Package 'growfunctions'

August 9, 2014

Type Package

Title Bayesian non-parametric dependent models for time-indexed functional data

Version 0.1

Date 2014-08-01

Author Terrance Savitsky

Maintainer Terrance Savitsky <tds151@gmail.com>

Description Estimates a collection of time-indexed functions under either of Gaussian process (GP) or intrinsic Gaussian Markov random field (iGMRF) prior formulations where a Dirichlet process mixture allows sub-groupings of the functions to share the same covariance or precision parameters. The GP and iGMRF formulations both support any number of additive covariance or precision terms,respectively, expressing either or both of multiple trend and seasonality.

License GPL (>= 3)

Depends R (>= 3.1.1), Rcpp (>= 0.11.2)

LinkingTo Rcpp (>= 0.11.2), RcppArmadillo (>= 0.4.000)

Imports Matrix(>= 1.1), spam(>= 0.41-0), mvtnorm(>= 1.0-0), ggplot2(>= 0.9.3.1), reshape2(>= 1.2.2), scales(>= 0.2.3)

Suggests testthat(>= 0.8.1)

Collate 'MSPE.R' 'cps.R' 'gen_informative_sample.R' 'gpdpgrow.R' 'gmrfdpgrow.R' 'gp_car_fit_compare_facet.R' 'gp_cluster_plot.R' 'gp_informative_compare_plot.R' 'help.R' 'plot_cluster.R' 'predict_plot.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-08-09 00:03:47

R topics documented:

`growfunctions-package` *Bayesian non-parametric models for estimating a set of denoised, latent functions from an observed collection of domain-indexed time-series.*

Description

Package:	growfunctions
Type:	Package
Version:	0.1
Date:	2014-08-01
License:	GPL (>= 3)
LazyLoad:	yes

Details

Parameterizes a model for a collection of noisy time-series indexed by domain or observation unit as an additive function process plus noise process. The latent functions are modeled under both Gaussian process (GP) and intrinsic Gaussian Markov random (iGMRF) field priors. Dependence is estimated among the set of functions by selecting a prior, G, on the covariance or precision parameters of the GP and iGMRF, respectively, where G receives a Dirichlet process (DP) prior. The resulting marginal prior on the functions is a nonparametric scale mixture over the covariance or precision parameters, with G the unknown mixing measure. Draws from a DP are almost surely discrete, allowing for ties (interpreted as clusters) among the covariance or precision parameters indexed by domain or observation unit. Functions are included that permit additional inference on the clustering properties over the domains. The mixture models allow specification of multiple additive latent functions for each of the GP and iGMRF DP mixture formulations. Each function (under a GP prior) may be specified with a covariance function selected from a set of available options that allow for combinations of trend and seasonal components. The same is true for precision matrix constructions under the iGMRF DP mixture.

ESTIMATION FUNCTIONS

`gpdpgrow` performs Bayesian nonparametric estimation of a set of dependent, denoised latent functions under a DP mixture of GP's in an unsupervised fashion based on user input of an `N x T` data matrix, `y`, where `N` denotes the number of domains or units and `T` denotes the number of time points. The DP prior estimates the dependent structure among the covariance parameters generating each `T x 1` function across the `N` domains. The user may specify multiple latent functions in an additive regression formulation, where each covariance kernel may be selected from the squared exponential (`"se"`), the rational quadratic (`"rq"`), or a quasi-periodic (product of a period and squared exponential to allow the periodic kernel to evolve) (`"sn"`).

`gmrfdpgrow` also inputs an `N x T` data matrix, `y`, but replaces the GP prior used in `gpdpgrow()` with an iGMRF prior. The DP mixture is over the precision parameter, `kappa`, that multiplies a fixed matrix, `Q`, which specifies the length-scale of dependence. The user may specify multiple functions in an additive formulation, each with its own precision matrix. The precision matrix is specified via two options. Input `q_type` indicates whether the precision matrix is a *trend* (`"tr"`) precision matrix or a *seasonal* (`"sn"`) precision matrix. Integer innput `q_order` controls the length scale of the estimated functions and specifies the difference order of the precision matrix in the case of `q_type = "tr"`, or the periodicity, in the case `q_type = "sn"`. Typically, `q_order` is set to `1` or `2` when `q_type = "tr"`, though other values are possible.

