = t_1 + \exp\left\{\frac{-g(t_2 - t_1)}{d - 0.5}\right\}$$
`corr`	Gaussian process correlation model. Choose between the isotropic power exponential family (`"exp"`) or the separable power exponential family (`"expsep"`, default); the current version also supports the isotropic Matern (`"matern"`) and single-index Model (`"sim"`) as "beta" functionality.
`BTE`	3-vector of Monte-carlo parameters (B)urn in, (T)otal, and (E)very. Predictive samples are saved every E MCMC rounds starting at round B, stopping at T.
`R`	Number of repeats or restarts of `BTE` MCMC rounds, default `R=1` is no restarts
`m0r1`	If `TRUE` (default) the responses `Z` will be scaled to have a mean of zero and a range of 1
`linburn`	If `TRUE` initializes MCMC with `B` (additional) rounds of Bayesian Linear CART (`btlm`); default is `FALSE`
`itemps`	Importance tempering (IT) inverse temperature ladder, or powers to improve mixing. See `default.itemps`. The default is no IT `itemps = NULL`
`pred.n`	`TRUE` (default) value results in prediction at the inputs `X`; `FALSE` skips prediction at `X` resulting in a faster implementation
`krige`	`TRUE` (default) value results in collection of kriging means and variances at predictive (and/or data) locations; `FALSE` skips the gathering of kriging statistics giving a savings in storage
`zcov`	If `TRUE` then the predictive covariance matrix is calculated– can be computationally (and memory) intensive if `X` or `XX` is large. Otherwise only the variances (diagonal of covariance matrices) are calculated (default). See outputs `Zp.s2`, `ZZ.s2`, etc., below
`Ds2x`	`TRUE` results in ALC (Active Learning–Cohn) computation of expected reduction in uncertainty calculations at the `XX` locations, which can be used for adaptive sampling; `FALSE` (default) skips this computation, resulting in a faster implementation

`improv`	`TRUE` results in samples from the improvement at locations `XX` with respect to the observed data minimum. These samples are used to calculate the expected improvement over `XX`, as well as to rank all of the points in `XX` in the order that they should be sampled to minimize the expected multivariate improvement (refer to Schonlau et al, 1998). Alternatively, `improv` can be set to any positive integer 'g', in which case the ranking is performed with respect to the expectation for improvement raised to the power 'g'. Increasing 'g' leads to rankings that are more oriented towards a global optimization. The option `FALSE` (default) skips these computations, resulting in a faster implementation. Optionally, a two-vector can be supplied where `improv[2]` is interpreted as the (maximum) number of points to rank by improvement. See the note below. If not specified, the entire `XX` matrix is ranked.
`sens.p`	Either `NULL` or a vector of parameters for sensitivity analysis, built by the function `sens`. Refer there for details
`nu`	"beta" functionality: fixed smoothness parameter for the Matern correlation function; `nu + 0.5` times differentiable predictive surfaces result
`trace`	`TRUE` results in a saving of samples from the posterior distribution for most of the parameters in the model. The default is `FALSE` for speed/storage reasons. See note below
`verb`	Level of verbosity of R-console print statements: from 0 (none); 1 (default) which shows the "progress meter"; 2 includes an echo of initialization parameters; up to 3 and 4 (max) with more info about successful tree operations
`...`	These ellipses arguments are interpreted as augmentations to the prior specification generated by `params <- tgp.default.params(ncol(X)+1)`. You may use these to specify a custom setting of any of default parameters in the output list `params` except those for which a specific argument is already provided (e.g., `params$corr` or `params$bprior`) or those which contradict the type of `b*` function being called (e.g., `params$tree` or `params$gamma`); these redundant or possibly conflicting specifications will be ignored. Refer to `tgp.default.params` for details on the prior specification

Details

The functions and their arguments can be categorized by whether or not they use treed partitioning (T), GP models, and jumps to the LLM (or LM)

blm	LM	Linear Model
btlm	T, LM	Treed Linear Model
bcart	T	Treed Constant Model
bgp	GP	GP Regression
bgpllm	GP, LLM	GP with jumps to the LLM
btgp	T, GP	treed GP Regression
btgpllm	T, GP, LLM	treed GP with jumps to the LLM

Each function implements a special case of the generic function `tgp` which is an interface to C/C++ code for treed Gaussian process modeling of varying parameterization. Documentation for `tgp` has been declared redundant, and has subsequently been removed. To see how the `b*` functions use `tgp` simply examine the function. In the latest version, with the addition of the ellipses "..." argument, there is nothing that can be done with the direct `tgp` function that cannot also be done with a `b*` function

Only functions in the T (tree) category take the `tree` argument; GP category functions take the `corr` argument; and LLM category functions take the `gamma` argument. Non-tree class functions omit the `parts` output, see below

`bcart` is the same as `btlm` except that only the intercept term in the LM is estimated; the others are zero, thereby implementing a Bayesian version of the original CART model

The `sens.p` argument contains a vector of parameters for sensitivity analysis. It should be `NULL` unless created by the `sens` function. Refer to `help(sens)` for details.

If `itemps =! NULL` then importance tempering (IT) is performed to get better mixing. After each restart (when `R > 1`) the observation counts are used to update the pseudo-prior. Stochastic approximation is performed in the first burn-in rounds (for `B-T` rounds, not `B`) when `c0` and `n0` are positive. Every subsequent burn-in after the first restart is for `B` rounds in order to settle-in after using the observation counts. See `default.itemps` for more details and an example

Please see `vignette("tgp")` for a detailed illustration

Value

bgp returns an object of class `"tgp"`. The function `plot.tgp` can be used to help visualize results.

An object of class `"tgp"` is a list containing at least the following components... The `parts` output is unique to the T (tree) category functions. Tree viewing is supported by `tgp.trees`

`X`	Input argument: `data.frame` of inputs `X`
`n`	Number of rows in `X`, i.e., `nrow(X)`
`d`	Number of cols in `X`, i.e., `ncol(X)`
`Z`	Vector of output responses `Z`
`XX`	Input argument: `data.frame` of predictive locations `XX`
`nn`	Number of rows in `XX`, i.e., `nrow(XX)`
`BTE`	Input argument: Monte-carlo parameters
`R`	Input argument: restarts
`linburn`	Input argument: initialize MCMC with linear CART
`params`	`list` of model parameters generated by `tgp.default.params` and subsequently modified according to the calling `b*` function and its arguments
`dparams`	Double-representation of model input parameters used by the C-code
`itemps`	`data.frame` containing the importance tempering ladders and pseudo-prior: `$k` has inverse inverse temperatures (from the input argument), `$k` has an *updated* pseudo-prior based on observation counts and (possibly) stochastic approximation during burn-in and (input) stochastic approximation parameters c_0 and n_0. See `default.itemps` for more info
`Zp.mean`	Vector of mean predictive estimates at `X` locations
`Zp.q1`	Vector of 5% predictive quantiles at `X` locations

`Zp.q2`	Vector of 95% predictive quantiles at `X` locations
`Zp.q`	Vector of quantile norms `Zp.q2-Zp.q1`
`Zp.s2`	If input `zcov = TRUE`, then this is a predictive covariance matrix for the inputs at locations `X`; otherwise then this is a vector of predictive variances at the `X` locations (diagonal of the predictive covariance matrix). Only appears when input `pred.n = TRUE`
`Zp.km`	Vector of (expected) kriging means at `X` locations
`Zp.vark`	Vector of posterior variance for kriging surface (no additive noise) at `X` locations
`Zp.ks2`	Vector of (expected) predictive kriging variances at `X` locations
`ZZ.mean`	Vector of mean predictive estimates at `XX` locations
`ZZ.q1`	Vector of 5% predictive quantiles at `XX` locations
`ZZ.q2`	Vector of 95% predictive quantiles at `XX` locations
`ZZ.q`	Vector of quantile norms `ZZ.q2-ZZ.q1`, used by the ALM adaptive sampling algorithm
`ZZ.s2`	If input `zcov = TRUE`, then this is a predictive covariance matrix for predictive locations `XX`; otherwise then this is a vector of predictive variances at the `XX` locations (diagonal of the predictive covariance matrix). Only appears when input `XX != NULL`
`ZpZZ.s2`	If input `zcov = TRUE`, then this is a predictive `n * nn` covariance matrix between locations in `X` and `XX`; Only appears when `zcov = TRUE` and both `pred.n = TRUE` and `XX != NULL`
`ZZ.km`	Vector of (expected) kriging means at `XX` locations
`ZZ.vark`	Vector of posterior variance for kriging surface (no additive noise) at `XX` locations
`ZZ.ks2`	Vector of (expected) predictive kriging variances at `XX` locations
`Ds2x`	If argument `Ds2x=TRUE`, this vector contains ALC statistics for `XX` locations
`improv`	If argument `improv` is `TRUE` or a positive integer, this is a 'matrix' with first column set to the expected improvement statistics for `XX` locations, and the second column set to a ranking in the order that they should be sampled to minimize the expected multivariate improvement raised to a power determined by the argument `improv`
`response`	Name of response `Z` if supplied by `data.frame` in argument, or "z" if none provided
`parts`	Internal representation of the regions depicted by partitions of the maximum a' posteriori (MAP) tree
`trees`	`list` of trees (**maptree** representation) which were MAP as a function of each tree height sampled between MCMC rounds `B` and `T`
`trace`	If `trace==TRUE`, this `list` contains traces of most of the model parameters and posterior predictive distributions at input locations `XX`. Otherwise the entry is `FALSE`. See note below

ess	Importance tempering effective sample size (ESS). If itemps==NULL this corresponds to the total number of samples collected, i.e.. R*(BTE[2]-BTE[1])/BTE[3]. Otherwise the ESS will be lower due to a non-zero coefficient of variation of the calculated importance tempering weights
sens	See sens documentation for more details

Note

Inputs X, XX, Z containing NaN, NA, or Inf are discarded with non-fatal warnings

Upon execution, MCMC reports are made every 1,000 rounds to indicate progress

Stationary (non-treed) processes on larger inputs (e.g., X,Z) of size greater than 500, *might* be slow in execution, especially on older machines. Once the C code starts executing, it can be interrupted in the usual way: either via Ctrl-C (Unix-alikes) or pressing the Stop button in the R-GUI. When this happens, interrupt messages will indicate which required cleanup measures completed before returning control to R.

Whereas most of the **tgp** models will work reasonably well with little or no change to the default prior specification, GP's with the "mrexpsep" correlation imply a very specific relationship between fine and coarse data, and a careful prior specification is usually required.

The ranks provided in the second column of the improv field of a tgp object are based on the expectation of a multivariate improvement that may or may not be raised to a positive integer power. They can thus differ significantly from a simple ranking of the first column of expected univariate improvement values.

Regarding trace=TRUE: Samples from the posterior will be collected for all parameters in the model. GP parameters are collected with reference to the locations in XX, resulting nn=nrow{XX} traces of d,g,s2,tau2, etc. Therefore, it is recommended that nn is chosen to be a small, representative, set of input locations. Besides GP parameters, traces are saved for the tree partitions, areas under the LLM, log posterior (as a function of tree height), and samples from the posterior predictive distributions. Note that since some traces are stored in files, multiple tgp/R sessions in the same working directory can clobber the trace files of other sessions

Author(s)

Robert B. Gramacy, <rbg@vt.edu>, and Matt Taddy, <mataddy@amazon.com>

References

Gramacy, R. B. (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC. (See Chapter 9.) https://bobby.gramacy.com/surrogates/

Gramacy, R. B. (2007). **tgp***: An* R *Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models.* Journal of Statistical Software, **19**(9). https://www.jstatsoft.org/v19/i09 doi:10.18637/jss.v019.i09

Robert B. Gramacy, Matthew Taddy (2010). *Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with* **tgp** *Version 2, an* R *Package for Treed Gaussian Process Models.* Journal of Statistical Software, **33**(6), 1–48. https://www.jstatsoft.org/v33/i06/ doi:10.18637/jss.v033.i06

Gramacy, R. B., Lee, H. K. H. (2007). *Bayesian treed Gaussian process models with an application to computer modeling*. Journal of the American Statistical Association, 103(483), pp. 1119-1130. Also available as ArXiv article 0710.4536 https://arxiv.org/abs/0710.4536

Gramacy, R. B. and Lee, K.H. (2008). *Gaussian Processes and Limiting Linear Models.* Computational Statistics and Data Analysis, 53, pp. 123-136. Also available as ArXiv article 0804.4685 https://arxiv.org/abs/0804.4685

Gramacy, R. B., Lee, H. K. H. (2009). *Adaptive design and analysis of supercomputer experiments.* Technometrics, 51(2), pp. 130-145. Also avaliable on ArXiv article 0805.4359 https://arxiv.org/abs/0805.4359

Robert B. Gramacy, Heng Lian (2011). *Gaussian process single-index models as emulators for computer experiments*. Available as ArXiv article 1009.4241 https://arxiv.org/abs/1009.4241

Chipman, H., George, E., & McCulloch, R. (1998). *Bayesian CART model search (with discussion).* Journal of the American Statistical Association, **93**, 935–960.

Chipman, H., George, E., & McCulloch, R. (2002). *Bayesian treed models.* Machine Learning, **48**, 303–324.

M. Schonlau and Jones, D.R. and Welch, W.J. (1998). *Global versus local search in constrained optimization of computer models.* In "New Developments and applications in experimental design", IMS Lecture Notes - Monograph Series 34. 11–25.

https://bobby.gramacy.com/r_packages/tgp/

See Also

plot.tgp, tgp.trees, predict.tgp, sens, default.itemps

Examples

```
##
## Many of the examples below illustrate the above
## function(s) on random data.  Thus it can be fun
## (and informative) to run them several times.
##

#
# simple linear response
#

# input and predictive data
X <- seq(0,1,length=50)
XX <- seq(0,1,length=99)
Z <- 1 + 2*X + rnorm(length(X),sd=0.25)

out <- blm(X=X, Z=Z, XX=XX) # try Linear Model
plot(out)   # plot the surface

#
# 1-d Example
#

# construct some 1-d nonstationary data
```

```
X <- seq(0,20,length=100)
XX <- seq(0,20,length=99)
Z <- (sin(pi*X/5) + 0.2*cos(4*pi*X/5)) * (X <= 9.6)
lin <- X>9.6;
Z[lin] <- -1 + X[lin]/10
Z <- Z + rnorm(length(Z), sd=0.1)

out <- btlm(X=X, Z=Z, XX=XX)  # try Linear CART
plot(out)    # plot the surface
tgp.trees(out)     # plot the MAP trees

out <- btgp(X=X, Z=Z, XX=XX)  # use a treed GP
plot(out)    # plot the surface
tgp.trees(out)     # plot the MAP trees


#
# 2-d example
# (using the isotropic correlation function)
#

# construct some 2-d nonstationary data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
XX <- exp2d.data$XX

# try a GP
out <- bgp(X=X, Z=Z, XX=XX, corr="exp")
plot(out)    # plot the surface

# try a treed GP LLM
out <- btgpllm(X=X, Z=Z, XX=XX, corr="exp")
plot(out)    # plot the surface
tgp.trees(out)     # plot the MAP trees

#
# Motorcycle Accident Data
#

# get the data
require(MASS)

# try a GP
out <- bgp(X=mcycle[,1], Z=mcycle[,2])
plot(out)   # plot the surface

# try a treed GP LLM
# best to use the "b0" beta linear prior to capture common
# common linear process throughout all regions (using the
# ellipses "...")
out <- btgpllm(X=mcycle[,1], Z=mcycle[,2], bprior="b0")
plot(out)   # plot the surface
tgp.trees(out)    # plot the MAP trees
```

`default.itemps` *Default Sigmoidal, Harmonic and Geometric Temperature Ladders*

Description

Parameterized by the minimum desired *inverse* temperature, this function generates a ladder of inverse temperatures `k[1:m]` starting at `k[1] = 1`, with `m` steps down to the final temperature `k[m] = k.min` progressing sigmoidally, harmonically or geometrically. The output is in a format convenient for the `b*` functions in the **tgp** package (e.g. `btgp`), including stochastic approximation parameters c_0 and n_0 for tuning the uniform pseudo-prior output by this function

Usage

```
default.itemps(m = 40, type = c("geometric", "harmonic","sigmoidal"),
               k.min = 0.1, c0n0 = c(100, 1000), lambda = c("opt",
               "naive", "st"))
```

Arguments

`m`	Number of temperatures in the ladder; `m=1` corresponds to *importance sampling* at the temperature specified by `k.min` (in this case all other arguments are ignored)
`type`	Choose from amongst two common defaults for simulated tempering and Metropolis-coupled MCMC, i.e., geometric (default) or harmonic, or a sigmoidal ladder (default) that concentrates more inverse temperatures near 1
`k.min`	Minimum inverse temperature desired
`c0n0`	Stochastic approximation parameters used to tune the simulated tempering pseudo-prior (`$pk`) to get a uniform posterior over the inverse temperatures; must be a 2-vector of positive integers `c(c0, n0)`; see the Geyer & Thompson reference below
`lambda`	Method for combining the importance samplers at each temperature. Optimal combination (`"opt"`) is the default, weighting the IS at each temperature k by $$\lambda_k \propto (\sum_i w_{ki})^2 / \sum_i w_{ki}^2.$$ Setting `lambda = "naive"` allows each temperature to contribute equally ($\lambda_k \propto$ 1, or equivalently ignores delineations due to temperature when using importance weights. Setting `lambda = "st"` allows only the first (cold) temperature to contribute to the estimator, thereby implementing *simulated tempering*

Details

The geometric and harmonic inverse temperature ladders are usually defined by an index $i = 1, \ldots, m$ and a parameter $\Delta_k > 0$. The geometric ladder is defined by

$$k_i = (1 + \Delta_k)^{1-i},$$

and the harmonic ladder by

$$k_i = (1 + \Delta_k(i-1))^{-1}.$$

Alternatively, specifying the minimum temperature k_{min} in the ladder can be used to uniquely determine Δ_k. E.g., for the geometric ladder

$$\Delta_k = k_{\text{min}}^{1/(1-m)} - 1,$$

and for the harmonic

$$\Delta_k = \frac{k_{\text{min}}^{-1} - 1}{m-1}.$$

In a similar spirit, the sigmoidal ladder is specified by first situating m indices $j_i \in \Re$ so that $k_1 = k(j_1) = 1$ and $k_m = k(j_m) = k_{\text{min}}$ under

$$k(j_i) = 1.01 - \frac{1}{1 + e^{j_i}}.$$

The remaining $j_i, i = 2, \ldots, (m-1)$ are spaced evenly between j_1 and j_m to fill out the ladder $k_i = k(j_i), i = 1, \ldots, (m-1)$.

For more details, see the *Importance tempering* paper cited below and a full demonstration in `vignette("tgp2")`

Value

The return value is a `list` which is compatible with the input argument `itemps` to the `b*` functions (e.g. `btgp`), containing the following entries:

`c0n0`	A copy of the `c0n0` input argument
`k`	The generated inverse temperature ladder; a vector with `length(k) = m` containing a decreasing sequence from `1` down to `k.min`
`pk`	A vector with `length(pk) = m` containing an initial pseudo-prior for the temperature ladder of `1/m` for each inverse temperature
`lambda`	IT method, as specified by the input argument

Author(s)

Robert B. Gramacy, <`rbg@vt.edu`>, and Matt Taddy, <`mataddy@amazon.com`>

References

Gramacy, R.B., Samworth, R.J., and King, R. (2010) *Importance Tempering.* ArXiV article 0707.4242 Statistics and Computing, 20(1), pp. 1-7; https://arxiv.org/abs/0707.4242.

For stochastic approximation and simulated tempering (ST):

Geyer, C.~and Thompson, E.~(1995). *Annealing Markov chain Monte Carlo with applications to ancestral inference.* Journal of the American Statistical Association, **90**, 909–920.

For the geometric temperature ladder:

Neal, R.M.~(2001) *Annealed importance sampling.* Statistics and Computing, **11**, 125–129

Justifying geometric and harmonic defaults:

Liu, J.S.~(1002) *Monte Carlo Strategies in Scientific Computing.* New York: Springer. Chapter 10 (pages 213 & 233)

https://bobby.gramacy.com/r_packages/tgp/

See Also

btgp

Examples

```
## comparing the different ladders
geo <- default.itemps(type="geometric")
har <- default.itemps(type="harmonic")
sig <- default.itemps(type="sigmoidal")
par(mfrow=c(2,1))
matplot(cbind(geo$k, har$k, sig$k), pch=21:23,
        main="inv-temp ladders", xlab="indx",
        ylab="itemp")
legend("topright", pch=21:23,
       c("geometric","harmonic","sigmoidal"), col=1:3)
matplot(log(cbind(sig$k, geo$k, har$k)), pch=21:23,
        main="log(inv-temp) ladders", xlab="indx",
        ylab="itemp")

## Not run:
## using Importance Tempering (IT) to improve mixing
## on the motorcycle accident dataset
library(MASS)
out.it <- btgpllm(X=mcycle[,1], Z=mcycle[,2], BTE=c(2000,22000,2),
        R=3, itemps=default.itemps(), bprior="b0", trace=TRUE,
        pred.n=FALSE)

## compare to regular tgp w/o IT
out.reg <- btgpllm(X=mcycle[,1], Z=mcycle[,2], BTE=c(2000,22000,2),
        R=3, bprior="b0", trace=TRUE, pred.n=FALSE)

## compare the heights explored by the three chains:
## REG, combining all temperatures, and IT
p <- out.it$trace$post
L <- length(p$height)
```

```
hw <- suppressWarnings(sample(p$height, L, prob=p$wlambda, replace=TRUE))
b <- hist2bar(cbind(out.reg$trace$post$height, p$height, hw))
par(mfrow=c(1,1))
barplot(b, beside=TRUE, xlab="tree height", ylab="counts", col=1:3,
        main="tree heights encountered")
legend("topright", c("reg MCMC", "All Temps", "IT"), fill=1:3)

## End(Not run)
```

---

dopt.gp *Sequential D-Optimal Design for a Stationary Gaussian Process*

Description

Create sequential D-Optimal design for a stationary Gaussian process model of fixed parameterization by subsampling from a list of candidates

Usage

```
dopt.gp(nn, X=NULL, Xcand, iter=5000, verb=0)
```

Arguments

`nn`	Number of new points in the design. Must be less than or equal to the number of candidates contained in `Xcand`, i.e., `nn <= nrow(Xcand)`
`X`	`data.frame`, `matrix` or vector of input locations which are forced into (already in) the design
`Xcand`	`data.frame`, `matrix` or vector of candidates from which new design points are subsampled. Must have the same dimension as `X`, i.e., `ncol(X) == ncol(Xcand)`
`iter`	number of iterations of stochastic accent algorithm, default `5000`
`verb`	positive integer indicating after how many rounds of stochastic approximation to print each progress statement; default `verb=0` results in no printing

Details

Design is based on a stationary Gaussian process model with stationary isotropic exponential correlation function with parameterization fixed as a function of the dimension of the inputs. The algorithm implemented is a simple stochastic ascent which maximizes `det(K)`– the covariance matrix constructed with locations `X` and a subset of `Xcand` of size `nn`. The selected design is *locally* optimal

Value

The output is a list which contains the inputs to, and outputs of, the C code used to find the optimal design. The chosen design locations can be accessed as list members `XX` or equivalently `Xcand[fi,]`.