`MSPE` Inputs a `gpdpgrow()` or `gmrfdpgrow()` object estimated where some data values deliberately set to missing (`NA`) and produces an out-of-sample mean square prediction error (MSPE), and a normalized MSPE (normalized by the variance of the missing test set). Both `gpdpgrow()` and `gmrfdpgrow()` returned objects also include a leave-one-out log-pseudo marginal likelihood `LPML` fit statistic estimated on the training set (so that no data are required to be left out). One would expect the `nMSPE` statistic to provide a greater penalty for model complexity since it is computed on data not used for estimation.

`predict_functions` Uses the model-estimated GP covariance parameters from `gpdpgrow()` or iGMRF precision parameters from `gmrfdpgrow()` to predict functions at *future* time points beyond the range of the data from their posterior predictive distributions. Both `gpdpgrow()` and `gmrfdpgrow()` will predict missing function values in the range of the data.

PLOT FUNCTIONS

cluster_plot inputs a returned object from either of gpdpgrow() or gmrfdpgrow() and produces two plots. The first plot creates panels for the clusters and aggregates line plots of posterior mean estimates for member denoised functions. An optional smoother may be drawn through the set of functions in each panel to differentiate the patterns expressed across clusters. A second plot renders the estimated posterior mean values (with an option for credible intervals) for a single or group of randomly-selected latent functions against the actual data values, y.

informative_plot inputs a list of returned objects, all from either of gpdpgrow() or gmrfdpgrow() (where the model type, GP or iGMRF, is communicated with input model) that compares credible intervals for covariance or precision parameters where the data are drawn from an informative sampling design (rather than as *iid*), so that the distribution for the population is not the same as that for the sample. One model conducts estimation in a fashion that ignores the informative design and the other incorporates sampling weights to account for the informativeness. Comparing the resulting estimated credible intervals provides a means to assess the sensitivity of estimated parameters to the sampling design. The set of objects are labeled with objects_labels with allowable inputs c("ignore","weight","iid"). One of the objects must have label "ignore", and the other must have label, weight. An additional object may also be input that is estimated from an iid sample drawn from the same population as the informative sample used for estimation under objects "ignore" and "weight". Both informative and iid samples are generated from the synthetic data function, gen_informative_sample().

fit_compare inputs a list of returned objects, each from either gpdpgrow() or gmrfdpgrow(), and plots the posterior mean estimate for a randomly-selected latent function as compared to the actual data in a set panels indexed by cluster and object. Allows comparison of fit performance of functions to data across varied model specifications.

predict_plot uses a returned object from predict_functions() to plot both estimated and predicted function values (with the option for credible intervals) where the prediction interval is outside the range of data. The plot function, cluster_plot, should be used where the predicted values are in the range of the data (and may, hence, be treated as missing values. The estimated functions are plotted in cluster_plot for the missing, as well as observed, data points).

DATA SETS (and functions to generate synthetic data sets)

cps Data derived from the Current Population Survey (CPS) administered to households in local areas within each of 51 states. These data capture a rectangular matrix of monthly-indexed all-other employment direct estimates computed from the set of household responses. The data capture 156 month observations (from 2000 - 2013) for each of the 51 states. Two objects are included: y_raw contains the 51 x 156 matrix of all-other employment counts. y contains the 51 x 156 matrix of all-other employment counts are standardization to (0,1), by state, to all them to be modeled together.

gen_informative_sample Generates an N x T population data matrix, y, and an associated n x T sample data matrix, y_obs, where the sample is drawn using a 1 or 2-stage informative process. The 1-stage sample uses unequal probability stratified sampling, while the 2-stage process samples the first stage in blocks, while the second stage samples with unequal probability from strata for selected blocks. Both block and strata memberships for population units are generated based on the variance of their time-series, y. The resulting sample is informative because the block and cluster memberships and selection probabilities are determined based on y.