`X`	Input argument: `data.frame` of inputs `X`, can be `NULL`
`nn`	Input argument: number new points in the design
`Xcand`	Input argument: `data.frame` of candidate locations `Xcand`
`ncand`	Number of rows in `Xcand`, i.e., `nncand` = `dim(Xcand)[1]`
`fi`	Vector of length `nn` describing the selected new design locations as indices into `Xcand`
`XX`	`data.frame` of selected new design locations, i.e., `XX` = `Xcand[fi,]`

Note

Inputs `X, Xcand` containing `NaN, NA, Inf` are discarded with non-fatal warnings. If `nn` > `dim(Xcand)[1]` then a non-fatal warning is displayed and execution commences with `nn` = `dim(Xcand)[1]`

In the current version there is no progress indicator. You will have to be patient. Creating D-optimal designs is no speedy task

Author(s)

Robert B. Gramacy, `<rbg@vt.edu>`, and Matt Taddy, `<mataddy@amazon.com>`

References

Gramacy, R. B. (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC. (See Chapter 6.) `https://bobby.gramacy.com/surrogates/`

Chaloner, K. and Verdinelli, I. (1995). *Bayesian experimental design: A review.* Statist. Sci., 10, (pp. 273–304).

See Also

`tgp.design`, `lhs`

Examples

```
#
# 2-d Exponential data
# (This example is based on random data.
# It might be fun to run it a few times)
#

# get the data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
Xcand <- exp2d.data$XX
```

```
# find a treed sequential D-Optimal design
# with 10 more points
dgp <- dopt.gp(10, X, Xcand)

# plot the d-optimally chosen locations
# Contrast with locations chosen via
# the tgp.design function
plot(X, pch=19, xlim=c(-2,6), ylim=c(-2,6))
points(dgp$XX)
```

`exp2d` *2-d Exponential Data*

Description

A 2-dimensional data set that can be used to validate non-stationary models.

Usage

```
data(exp2d)
```

Format

A `data frame` with 441 observations on the following 4 variables.

`X1` Numeric vector describing the first dimension of `X` inputs

`X2` Numeric vector describing the second dimension of `X` inputs

`Z` Numeric vector describing the response `Z(X)+N(0,sd=0.001)`

`Ztrue` Numeric vector describing the true response `Z(X)`, without noise

Details

The true response is evaluated as

$$Z(X) = x_1 * \exp(x_1^2 - x_2^2).$$

Zero-mean normal noise with `sd=0.001` has been added to the true response

Note

This data is used in the examples of the functions listed below in the “See Also” section via the `exp2d.rand` function

Author(s)

Robert B. Gramacy, <`rbg@vt.edu`>, and Matt Taddy, <`mataddy@amazon.com`>

References

Gramacy, R. B. (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC. https://bobby.gramacy.com/surrogates/

Gramacy, R. B. (2007). **tgp***: An* R *Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models.* Journal of Statistical Software, **19**(9). https://www.jstatsoft.org/v19/i09 doi:10.18637/jss.v019.i09

Robert B. Gramacy, Matthew Taddy (2010). *Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with* **tgp** *Version 2, an* R *Package for Treed Gaussian Process Models.* Journal of Statistical Software, **33**(6), 1–48. https://www.jstatsoft.org/v33/i06/. doi:10.18637/jss.v033.i06

Gramacy, R. B., Lee, H. K. H. (2008). *Bayesian treed Gaussian process models with an application to computer modeling*. Journal of the American Statistical Association, 103(483), pp. 1119-1130. Also available as ArXiv article 0710.4536 https://arxiv.org/abs/0710.4536

https://bobby.gramacy.com/r_packages/tgp/

See Also

`exp2d.rand`, `exp2d.Z`, `btgp`, and other `b*` functions

`exp2d.rand` *Random 2-d Exponential Data*

Description

A Random subsample of `data(exp2d)`, or Latin Hypercube sampled data evaluated with `exp2d.Z`

Usage

```
exp2d.rand(n1 = 50, n2 = 30, lh = NULL, dopt = 1)
```

Arguments

`n1`	Number of samples from the first, interesting, quadrant
`n2`	Number of samples from the other three, uninteresting, quadrants
`lh`	If `!is.null(lh)` then Latin Hypercube (LH) sampling (`lhs`) is used instead of subsampling from `data(exp2d)`; `lh` should be a single nonnegative integer specifying the desired number of predictive locations, `XX`; or, it should be a vector of length 4, specifying the number of predictive locations desired from each of the four quadrants (interesting quadrant first, then counter-clockwise)
`dopt`	If `dopt` >= `2` then d-optimal subsampling from LH candidates of the multiple indicated by the value of `dopt` will be used. This argument only makes sense when `!is.null(lh)`

Details

When is.null(lh), data is subsampled without replacement from data(exp2d). Of the n1 + n2 <= 441 input/response pairs X,Z, there are n1 are taken from the first quadrant, i.e., where the response is interesting, and the remaining n2 are taken from the other three quadrants. The remaining 441 - (n1 + n2) are treated as predictive locations

Otherwise, when !is.null(lh), Latin Hypercube Sampling (lhs) is used

If dopt >= 2 then n1*dopt LH candidates are used for to get a D-optimal subsample of size n1 from the first (interesting) quadrant. Similarly n2*dopt in the rest of the un-interesting region. A total of lh*dopt candidates will be used for sequential D-optimal subsampling for predictive locations XX in all four quadrants assuming the already-sampled X locations will be in the design.

In all three cases, the response is evaluated as

$$Z(X) = x_1 * \exp(x_1^2 - x_2^2).$$

thus creating the outputs Ztrue and ZZtrue. Zero-mean normal noise with sd=0.001 is added to the responses Z and ZZ

Value

Output is a list with entries:

X	2-d data.frame with n1 + n2 input locations
Z	Numeric vector describing the responses (with noise) at the X input locations
Ztrue	Numeric vector describing the true responses (without noise) at the X input locations
XX	2-d data.frame containing the remaining 441 - (n1 + n2) input locations
ZZ	Numeric vector describing the responses (with noise) at the XX predictive locations
ZZtrue	Numeric vector describing the responses (without noise) at the XX predictive locations

Author(s)

Robert B. Gramacy, <rbg@vt.edu>, and Matt Taddy, <mataddy@amazon.com>

References

Gramacy, R. B. (2007). **tgp***: An* R *Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models.* Journal of Statistical Software, **19**(9). https://www.jstatsoft.org/v19/i09 doi:10.18637/jss.v019.i09

Robert B. Gramacy, Matthew Taddy (2010). *Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with* **tgp** *Version 2, an* R *Package for Treed Gaussian Process Models.* Journal of Statistical Software, **33**(6), 1–48. https://www.jstatsoft.org/v33/i06/ doi:10.18637/jss.v033.i06

Gramacy, R. B., Lee, H. K. H. (2008). *Bayesian treed Gaussian process models with an application to computer modeling*. Journal of the American Statistical Association, 103(483), pp. 1119-1130. Also available as ArXiv article 0710.4536 https://arxiv.org/abs/0710.4536

https://bobby.gramacy.com/r_packages/tgp/

See Also

lhs, exp2d, exp2d.Z, btgp, and other b* functions

Examples

```
## randomly subsampled data
## ------------------------

eds <- exp2d.rand()

# higher span = 0.5 required because the data is sparse
# and was generated randomly
eds.g <- interp.loess(eds$X[,1], eds$X[,2], eds$Z, span=0.5)

# perspective plot, and plot of the input (X & XX) locations
par(mfrow=c(1,2), bty="n")
persp(eds.g, main="loess surface", theta=-30, phi=20,
      xlab="X[,1]", ylab="X[,2]", zlab="Z")
plot(eds$X, main="Randomly Subsampled Inputs")
points(eds$XX, pch=19, cex=0.5)

## Latin Hypercube sampled data
## ----------------------------

edlh <- exp2d.rand(lh=c(20, 15, 10, 5))

# higher span = 0.5 required because the data is sparse
# and was generated randomly
edlh.g <- interp.loess(edlh$X[,1], edlh$X[,2], edlh$Z, span=0.5)

# perspective plot, and plot of the input (X & XX) locations
par(mfrow=c(1,2), bty="n")
persp(edlh.g, main="loess surface", theta=-30, phi=20,
      xlab="X[,1]", ylab="X[,2]", zlab="Z")
plot(edlh$X, main="Latin Hypercube Sampled Inputs")
points(edlh$XX, pch=19, cex=0.5)

# show the quadrants
abline(h=2, col=2, lty=2, lwd=2)
abline(v=2, col=2, lty=2, lwd=2)


## Not run:
## D-optimal subsample with a factor of 10 (more) candidates
## ---------------------------------------------------------

edlhd <- exp2d.rand(lh=c(20, 15, 10, 5), dopt=10)

# higher span = 0.5 required because the data is sparse
# and was generated randomly
edlhd.g <- interp.loess(edlhd$X[,1], edlhd$X[,2], edlhd$Z, span=0.5)
```

```
# perspective plot, and plot of the input (X & XX) locations
par(mfrow=c(1,2), bty="n")
persp(edlhd.g, main="loess surface", theta=-30, phi=20,
      xlab="X[,1]", ylab="X[,2]", zlab="Z")
plot(edlhd$X, main="D-optimally Sampled Inputs")
points(edlhd$XX, pch=19, cex=0.5)

# show the quadrants
abline(h=2, col=2, lty=2, lwd=2)
abline(v=2, col=2, lty=2, lwd=2)

## End(Not run)
```

---

exp2d.Z *Random Z-values for 2-d Exponential Data*

Description

Evaluate the functional (mean) response for the 2-d exponential data (truth) at the `X` inputs, and randomly sample noisy `Z`–values having normal error with standard deviation provided.

Usage

```
exp2d.Z(X, sd=0.001)
```

Arguments

`X`	Must be a `matrix` or a `data.frame` with two columns describing input locations
`sd`	Standard deviation of iid normal noise added to the responses

Details

The response is evaluated as

$$Z(X) = x_1 * \exp(x_1^2 - x_2^2).$$

thus creating the outputs `Z` and `Ztrue`. Zero-mean normal noise with `sd=0.001` is added to the responses `Z` and `ZZ`

Value

Output is a `data.frame` with columns:

`Z`	Numeric vector describing the responses (with noise) at the `X` input locations
`Ztrue`	Numeric vector describing the true responses (without noise) at the `X` input locations

Author(s)

Robert B. Gramacy, <rbg@vt.edu>, and Matt Taddy, <mataddy@amazon.com>

References

Gramacy, R. B. (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC. `https://bobby.gramacy.com/surrogates/`

Gramacy, R. B. (2007). **tgp***: An* R *Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models.* Journal of Statistical Software, **19**(9). `https://www.jstatsoft.org/v19/i09` doi:10.18637/jss.v019.i09

Robert B. Gramacy, Matthew Taddy (2010). *Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with* **tgp** *Version 2, an* R *Package for Treed Gaussian Process Models.* Journal of Statistical Software, **33**(6), 1–48. `https://www.jstatsoft.org/v33/i06/` doi:10.18637/jss.v033.i06

Gramacy, R. B., Lee, H. K. H. (2008). *Bayesian treed Gaussian process models with an application to computer modeling*. Journal of the American Statistical Association, 103(483), pp. 1119-1130. Also available as ArXiv article 0710.4536 `https://arxiv.org/abs/0710.4536`

`https://bobby.gramacy.com/r_packages/tgp/`

See Also

`exp2d`, `exp2d.rand`

Examples

```
N <- 20
x <- seq(-2,6,length=N)
X <- expand.grid(x, x)
Zdata <- exp2d.Z(X)
persp(x,x,matrix(Zdata$Ztrue, nrow=N), theta=-30, phi=20,
      main="Z true", xlab="x1", ylab="x2", zlab="Ztrue")
```

`friedman.1.data` *First Friedman Dataset and a variation*

Description

Generate X and Y values from the 10-dim “first” Friedman data set used to validate the Multivariate Adaptive Regression Splines (MARS) model, and a variation involving boolean indicators. This test function has three non-linear and interacting variables, along with two linear, and five which are irrelevant. The version with indicators has parts of the response turned on based on the setting of the indicators

Usage

```
friedman.1.data(n = 100)
fried.bool(n = 100)
```

Arguments

`n`	Number of samples desired

Details

In the original formulation, as implemented by `friedman.1.data` the function has 10-dim inputs `X` are drawn from Unif(0,1), and responses are $N(m(X), 1)$ where $m(\mathbf{x}) = E[f(\mathbf{x})]$ and

$$m(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$$

The variation `fried.bool` uses indicators $I \in \{1, 2, 3, 4\}$. The function also has 10-dim inputs `X` with columns distributed as Unif(0,1) and responses are $N(m(\mathbf{x}, I), 1)$ where $m(\mathbf{x}, I) = E(f(\mathbf{x}, I)$ and

$$m(\mathbf{x}, I) = f_1(\mathbf{x})_{[I=1]} + f_2(\mathbf{x})_{[I=2]} + f_3(\mathbf{x})_{[I=3]} + m([x_{10}, \cdots, x_1])_{[I=4]}$$

where

$$f_1(\mathbf{x}) = 10\sin(\pi x_1 x_2),\ f_2(\mathbf{x}) = 20(x_3 - 0.5)^2,\ \text{and } f_3(\mathbf{x}) = 10x_4 + 5x_5.$$

The indicator I is coded in binary in the output data frame as: `c(0,0,0)` for `I=1`, `c(0,0,1)` for `I=2`, `c(0,1,0)` for `I=3`, and `c(1,0,0)` for `I=4`.

Value

Output is a `data.frame` with columns

`X.1, ..., X.10`	describing the 10-d randomly sampled inputs
`I.1, ..., I.3`	boolean version of the indicators provided only for `fried.bool`, as described above
`Y`	sample responses (with N(0,1) noise)
`Ytrue`	true responses (without noise)

Note

An example using the original version of the data (`friedman.1.data`) is contained in the first package vignette: `vignette("tgp")`. The boolean version `fried.bool` is used in second vignette `vignette("tgp2")`

Author(s)

Robert B. Gramacy, <`rbg@vt.edu`>, and Matt Taddy, <`mataddy@amazon.com`>

References

Gramacy, R. B. (2007). **tgp***: An* R *Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models.* Journal of Statistical Software, **19**(9). `https://www.jstatsoft.org/v19/i09` doi:10.18637/jss.v019.i09

Robert B. Gramacy, Matthew Taddy (2010). *Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with* **tgp** *Version 2, an* R *Package for Treed Gaussian Process*

Models. Journal of Statistical Software, **33**(6), 1–48. https://www.jstatsoft.org/v33/i06/. doi:10.18637/jss.v033.i06

Friedman, J. H. (1991). *Multivariate adaptive regression splines.* "Annals of Statistics", **19**, No. 1, 1–67.

Gramacy, R. B., Lee, H. K. H. (2008). *Bayesian treed Gaussian process models with an application to computer modeling*. Journal of the American Statistical Association, 103(483), pp. 1119-1130. Also available as ArXiv article 0710.4536 https://arxiv.org/abs/0710.4536

Chipman, H., George, E., & McCulloch, R. (2002). *Bayesian treed models.* Machine Learning, **48**, 303–324.

https://bobby.gramacy.com/r_packages/tgp/

See Also

bgpllm, btlm, blm, bgp, btgpllm, bgp

---

interp.loess *Lowess 2-d interpolation onto a uniform grid*

Description

Use the loess function to interpolate the two-dimensional x, y, and z data onto a uniform grid. The output produced is an object directly usable by the plotting functions persp, image, and contour, etc.

This function is designed as an alternative to the interp functions from the **akima** library.

Usage

```
interp.loess(x, y, z, gridlen = c(40,40), span = 0.1, ...)
```

Arguments

x	Vector of X spatial input locations
y	Vector of Y spatial input locations
z	Vector of Z responses interpreted as Z = f(X,Y)
gridlen	Size of the interpolated grid to be produced in x and y. The default of gridlen = c(40,40) causes a 40 * 40 grid of X, Y, and Z values to be computed.
span	Kernel span argument to the loess function with default setting span = 0.1 set significantly lower than the the loess default – see note below.
...	Further arguments to be passed to the loess function

Details

Uses expand.grid function to produce a uniform grid of size gridlen with domain equal to the rectangle implied by X and Y. Then, a loess a smoother is fit to the data Z = f(X,Y). Finally, predict.loess is used to predict onto the grid.

Value

The output is a list compatible with the 2-d plotting functions `persp`, `image`, and `contour`, etc.

The list contains...

`x`	Vector of with `length(x) == gridlen` of increasing `X` grid locations
`y`	Vector of with `length(y) == gridlen` of increasing `Y` grid locations
`z`	`matrix` of interpolated responses `Z = f(X,Y)` where `z[i,j]` contains an estimate of `f(x[i],y[j])`

Note

As mentioned above, the default `span = 0.1` parameter is significantly smaller that the default `loess` setting. This asserts a tacit assumption that the input is densely packed and that the noise in `z`'s is small. Such should be the case when the data are output from a **tgp** regression – this function was designed specifically for this situation. For data that is random or sparse, simply choose higher setting, e.g., the default `loess` setting of `span = 0.75`, or a more intermediate setting of `span = 0.5` as in the example below

Author(s)

Robert B. Gramacy, `<rbg@vt.edu>`, and Matt Taddy, `<mataddy@amazon.com>`

References

https://bobby.gramacy.com/r_packages/tgp/

See Also

`interp`, `loess`, `persp`, `image`, `contour`

Examples

```
# random data
ed <- exp2d.rand()

# higher span = 0.5 required because the data is sparse
# and was generated randomly
ed.g <- interp.loess(ed$X[,1], ed$X[,2], ed$Z, span=0.5)

# perspective plot
persp(ed.g)
```

itemps	*Functions to plot summary information about the sampled inverse temperatures, tree heights, etc., stored in the traces of a "tgp"-class object*

Description

Functions for making barplots summarizing the progress of importance tempering. The `itemps.barplot` function can be used to make a histogram of the inverse temperatures visited in the trans-temporal Markov chain. The `hist2bar` function is useful for making a histogram of integer-valued samples (e.g., tree heights) encountered in one or several Markov chains

Usage

```
itemps.barplot(obj, main = NULL, xlab = "itemps",
               ylab = "counts", plot.it = TRUE, ...)
hist2bar(x)
```

Arguments

`obj`	`"tgp"`-class object
`main`	Main plot label to be augmented by `itemps.barplot`
`xlab`	Label for the x-axis
`ylab`	Label for the y-axis
`plot.it`	whether to plot the `barplot` in addition to returning the `data.frame` for later use in a `barplot` call
`...`	other arguments passed to `barplot` if `plot.it = TRUE`
`x`	`matrix` of integers whose columns are treated as different realizations of similar processes producing where each row represents a sample (e.g., tree height) under that process

Details

`itemps.barplot` specifically works with the `$trace` field of a `"tgp"`-class object. An error will be produced if this field is `NULL`, i.e., if the `b*` function used the create the object was not run with the argument `trace=TRUE`

The `hist2bar` function can be used on any integer (or discrete) valued matrix. The columns are interpreted as different realizations of similar processes for comparison with one another via a histogram.

The histogram is obtained with the `barplot` command used with the argument `beside=TRUE`. See the examples section of `default.itemps`

Value

Both functions return a `data.frame` that can be used within the `barplot` function with argument `beside=TRUE`

Author(s)

Robert B. Gramacy, `<rbg@vt.edu>`, and Matt Taddy, `<mataddy@amazon.com>`

References

Gramacy, R.B., Samworth, R.J., and King, R. (2007) *Importance Tempering.* ArXiv article 0707.4242 `https://arxiv.org/abs/0707.4242`

`https://bobby.gramacy.com/r_packages/tgp/`

See Also

`default.itemps`, `vignette(tgp2)`, `barplot`

`lhs` *Latin Hypercube sampling*

Description

Draw a (random) Latin Hypercube (LH) sample of size `n` from in the region outlined by the provided rectangle

Usage

```
lhs(n, rect, shape=NULL, mode=NULL)
```

Arguments

`n`	Size of the LH sample
`rect`	Rectangle describing the domain from which the LH sample is to be taken. The rectangle should be a `matrix` or `data.frame` with `ncol(rect) = 2`, and number of rows equal to the dimension of the domain. For 1-d data, a vector of length 2 is allowed
`shape`	Optional vector of shape parameters for the Beta distribution. Vector of length equal to the dimension of the domain, with elements > 1. If this is specified, the LH sample is proportional to a joint pdf formed by independent Beta distributions in each dimension of the domain, scaled and shifted to have support defined by `rect`. Only concave Beta distributions with `shape` > 1 are supported.
`mode`	Optional vector of mode values for the Beta distribution. Vector of length equal to the dimension of the domain, with elements within the support defined by `rect`. If `shape` is specified, but this is not, then the scaled Beta distributions will be symmetric

Value

The output is a `matrix` with `n` rows and `nrow(rect)` columns. Each of the `n` rows represents a sample point.

Note

The domain bounds specified by the rows of `rect` can be specified backwards with no change in effect.

Author(s)

Robert B. Gramacy, `<rbg@vt.edu>`, and Matt Taddy, `<mataddy@amazon.com>`

References

Gramacy, R. B. (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC. (See Chapter 4.) https://bobby.gramacy.com/surrogates/

McKay, M. D., W. J. Conover and R. J. Beckman. (1979). *A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code*, Technometrics 21: (pp. 239–245).