Author(s)

Terrance Savitsky <tds151@gmail.com>

References

T. D. Savitsky (2014) Bayesian Non-parametric Functional Mixture Estimation for Time-indexed data. submitted to: Annals of Applied Statistics.

T. D. Savitsky, D. Toth and M. Sverchkov (2014) Bayesian Estimation Under Informative Sampling, Submitted to: Journal of the American Statistical Association.

T. D. Savitsky (2014) Bayesian Non-Parametric Mixture Estimation for Time-Indexed Functional Data for R. Submitted to: Journal of Statistical Software.

Examples

```
{
library(growfunctions)

## load the monthly employment count data for a collection of
## U.S. states from the Current
## Population Survey (cps)
data(cps)
## subselect the columns of N x T, y, associated with
## the years 2009 - 2013
## to examine the state level employment levels
## during the "great recession"
y_short             <- cps$y[,(cps$yr_label %in% c(2009:2013))]

## run DP mixture of GP's to estimate posterior distributions
## for model parameters
## uses default setting of a single "rational quadratic"
## covariance formula
## A short number of iterations is used for illustration
## Run for 500 iterations with half as burn-in to
## get a more useful result
res_gp              <- gpdpgrow(y = y_short,
                                n.iter = 7,
                                n.burn = 3,
                                n.thin = 1,
                                n.tune = 0)
## 2 plots of estimated functions: 1. faceted by cluster and fit;
## 2.  data for experimental units.
## for a group of randomly-selected functions
fit_plots_gp        <- cluster_plot( object = res_gp,
                                     units_name = "state",
                                     units_label = cps$st,
                                     single_unit = FALSE,
                                     credible = TRUE )

## Run the DP mixture of iGMRF's to estimate posterior
## distributions for model parameters
## Under default RW2(kappa) = order 2 trend
## precision term
## A short number of iterations is used for illustration
## Run for 2000 iterations with half as burn-in to
## get a more useful result
```

```
res_gmrf            <- gmrfdpgrow(y = y_short,
                                  n.iter = 30,
                                  n.burn = 15,
                                  n.thin = 1)

## 2 plots of estimated functions: 1. faceted by cluster and fit;
## 2.  data for experimental units.
## for a group of randomly-selected functions
fit_plots_gmrf      <- cluster_plot( object = res_gmrf,
                                  units_name = "state",
                                  units_label = cps$st,
                                  single_unit = FALSE,
                                  credible = TRUE )

## visual comparison of fit performance
## between gpdpgrow() and gmrfdpgrow()
## or any two objects returned from any
## combination of these estimation
## functions
objects                       <- vector("list",2)
objects[[1]]                  <- res_gmrf
objects[[2]]                  <- res_gp
label.object                  <- c("gmrf_tr2","gp_rq")
## the map data.frame object from fit_plots gp
## includes a field that
## identifies cluster assignments
## for each unit (or domain)
H                           <- fit_plots_gp$map$cluster
fit_plot_compare_facet      <- fit_compare( objects = objects,
                                 H = H,
                                 label.object = label.object,
                                 y.axis.label = "normalized y values",
                                 units_name = "state",
                                 units_label = cps$st)
}
```

`cluster_plot` *Plot estimated functions for experimental units faceted by cluster versus data to assess fit.*

Description

Uses as input the output object from the gpdpgrow() and gmrfdpgrow() functions.