See Also

`tgp.design`, `dopt.gp`, `sens`

Examples

```
# get and plot a 2-d LH design
s1 <- lhs(10, rbind(c(-2,3), c(0.5, 0.8)))
plot(s1)

# plot a grid to show that there is one sample
# in each grid location
abline(v=seq(-2,3,length=11), lty=2, col=3)
abline(h=seq(0.5,0.8,length=11), lty=2, col=3)
```

`mapT` *Plot the MAP partition, or add one to an existing plot*

Description

Plot the maximum a' posteriori (MAP) tree from a `"tgp"`-class object, or add one on top of an existing plot. Like `plot.tgp`, projections and slices of trees can be plotted as specified

Usage

```
mapT(out, proj = NULL, slice = NULL, add = FALSE, lwd = 2, ...)
```

Arguments

`out`	`"tgp"`-class object which is the output of one the model functions with tree support (e.g. `btgpllm`)
`proj`	1-or-2-Vector describing the dimensions to be shown in a projection. The argument is ignored for 1-d data, i.e., if `x$d == 1`. For 2-d data, no projection needs to be specified— the default argument (`proj = NULL`) will result in a 2-d plot. 1-d projections of 2-d or higher trees are are supported, e.g., `proj = c(2)` would show the second variable projection. For 3-d data or higher, `proj=NULL` defaults to `proj = c(1,2)` which plots a 2-d projection of the trees for the first two variables. Slices have priority over projections— see next argument (`slice`)— when non-null arguments are provided for both.
`slice`	`list` object with `x` and `z` fields, which are vectors of equal length describing the slice to be plotted, i.e., which z-values of the treed partitions in the `x$d - 2` inputs `x$X` and `x$XX` should be fixed to in order to obtain a 2-d visualization. For example, for 4-d data, `slice = list(x=(2,4), z=c(0.2, 1.5)` will result in a 2-d plot of the first and third dimensions which have the second and fourth slice fixed at 0.5 and 1.5. The default is `NULL`, yielding to the `proj` argument. Argument is ignored for 1-d data, i.e., if `x$d == 1`
`add`	Specify whether the to add partitions to an existing plot (`add = TRUE`) or to make a new plot showing the data `out$X` along with the partitions (default `add = FALSE`)
`lwd`	Plotting argument specifying the width of the lines used to depict the partitions
`...`	Additional arguments to `plot` used when `add = FALSE`

Value

The only output of this function is a beautiful region-representation of the MAP tree.

Note

For examples, see `vignette("tgp")` and the examples provided in the documentation for the `tgp.design` function

Author(s)

Robert B. Gramacy, <rbg@vt.edu>, and Matt Taddy, <mataddy@amazon.com>

References

https://bobby.gramacy.com/r_packages/tgp/

See Also

`plot.tgp`, `tgp.trees`, `tgp.design`, `vignette("tgp")` optim.tgp *Surrogate-based optimization of noisy black-box function*

Description

Optimize (minimize) a noisy black-box function (i.e., a function which may not be differentiable, and may not execute deterministically). A `b*` **tgp** model is used as a surrogate for adaptive sampling via improvement (and other) statistics. Note that this function is intended as a skeleton to be tailored as required for a particular application

Usage

```
optim.step.tgp(f, rect, model = btgp, prev = NULL, X = NULL,
    Z = NULL, NN = 20 * length(rect), improv = c(1, 5), cands = c("lhs", "tdopt"),
   method = c("L-BFGS-B", "Nelder-Mead", "BFGS", "CG", "SANN",  "optimize"), ...)
optim.ptgpf(start, rect, tgp.obj,
     method=c("L-BFGS-B", "Nelder-Mead", "BFGS", "CG", "SANN", "optimize"))
```

Arguments

`f`	A function to be optimized, having only one free argument
`rect`	`matrix` indicating the domain of the argument of `f` over which an optimal should be searched; must have `ncol(rect) = 2` and `nrow` agreeing with the argument of `f` indicating the dimension of the data. For 1-d data, a vector of length 2 is allowed
`model`	The `b*` regression model used as a surrogate for optimization; see `btgp`, and others, for more detail
`prev`	The output from a previous call to `optim.step.tgp`; this should be a `list` with entries as described the "Value" section below
`X`	`data.frame`, `matrix`, or vector of current inputs `X`, to be augmented
`Z`	Vector of current output responses `Z` of length equal to the leading dimension (rows) of `X`, i.e., `length(Z) == nrow(X)`, to be augmented
`NN`	Number of candidate locations (`XX`) at which to sample from the improvement statistic
`improv`	Indicates the `improv` argument provided to a `b* model` function for sampling from the improvement statistic at the `NN` candidate locations (`XX`); see `btgp`, and others, for more detail
`cands`	The type of candidates (`XX`) at which samples from the improvement statistics are gathered. The default setting of `"lhs"` is recommended. However, a sequential treed D-optimal design can be used with `"tdopt"` for a more global exploration; see `tgp.design` for more details
`method`	A method from `optim`, or `"optimize"` which uses `optimize` as appropriate (when the input-space is 1-d)
`...`	Further arguments to the `b* model` function

`start`	A starting value for optimization of the MAP predictive (kriging) surface of a `"tgp"`-class object. A good starting value is the `X` or `XX` location found to be a minimum in the mean predictive surface contained in `"tgp"`-class object
`tgp.obj`	A `"tgp"`-class object that is the output of one of the `b*` functions: `blm`, `btlm` `bgp`, `bgpllm`, `btgp`, or `btgpllm`, as can be used by `predict.tgp` for optimizing on the MAP predictive (surrogate) kriging surface

Details

`optim.step.tgp` executes one step in a search for the global optimum (minimum) of a noisy function (`Z~f(X)`) in a bounded rectangle (`rect`). The procedure essentially fits a tgp `model` and samples from the posterior distribution of improvement statistics at `NN+1` candidates locations. `NN` of the candidates come from `cands`, i.e., `"lhs"` or `"tdopt"`, plus one which is the location of the minima found in a previous run via `prev` by using `optim` (with a particular `method` or `optimize` instead) on the MAP `model` predictive surface using the `"tgp"`-class object contained therein. The `improv[2]` with the the highest expected improvement are recommended for adding into the design on output.

`optim.ptgpf` is the subroutine used by `optim.step.tgp` to find optimize on the MAP (surrogate) predictive surface for the `"tgp"`-class object contained in `prev`.

Please see `vignette("tgp2")` for a detailed illustration

Value

The `list` return has the following components.

`X`	A `matrix` with `nrow(rect)` columns whose rows contain recommendations for input locations to add into the design
`progress`	A one-row `data.frame` indicating the the `X`-location and objective value of the current best guess of the solution to the (kriging) surrogate optimization along with the maximum values of the improvement statistic
`obj`	the `"tgp"`-class object output from the `model` function

Note

The ellipses (`...`) argument is used differently here, as compared to `optim`, and `optimize`. It allows further arguments to be passed to the `b*` `model` function, whereas for `optim` it would describe further (static) arguments to the function `f` to be optimized. If static arguments need to be set for `f`, then we recommend setting defaults via the `formals` of `f`

Author(s)

Robert B. Gramacy, `<rbg@vt.edu>`, and Matt Taddy, `<mataddy@amazon.com>`

References

Gramacy, R. B. (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC. (See Chapter 7.) `https://bobby.gramacy.com/surrogates/`

Matthew Taddy, Herbert K.H. Lee, Genetha A. Gray, and Joshua D. Griffin. (2009) *Bayesian guided pattern search for robust local optimization.* Technometrics, 51(4), pp. 389-401

https://bobby.gramacy.com/r_packages/tgp/

See Also

btgp, etc., optim, optimize, tgp.design, predict.tgp, dopt.gp

Examples

```
## optimize the simple exponential function
f <- function(x) { exp2d.Z(x)$Z }

## create the initial design with D-optimal candidates
rect <- rbind(c(-2,6), c(-2,6))
Xcand <- lhs(500, rect)
X <- dopt.gp(50, X=NULL, Xcand)$XX
Z <- f(X)

## do 10 rounds of adaptive sampling
out <- progress <- NULL
for(i in 1:10) {

  ## get recommendations for the next point to sample
  out <- optim.step.tgp(f, X=X, Z=Z, rect=rect, prev=out)

  ## add in the inputs, and newly sampled outputs
  X <- rbind(X, out$X)
  Z <- c(Z, f(out$X))

  ## keep track of progress and best optimum
  progress <- rbind(progress, out$progress)
  print(progress[i,])
}

## plot the progress so far
par(mfrow=c(2,2))
plot(out$obj, layout="surf")
plot(out$obj, layout="as", as="improv")
matplot(progress[,1:nrow(rect)], main="optim results",
        xlab="rounds", ylab="x[,1:2]", type="l", lwd=2)
plot(log(progress$improv), type="l", main="max log improv",
     xlab="rounds", ylab="max log(improv)")
```

---

partition *Partition data according to the MAP tree*

Description

Partition data according to the maximum a' posteriori (MAP) tree contained in a "tgp"-class object.

Usage

```
partition(X, out)
```

Arguments

`X`	`data.frame`, `matrix`, or vector of inputs `X` with the same dimension of `out$X`, i.e., `ncol(X) == ncol(out$X)`
`out`	`"tgp"`-class object which is the output of one the model functions with tree support (e.g. `btgpllm`, `btgp`, `btlm`)

Value

Output is a list of `data.frames` populated with the inputs `X` contained in each region of the partition of the MAP tree in the `"tgp"`-class object `out`

Author(s)

Robert B. Gramacy, <`rbg@vt.edu`>, and Matt Taddy, <`mataddy@amazon.com`>

References

`https://bobby.gramacy.com/r_packages/tgp/`

See Also

`tgp.design`, `tgp.trees`

Examples

```
#
# 2-d Exponential data
# (This example is based on random data.
# It might be fun to run it a few times)
#

# get the data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
Xcand <- exp2d.data$XX

# fit treed GP LLM model to data w/o prediction
# basically just to get MAP tree (and plot it)
out <- btgpllm(X=X, Z=Z, pred.n=FALSE, BTE=c(2000,3000,2))
tgp.trees(out)

# find a treed sequential D-Optimal design
# with 10 more points
Xcand.parts <- partition(Xcand, out)
```

plot.tgp *Plotting for Treed Gaussian Process Models*

Description

A generic function for plotting of "tgp"-class objects. 1-d posterior mean and error plots, 2-d posterior mean and error image and perspective plots, and 3+-dimensional mean and error image and perspective plots are supported via projection and slicing.

Usage

```
## S3 method for class 'tgp'
plot(x, pparts = TRUE, proj = NULL, slice = NULL,
         map = NULL, as = NULL, center = "mean", layout = "both",
         main = NULL, xlab = NULL, ylab = NULL, zlab = NULL, pc = "pc",
         gridlen = c(40,40), span = 0.1, pXX = TRUE,
         legendloc = "topright", maineff = TRUE,  mrlayout="both",
         rankmax = 20, ...)
```

Arguments

x	"tgp"-class object that is the output of one of the b* functions: blm, btlm bgp, bgpllm, btgp, or btgpllm
pparts	If TRUE, partition-regions are plotted (default), otherwise they are not
proj	1-or-2-Vector describing the dimensions to be shown in a projection. The argument is ignored for 1-d data, i.e., if x$d == 1. For 2-d data, no projection needs be specified— the default argument (proj = NULL) will result in a 2-d perspective or image plot. 1-d projections of 2-d or higher data are are supported, e.g., proj = c(2) would show the second variable projection. For 3-d data or higher, proj=NULL defaults to proj = c(1,2) which plots a 2-d projection for the first two variables. Slices have priority over the projections— see next argument (slice)— when non-null arguments are provided for both.
slice	list object with x and z fields, which are vectors of equal length describing the slice to be plotted, i.e., which z-values of the x$d - 2 inputs x$X and x$XX should be fixed to in order to obtain a 2-d visualization. For example, for 4-d data, slice = list(x=(2,4), z=c(0.2, 1.5) will result in a 2-d plot of the first and third dimensions which have the second and fourth slice fixed at 0.5 and 1.5. The default is NULL, yielding to the proj argument. Argument is ignored for 1-d data, i.e., if x$d == 1
map	Optional 2-d map (longitude and latitude) from **maps** to be shown on top of image plots
center	Default center = "mean" causes the posterior predictive mean to be plotted as the centering statistic. Otherwise the median can be used with center = "med", or the kriging mean with center = "km"

`as`	Optional string indicator for plotting of adaptive sampling statistics: specifying `as = "alm"` for ALM, `as = "s2"` for predictive variance, `as = "ks2"` for expected kriging variance, `as = "alc"` for ALC, and `as = "improv"` for expected improvement (about the minimum, see the `rankmax` argument below). The default `as = NULL` plots error-bars (1d-plots) or error magnitudes (2d-plots), which is essentially the same as `as = "alm"`
`layout`	Specify whether to plot the mean predictive surface (`layout = "surf"`), the error or adaptive sampling statistics (`layout = "as"`), or default (`layout = "both"`) which shows both. If `layout = "sens"`, plot the results of a sensitivity analysis (see `sens`) in a format determined by the argument `maineff` below.
`main`	Optional `character` string to add to the main title of the plot
`xlab`	Optional `character` string to add to the x label of the plots
`ylab`	Optional `character` string to add to the y label of the plots
`zlab`	Optional `character` string to add to the z label of the plots; ignored unless `pc = "p"`
`pc`	Selects perspective-posterior mean and image-error plots (`pc = "pc"`, the default) or a double–image plot (`pc = "c"`)

(only valid for 2-d plots)

`gridlen`	Number of regular grid points for 2-d slices and projections in x and y. The default of `gridlen = c(40,40)` causes a `40 * 40` grid of `X`, `Y`, and `Z` values to be computed. Ignored for 1-d plots and projections
`span`	Span for `loess` kernel. The **tgp** package default (`span = 0.1`) is set lower than the `loess` default. Smaller spans can lead to warnings from `loess` when the data or predictive locations are sparse and ugly plots may result. In this case, try increasing the span
`pXX`	scalar logical indicating if `XX` locations should be plotted
`legendloc`	Location of the `legend` included in the plots of sensitivity analyses produced with `layout = "sens"`, or 1-d plots of multi-resolution models (with `corr = "mrexpsep"`) and option `mrlayout = "both"`; otherwise the argument is ignored
`maineff`	Format for the plots of sensitivity analyses produced with `layout = "sens"`; otherwise the argument is ignored. If `maineff=TRUE` main effect plots are produced alongside boxplots for posterior samples of the sensitivity indices, and if `FALSE` only the boxplots are produced. Alternatively, `maineff` can be a matrix containing input dimensions in the configuration that the corresponding main effects are to be plotted; that is, `mfrow=dim(maineff)`. In this case, a 90 percent interval is plotted with each main effect and the sensitivity index boxplots are not plotted.
`mrlayout`	The plot layout for double resolution tgp objects with `params$corr == "mrexpsep"`. For the default `mrlayout="both"`, the coarse and fine fidelity are plotted together, either on the same plot for 1D inputs or through side-by-side image plots of the predicted `center` with axis determined by `proj` for inputs of greater dimension. Note that many of the standard arguments – such as `slice`, `pc`, and `map` – are either non-applicable or unsupported for `mrlayout="both"`. If `mrlayout="coarse"` or `mrlayout="fine"`, prediction for the respective fidelity is plotted as usual and all of the standard options apply.

`rankmax`	When `as = "improv"` is provided, the posterior expected improvements are plotted according the the first column of the `improv` field of the `"tgp"`-class object. Text is added to the plot near the `XX` positions of the first `1:rankmax` predictive locations with the highest ranks in the second column of the `improv` field.
`...`	Extra arguments to 1-d (`plot`) and 2-d plotting functions `persp` and `image`

Value

The only output of this function is beautiful plots

Note

This plotting function is provided with the intention that it will be used as an aid in the visualization of `"tgp"`-class objects. Users are encouraged to use the source code for this function in order to develop custom plotting functions.

1-d projections for 3-d or higher data are also available by specifying a 1-d projection argument (e.g. `proj=c(1)` for projecting onto the first input variable).

For examples, see `vignette("tgp")` and the help files of those functions in "See Also", below

Author(s)

Robert B. Gramacy, <`rbg@vt.edu`>, and Matt Taddy, <`mataddy@amazon.com`>

References

`https://bobby.gramacy.com/r_packages/tgp/`

See Also

`plot`, `bgpllm`, `btlm`, `blm`, `bgp`, `btgpllm`, `predict.tgp`, `tgp.trees`, `mapT`, `loess`, `sens`

`predict.tgp` *Predict method for Treed Gaussian process models*

Description

This generic prediction method was designed to obtain samples from the posterior predictive distribution after the `b*` functions have finished. Samples, or kriging mean and variance estimates, can be obtained from the MAP model encoded in the `"tgp"`-class object, or this parameterization can be used as a jumping-off point in obtaining further samples from the joint posterior and posterior predictive distributions

Usage

```
## S3 method for class 'tgp'
predict(object, XX = NULL, BTE = c(0, 1, 1), R = 1,
            MAP = TRUE, pred.n = TRUE, krige = TRUE, zcov = FALSE,
            Ds2x = FALSE, improv = FALSE, sens.p = NULL, trace = FALSE,
            verb = 0, ...)
```

Arguments

object	"tgp"-class object that is the output of one of the b* functions: blm, btlm bgp, bgpllm, btgp, or btgpllm
XX	Optional data.frame, matrix, or vector of predictive input locations with ncol(XX) == ncol(object$X)
BTE	3-vector of Monte-carlo parameters (B)urn in, (T)otal, and (E)very. Predictive samples are saved every E MCMC rounds starting at round B, stopping at T. The default BTE=c(0,1,1) is specified to give the kriging means and variances as outputs, plus one sample from the posterior predictive distribution
R	Number of repeats or restarts of BTE MCMC rounds, default R=1 is no restarts
MAP	When TRUE (default) predictive data (i.e., kriging mean and variance estimates, and samples from the posterior predictive distribution) are obtained for the *fixed* MAP model encoded in object. Otherwise, when MAP=FALSE sampling from the joint posterior of the model parameters (i.e., tree and GPs) and the posterior predictive distribution are obtained starting from the MAP model and proceeding just as the b* functions
pred.n	TRUE (default) value results in prediction at the inputs X; FALSE skips prediction at X resulting in a faster implementation
krige	TRUE (default) value results in collection of kriging means and variances at predictive (and/or data) locations; FALSE skips the gathering of kriging statistics giving a savings in storage
zcov	If TRUE then the predictive covariance matrix is calculated– can be computationally (and memory) intensive if X or XX is large. Otherwise only the variances (diagonal of covariance matrices) are calculated (default). See outputs Zp.s2, ZZ.s2, etc., below
Ds2x	TRUE results in ALC (Active Learning–Cohn) computation of expected reduction in uncertainty calculations at the X locations, which can be used for adaptive sampling; FALSE (default) skips this computation, resulting in a faster implementation
improv	TRUE results in samples from the improvement at locations XX with respect to the observed data minimum. These samples are used to calculate the expected improvement over XX, as well as to rank all of the points in XX in the order that they should be sampled to minimize the expected multivariate improvement (refer to Schonlau et al, 1998). Alternatively, improv can be set to any positive integer 'g', in which case the ranking is performed with respect to the expectation for improvement raised to the power 'g'. Increasing 'g' leads to rankings that are more oriented towards a global optimization. The option FALSE (default) skips these computations, resulting in a faster implementation. Optionally, a two-vector can be supplied where improv[2] is interpreted as the (maximum) number of points to rank by improvement. See the note in btgp documentation. If not specified, then the larger of 10% of nn = nrow(XX) and min(10, nn) is taken by default
sens.p	Either NULL or a vector of parameters for sensitivity analysis, built by the function sens. Refer there for details

`trace`	`TRUE` results in a saving of samples from the posterior distribution for most of the parameters in the model. The default is `FALSE` for speed/storage reasons. See note below
`verb`	Level of verbosity of R-console print statements: from 0 (default: none); 1 which shows the "progress meter"; 2 includes an echo of initialization parameters; up to 3 and 4 (max) with more info about successful tree operations
`...`	Ellipses are not used in the current version of `predict.tgp`. They are are only included in order to maintain S3 generic/method consistency

Details

While this function was designed with prediction in mind, it is actually far more general. It allows a continuation of MCMC sampling where the `b*` function left off (when `MAP=FALSE`) with a possibly new set of predictive locations `XX`. The intended use of this function is to obtain quick kriging-style predictions for a previously-fit MAP estimate (contained in a `"tgp"`-class object) on a new set of predictive locations `XX`. However, it can also be used simply to extend the search for an MAP model when `MAP=FALSE`, `pred.n=FALSE`, and `XX=NULL`

Value

The output is the same, or a subset of, the output produced by the `b*` functions, for example see `btgp`

Note

It is important to note that this function is not a replacement for supplying `XX` to the `b*` functions, which is the only way to get fully Bayesian samples from the posterior prediction at new inputs. It is only intended as a post-analysis (diagnostic) tool.

Inputs `XX` containing `NaN`, `NA`, or `Inf` are discarded with non-fatal warnings. Upon execution, MCMC reports are made every 1,000 rounds to indicate progress.

If `XX`s are provided which fall outside the range of `X` inputs provided to the original `b*` function, then those will not be extrapolated properly, due to the way that bounding rectangles are defined in the original run. For a workaround, supply `out$Xsplit <- rbind(X, XX)` before running `predict` on `out`.

See note for `btgp` or another `b*` function regarding the handling and appropriate specification of `traces`.

The `"tgp"` class output produced by `predict.tgp` can also be used as input to `predict.tgp`, as well as others (e.g., `plot.tgp`.

Author(s)

Robert B. Gramacy, `<rbg@vt.edu>`, and Matt Taddy, `<mataddy@amazon.com>`

References

`https://bobby.gramacy.com/r_packages/tgp/`

See Also

predict, blm, btlm, bgp, btgp, bgpllm, btgpllm, plot.tgp

Examples

```
## revisit the Motorcycle data
require(MASS)

## fit a btgpllm without predictive sampling (for speed)
out <- btgpllm(X=mcycle[,1], Z=mcycle[,2], bprior="b0",
        pred.n=FALSE)
## nothing to plot here because there is no predictive data

## save the "tgp" class output object for use later and
save(out, file="out.Rsave")

## then remove it (for illustrative purposes)
out <- NULL

## (now imagine emailing the out.Rsave file to a friend who
## then performs the following in order to use your fitted
## tgp model on his/her own predictive locations)

## load in the "tgp" class object we just saved
load("out.Rsave")

## new predictive locations
XX <- seq(2.4, 56.7, length=200)

## now obtain kriging estimates from the MAP model
out.kp <- predict(out, XX=XX, pred.n=FALSE)
plot(out.kp, center="km", as="ks2")

## actually obtain predictive samples from the MAP
out.p <- predict(out, XX=XX, pred.n=FALSE, BTE=c(0,1000,1))
plot(out.p)

## use the MAP as a jumping-off point for more sampling
out2 <- predict(out, XX, pred.n=FALSE, BTE=c(0,2000,2),
                MAP=FALSE, verb=1)
plot(out2)

## (generally you would not want to remove the file)
unlink("out.Rsave")
```

---

sens *Monte Carlo Bayesian Sensitivity Analysis*

Description

Fully Bayesian Monte Carlo sensitivity analysis scheme, based upon any of the regression models contained in the **tgp** package. Random Latin hypercube samples are drawn at each MCMC iteration in order to estimate main effects as well as 1st order and total sensitivity indices.

Usage

```
sens(X, Z, nn.lhs, model = btgp, ngrid = 100, span = 0.3,
     BTE = c(3000,8000,10), rect = NULL, shape = NULL, mode = NULL,
     ...)
```

Arguments

`X`	`data.frame`, `matrix`, or vector of inputs `X`
`Z`	Vector of output responses `Z` of length equal to the leading dimension (rows) of `X`, i.e., `length(Z) == nrow(X)`
`nn.lhs`	Size of each Latin hypercube drawn for use in the Monte Carlo integration scheme. Total number of locations for prediction is `nn.lhs*(ncol(X)+2)`
`model`	Either the regression model used for prediction, or `NULL`. If `model=NULL`, then the function just returns the `sens.p` vector of parameters to be passed with a regression model call. This can be used to perform sensitivity analysis through the `predict.tgp` framework
`ngrid`	The number of grid points in each input dimension upon which main effects will be estimated.
`span`	Smoothing parameter for main effects integration: the fraction of `nn.lhs` points that will be included in a moving average window that is used to estimate main effects at the `ngrid` locations in each input dimension.
`BTE`	3-vector of Monte-Carlo parameters (B)urn in, (T)otal, and (E)very. Predictive samples are saved every E MCMC rounds starting at round B, stopping at T
`rect`	Rectangle describing the domain of the uncertainty distribution with respect to which the sensitivity is to be determined. This defines the domain from which the LH sample is to be taken. The rectangle should be a `matrix` or `data.frame` with `ncol(rect) = 2`, and number of rows equal to the dimension of the domain. For 1-d data, a vector of length 2 is allowed. Defaults to the input data range (`X`).
`shape`	Optional vector of shape parameters for the Beta distribution. Vector of length equal to the dimension of the domain, with elements > 1. If specified, the uncertainty distribution (i.e. the LH sample) is proportional to a joint pdf formed by independent Beta distributions in each dimension of the domain, scaled and shifted to have support defined by `rect`. Only concave Beta distributions with `shape` > 1 are supported. If unspecified, the uncertainty distribution is uniform over `rect`. The specification `shape[i]=0` instructs `sens` to treat the i'th dimension as a binary variable. In this case, `mode[i]` is the probability parameter for a bernoulli uncertainty distribution, and we must also have `rect[i,]=c(0,1)`.
`mode`	Optional vector of mode values for the Beta uncertainty distribution. Vector of length equal to the dimension of the domain, with elements within the support defined by `rect`. If `shape` is specified, but this is not, then the scaled Beta distributions will be symmetric.

`...`	Extra arguments to the **tgp** `model`.

Details

Saltelli (2002) describes a Latin Hypercube sampling based method for estimation of the 'Sobal' sensitivity indices:

1st Order for input i,

$$S(i) = \text{Var}(E[f|x_i])/\text{Var}(f),$$

where x_i is the i-th input.

Total Effect for input i,

$$T(i) = E[\text{Var}(f|x_{-i})]/\text{Var}(f),$$

where x_{-i} is all inputs except for the i-th.

All moments are with respect to the appropriate marginals of the uncertainty distribution U – that is, the probability distribution on the inputs with respect to which sensitivity is being investigated. Under this approach, the integrals involved are approximated through averages over properly chosen samples based on two LH samples proportional to U. If `nn.lhs` is the sample size for the Monte Carlo estimate, this scheme requires `nn.lhs*(ncol(X)+2)` function evaluations.

The `sens` function implements the method for unknown functions f, through prediction via one of the **tgp** regression models conditional on an observed set of `X` locations. At each MCMC iteration of the **tgp** model fitting, the `nn.lhs*(ncol(X)+2)` locations are drawn randomly from the LHS scheme and realizations of the sensitivity indices are calculated. Thus we obtain a posterior sample of the indices, incorporating variability from both the Monte Carlo estimation and uncertainty about the function output. Since a subset of the predictive locations are actually an LHS proportional to the uncertainty distribution, we can also estimate the main effects through simple non-parametric regression (a moving average).

Please see `vignette("tgp2")` for a detailed illustration

Value

The output is a `"tgp"`-class object. The details for `btgp` contain a complete description of this output. The list entry that is relevance to sensitivity analysis is `sens`, which itself has entries:

`par`	This contains a `list` of the input parameters used in the sensitivity analysis, as outlined above.
`Xgrid`	A `matrix` containing a grid in each input dimension (by column) over which the main effects are estimated.
`ZZ.mean`	A `matrix`, where each column contains the mean main effects over the corresponding column of `sens.Xgrid`.
`ZZ.q1`	A `matrix`, where each column contains the 5th percentile main effects over the corresponding column of `sens.Xgrid`.
`ZZ.q2`	A `matrix`, where each column contains the 5th percentile main effects over the corresponding column of `sens.Xgrid`.
`S`	A `matrix`, where each column contains the posterior sample of 1st order sensitivity indices for the corresponding input dimension.
`T`	A `matrix`, where each column contains the posterior sample of total sensitivity indices for the corresponding input dimension.

Note

The quality of sensitivity analysis is dependent on the size of the LH samples used for integral approximation; as with any Monte Carlo integration scheme, the sample size (`nn.lhs`) must increase with the dimensionality of the problem. The total sensitivity indices T are forced non-negative, and if negative values occur it is necessary to increase `nn.lhs`. The `plot.tgp` function replaces negative values with zero for illustration.

Author(s)

Robert B. Gramacy, <`rbg@vt.edu`>, and Matt Taddy, <`mataddy@amazon.com`>

References

Gramacy, R. B. (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC. (See Chapter 8.) https://bobby.gramacy.com/surrogates/

R.D. Morris, A. Kottas, M. Taddy, R. Furfaro, and B. Ganapol. (2009) *A statistical framework for the sensitivity analysis of radiative transfer models.* IEEE Transactions on Geoscience and Remote Sensing, to appear.

Saltelli, A. (2002) *Making best use of model evaluations to compute sensitivity indices.* Computer Physics Communications, 145, 280-297.

See Also

`btgp`, `plot.tgp`, `predict.tgp`, `lhs`

Examples

```
# Take a look at the air quality in New York: Sensitivity of
# ozone levels with respect to solar radiation, wind, and
# temperature. See help(airquality) for details.
X <- airquality[,2:4]
Z <- airquality$Ozone

# Uncertainty distribution is the default: uniform over range(X)
# There is missing data, which is removed automatically by tgp
# range(X).

s <- suppressWarnings(sens(X=X, Z=Z, nn.lhs=300, model=btgp,
          ngrid=100, span=0.3, BTE=c(5000,10000,10)))

# plot the results
plot(s, layout="sens", ylab="Ozone", main="main effects")

# plot only the sensitivity indices
plot(s, layout="sens", ylab="Ozone", maineff=FALSE)

# plot only the main effects, side by side
plot(s, layout="sens", ylab="Ozone", main="", maineff=t(1:3))
```

```
# build a 'sens.p' parameter vector for a data-dependent
# informative uncertainty distribution.  For each variable,
# the input distribution will be a scaled Beta with shape=2,
# and mode equal to the data mean
rect <- t(apply(X, 2, range, na.rm=TRUE))
mode <- apply(X , 2, mean, na.rm=TRUE)
shape <- rep(2,3)

# plot a sample from the marginal uncertainty distribution.
Udraw <- lhs(300, rect=rect, mode=mode, shape=shape)
par(mfrow=c(1,3))
for(i in 1:3) hist(Udraw[,i], breaks=15,xlab=names(X)[i])

# build sens.p with the 'sens' function.
sens.p <- suppressWarnings(sens(X=X, Z=Z, nn.lhs=300, model=NULL,
               ngrid=100, rect=rect, shape=shape, mode=mode))

# Use predict.tgp to quickly analyze with respect to this new
# uncertainty distribution without re-running the MCMC, then
# plot the results.

s.new <- predict(s, BTE=c(1,1000,1), sens.p=sens.p, verb=1)
plot(s.new, layout="sens", ylab="Ozone", main="main effects")
```

`tgp.default.params` *Default Treed Gaussian Process Model Parameters*

Description

Construct a default list of parameters to the `b*` functions– the interfaces to treed Gaussian process modeling

Usage

```
tgp.default.params(d, meanfn = c("linear", "constant"),
                 corr = c("expsep", "exp", "mrexpsep", "matern", "sim", "twovar"),
                     splitmin = 1, basemax = d, ...)
```

Arguments

`d` number of input dimensions `ncol(X)`

`meanfn` A choice of mean function for the process. When `meanfn = "linear"` (default), then we have the process

$$Z = (\mathbf{1} \;\; \mathbf{X})\beta + W(\mathbf{X})$$

where $W(\mathbf{X})$ represents the Gaussian process part of the model (if present). Otherwise, when `meanfn = "constant"`, then

$$Z = \beta_0 + W(\mathbf{X})$$

`corr`	Gaussian process correlation model. Choose between the isotropic power exponential family (`"exp"`) or the separable power exponential family (`"expsep"`, default); the current version also supports the isotropic Matern (`"matern"`) and single-index model (`"sim"`) and `"twovar"` as "beta" functionality. The option `"mrexpsep"` uses a multi-resolution GP model, a depricated feature in the package (docs removed)
`splitmin`	Indicates which column of the inputs `X` should be the first to allow splits via treed partitioning. This is useful for excluding certain input directions from the partitioning mechanism
`basemax`	Indicates which column of the inputs `X` should be the last be fit under the base model (e.g., LM or GP). This is useful for allowing some input directions (e.g., binary indicators) to only influence the tree partitioning mechanism, and not the base model(s) at the leaves of the tree
`...`	These ellipses arguments are interpreted as augmentations to the prior specification. You may use these to specify a custom setting of any of default parameters in the output list detailed below

Value

The output is the following list of `params`...

`col`	dimension of regression coefficients β: 1 for input `meanfn = "constant"`, or `ncol(X)+1` for `meanfn = "linear"`
`meanfn`	copied from the inputs
`corr`	copied from the inputs
`bprior`	Linear (beta) prior, default is `"bflat"` which gives an "improper" prior which can perform badly when the signal-to-noise ratio is low. In these cases the "proper" hierarchical specification `"b0"`, `"bmzt"`, or `"bmznot"` prior may perform better
`beta`	`rep(0,col)` starting values for beta linear parameters
`tree`	`c(0.5,2,max(c(10,col+1)),1,d)` indicating the tree prior process parameters α, β, *minpart*, *splitmin* and *basemax*: $$p_{\mathrm{split}}(\eta, \mathcal{T}) = \alpha * (1+\eta)^\beta$$ with zero probability given to trees with partitions containing less than `nmin` data points; *splitmin* indicates the first column of `X` which where treed partitioning is allowed; *basemax* gives the last column where the base model is used
`s2.p`	`c(5,10)` σ^2 inverse-gamma prior parameters `c(a0, g0)` where `g0` is rate parameter
`tau2.p`	`c(5,10)` τ^2 inverse-gamma prior parameters `c(a0, g0)` where `g0` is rate parameter
`d.p`	c(1.0,20.0,10.0,10.0) Mixture of gamma prior parameter (initial values) for the range parameter(s) `c(a1,g1,a2,g2)` where `g1` and `g2` are rate parameters. If `corr="mrexpsep"`, then this is a vector of length 8: The first four parameters remain the same and correspond to the "coarse" process, and the second set of four values, which default to `c(1,10,1,10)`, are the equivalent prior parameters for the range parameter(s) in the residual "fine" process.

`nug.p`	`c(1,1,1,1)` Mixture of gamma prior parameter (initial values) for the nugget parameter `c(a1,g1,a2,g2)` where `g1` and `g2` are rate parameters; default reduces to simple exponential prior; specifying `nug.p = 0` fixes the nugget parameter to the "starting" value in `gd[1]`, i.e., it is excluded from the MCMC
`gamma`	`c(10,0.2,10)` LLM parameters c(g, t1, t2), with growth parameter `g > 0` minimum parameter `t1 >= 0` and maximum parameter `t1 >= 0`, where `t1 + t2 <= 1` specifies $p(b\|d) = t_1 + \exp\left\{\frac{-g(t_2 - t_1)}{d - 0.5}\right\}$
`d.lam`	`"fixed"` Hierarchical exponential distribution parameters to `a1`, `g1`, `a2`, and `g2` of the prior distribution for the range parameter `d.p`; `"fixed"` indicates that the hierarchical prior is "turned off"
`nug.lam`	`"fixed"` Hierarchical exponential distribution parameters to `a1`, `g1`, `a2`, and `g2` of the prior distribution for the nug parameter `nug.p`; `"fixed"` indicates that the hierarchical prior is "turned off"
`s2.lam`	`c(0.2,10)` Hierarchical exponential distribution prior for `a0` and `g0` of the prior distribution for the s2 parameter `s2.p`; `"fixed"` indicates that the hierarchical prior is "turned off"
`tau2.lam`	`c(0.2,0.1)` Hierarchical exponential distribution prior for `a0` and `g0` of the prior distribution for the s2 parameter `tau2.p`; `"fixed"` indicates that the hierarchical prior is "turned off"
`delta.p`	`c(1,1,1,1)` Parameters in the mixture of gammas prior on the delta scaling parameter for `corr="mrexpsep"`: `c(a1,g1,a2,g2)` where `g1` and `g2` are rate parameters; default reduces to simple exponential prior. Delta scales the variance of the residual "fine" process with respect to the variance of the underlying "coarse" process.
`nugf.p`	`c(1,1,1,1)` Parameters in the mixture of gammas prior on the residual "fine" process nugget parameter for `corr="mrexpsep"`: `c(a1,g1,a2,g2)` where `g1` and `g2` are rate parameters; default reduces to simple exponential prior.
`dp.sim`	`basemax * basemax` RW-MVN proposal covariance matrix for GP-SIM models; only appears when `corr="sim"`, the default is `diag(rep(0.2, basemax))`

Note

Please refer to the examples for the functions in "See Also" below, `vignette("tgp")` and `vignette(tgp2)`

Author(s)

Robert B. Gramacy, `<rbg@vt.edu>`, and Matt Taddy, `<mataddy@amazon.com>`

References

Gramacy, R. B. (2007). **tgp***: An* R *Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models.* Journal of Statistical Software, **19**(9). `https://www.jstatsoft.org/v19/i09` doi:10.18637/jss.v019.i09

Robert B. Gramacy, Matthew Taddy (2010). *Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with* **tgp** *Version 2, an* R *Package for Treed Gaussian Process*

Models. Journal of Statistical Software, **33**(6), 1–48. https://www.jstatsoft.org/v33/i06/ doi:10.18637/jss.v033.i06

Gramacy, R. B., Lee, H. K. H. (2008). *Bayesian treed Gaussian process models with an application to computer modeling*. Journal of the American Statistical Association, 103(483), pp. 1119-1130. Also available as ArXiv article 0710.4536 https://arxiv.org/abs/0710.4536

Robert B. Gramacy, Heng Lian (2011). *Gaussian process single-index models as emulators for computer experiments*. Available as ArXiv article 1009.4241 https://arxiv.org/abs/1009.4241

https://bobby.gramacy.com/r_packages/tgp/

See Also

`blm`, `btlm`, `bgp`, `btgp`, `bgpllm`, `btgpllm`

`tgp.design` *Sequential Treed D-Optimal Design for Treed Gaussian Process Models*

Description

Based on the maximum a' posteriori (MAP) treed partition extracted from a `"tgp"`-class object, calculate independent sequential treed D-Optimal designs in each of the regions.

Usage

```
tgp.design(howmany, Xcand, out, iter = 5000, verb = 0)
```

Arguments

`howmany`	Number of new points in the design. Must be less than the number of candidates contained in `Xcand`, i.e., `howmany <= nrow(Xcand)`
`Xcand`	`data.frame`, `matrix` or vector of candidates from which new design points are subsampled. Must have `nrow(Xcand) == nrow(out$X)`
`out`	`"tgp"`-class object output from one of the model functions which has tree support, e.g., `btgpllm`, `btgp`, `btlm`
`iter`	number of iterations of stochastic accent algorithm, default `5000`
`verb`	positive integer indicating after how many rounds of stochastic approximation in `dopt.gp` to print each progress statement; default `verb=0` results in no printing

Details

This function partitions `Xcand` and `out$X` based on the MAP tree (obtained on `"tgp"`-class `out` with `partition`) and calls `dopt.gp` in order to obtain a D-optimal design under independent stationary Gaussian processes models defined in each region. The aim is to obtain a design where new points from `Xcand` are spaced out relative to themselves, and relative to the existing locations (`out$X`) in the region. The number of new points from each region of the partition is proportional to the number of candidates `Xcand` in the region.

Value

Output is a list of `data.frames` containing `XX` design points for each region of the MAP tree in `out`

Note

Input `Xcand` containing `NaN, NA, Inf` are discarded with non-fatal warnings

D-Optimal computation in each region is preceded by a print statement indicated the number of new locations to be chosen and the number of candidates in the region. Other than that, there are no other indicators of progress. You will have to be patient. Creating treed sequential D-optimal designs is no speedy task. At least it faster than the non-treed version (see `dopt.gp`).

The example below is also part of `vignette("tgp")`. Please see `vignette("tgp2")` for a similar example based on optimization using the `optim.step.tgp`

Author(s)

Robert B. Gramacy, `<rbg@vt.edu>`, and Matt Taddy, `<mataddy@amazon.com>`

References

Gramacy, R. B. (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC. (See Chapter 9.) `https://bobby.gramacy.com/surrogates/`

Gramacy, R. B. (2007). **tgp***: An* R *Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models.* Journal of Statistical Software, **19**(9). `https://www.jstatsoft.org/v19/i09` doi:10.18637/jss.v019.i09

Robert B. Gramacy, Matthew Taddy (2010). *Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with* **tgp** *Version 2, an* R *Package for Treed Gaussian Process Models.* Journal of Statistical Software, **33**(6), 1–48. `https://www.jstatsoft.org/v33/i06/` doi:10.18637/jss.v033.i06

Gramacy, R. B., Lee, H. K. H. (2006). *Adaptive design and analysis of supercomputer experiments.* Technometrics, 51(2), pp. 130-145. Also avaliable on ArXiv article 0805.4359 `https://arxiv.org/abs/0805.4359`

Gramacy, R. B., Lee, H. K. H., & Macready, W. (2004). *Parameter space exploration with Gaussian process trees.* ICML (pp. 353–360). Omnipress & ACM Digital Library.

`https://bobby.gramacy.com/r_packages/tgp/`

See Also

`bgpllm`, `btlm`, `blm`, `bgp`, `btgpllm`, `plot.tgp`, `dopt.gp`, `lhs`, `partition`, `optim.step.tgp`

Examples

```
#
# 2-d Exponential data
# (This example is based on random data.
# It might be fun to run it a few times)
#
```

```
# get the data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
Xcand <- exp2d.data$XX

# fit treed GP LLM model to data w/o prediction
# basically just to get MAP tree (and plot it)
out <- btgpllm(X=X, Z=Z, pred.n=FALSE, corr="exp")
tgp.trees(out)

# find a treed sequential D-Optimal design
# with 10 more points.  It is interesting to
# contrast this design with one obtained via
# the dopt.gp function
XX <- tgp.design(10, Xcand, out)

# now fit the model again in order to assess
# the predictive surface at those new design points
dout <- btgpllm(X=X, Z=Z, XX=XX, corr="exp")
plot(dout)
```

`tgp.trees` *Plot the MAP Tree for each height encountered by the Markov Chain*

Description

Plot the maximum a' posteriori (MAP) tree as a function of tree height, and show the log posterior probabilities for comparison.

Usage

```
tgp.trees(out, heights = NULL, main = NULL, ...)
```

Arguments

`out`	`"tgp"`-class object which is the output of one the model functions with tree support (e.g. `btgpllm`)
`heights`	Index vector of length less than `length(out$trees)` describing trees to plot by their height. Default (`NULL`) is to plot all trees, one for each height encountered when sampling from the Markov chain of the tree posterior. This is equivalent to `heights = out$posts$height`. Specifying `heights = "map"` causes (only) the maximum a' posteriori (MAP) height tree to be plotted
`main`	Optional character string to add to the main title of the plot
`...`	Extra arguments to the `draw.tree` function from **maptree**

Details

The maximum a' posteriori (MAP) tree encountered at each height (in the MCMC chain) is plotted, and the log posterior probabilities are shown for comparison. The text at the branches in the tree show the splitting variable and value. The text at the leaves show the number of input data points (`X` and `Z`) that fall into the region(s) along with an estimate of the variability therein.

Value

The only output of this function is beautiful tree diagrams.

Note

Plotting trees that the **maptree** library is installed, which itself requires that the **combinat** library also be installed.

See `vignette("tgp")` and the examples sections of the functions under "See Also", below

Author(s)

Robert B. Gramacy, `<rbg@vt.edu>`, and Matt Taddy, `<mataddy@amazon.com>`

References

`https://bobby.gramacy.com/r_packages/tgp/`

See Also

`bgpllm`, `btlm`, `blm`, `bgp`, `btgpllm`, `plot.tgp`, `mapT`, `vignette("tgp")`

Index