Usage

```
cluster_plot(object, N_clusters = NULL, time_points = NULL,
  units_name = "unit", units_label = NULL, date_field = NULL,
  x.axis.label = NULL, y.axis.label = NULL, smoother = TRUE,
  sample_rate = 1, single_unit = FALSE, credible = FALSE,
  num_plot = NULL)
```

Arguments

object	A gpdpgrow or gmrfdpgrow object.
N_clusters	Denotes the number of largest sized (in terms of membership) clusters to plot. Defaults to all clusters.
time_points	Inputs a vector of common time points at which the collections of functions were observed (with the possibility of intermittent missingness). The length of time_points should be equal to the number of columns in the data matrix, y. Defaults to time_points = 1:ncol(y).
units_name	The plot label for observation units. Defaults to units_name = "function".
units_label	A vector of labels to apply to the observation units with length equal to the number of unique units. Defaults to sequential numeric values as input with data, y.
date_field	A vector of Date values for labeling the x-axis tick marks. Defaults to 1:T .
x.axis.label	Text label for x-axis. Defaults to "time".
y.axis.label	Text label for y-axis. Defaults to "function values".
smoother	A scalar boolean input indicating whether to co-plot a smoother line through the functions in each cluster.
sample_rate	A numeric value in (0,1] indicating percent of functions to randomly sample within each cluster to address over-plotting. Defaults to 1.
single_unit	A scalar boolean indicating whether to plot the fitted vs data curve for only a single experimental units (versus a random sample of 6). Defaults to single_unit = FALSE.
credible	A scalar boolean indicating whether to plot 95 percent credible intervals for estimated functions, bb, when plotting fitted functions versus data. Defaults to credible = FALSE
num_plot	A scalar integer indicating how many randomly-selected functions to plot (each in it's own plot panel) in the plot of functions versus the observed time series in the case that single_unit == TRUE. Defaults to num_plot = 6.

Value

A list object containing the plot of estimated functions, faceted by cluster, and the associated data.frame object.

p.cluster	A ggplot2 plot object
dat.cluster	A data.frame object used to generate p.cluster.

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow, gmrfdpgrow

Examples

```
{
library(growfunctions)

## load the monthly employment count data for a collection of
## U.S. states from the Current
## Population Survey (cps)
data(cps)
## subselect the columns of N x T, y, associated with
## the years 2008 - 2013
## to examine the state level employment levels
## during the "great recession"
y_short             <- cps$y[,(cps$yr_label %in% c(2008:2013))]

## Run the DP mixture of iGMRF's to estimate posterior
## distributions for model parameters
## Under default RW2(kappa) = order 2 trend
## precision term
res_gmrf            <- gmrfdpgrow(y = y_short,
                                  n.iter = 40,
                                  n.burn = 20,
                                  n.thin = 1)

## 2 plots of estimated functions: 1. faceted by cluster and fit;
## 2.  data for experimental units.
## for a group of randomly-selected functions
fit_plots_gmrf      <- cluster_plot( object = res_gmrf,
                                  units_name = "state",
                                  units_label = cps$st,
                                  single_unit = FALSE,
                                  credible = TRUE )
}
```

---

cps *Monthly employment counts from 1990 - 2013 from the Current Population Survey*

Description

Monthly employment counts published by the U.S. Bureau of Labor Statistics in the Current Population Survey (CPS) for each of `N = 51` states (including the District of Columbia). This dataset covers `T = 278` months from *1990* the first two months of *2013*. The data include a N x T matrix, `y_raw`, of raw employment counts, as well as set of standardized values, `y`, where the standardization is done within state. The standardized data matrix is used in our `gpdpgrow` and `gmrfdpgrow` estimating functions because the standardization facilitates comparisons of the time-series across states.

Usage

```
cps
```

Format

A list object of 5 objects supporting a data matrix of N = 51 state time series for T = 278 months.

Details

- y. An (N = 51) x (T = 278) matrix of standardized employment count estimates for N = 51 states for T = 278 months, beginning in 1990. The counts are standardized to (0,1) for each state series
- y_raw. An N x T matrix of estimated monthly employment counts for N = 51 states.
- st. Two-digit labels for each of the N states in the order presented in y and y_raw.
- dte. A Date vector of length T that presents the set of dates (in y-m-d format) associated to the T time points presented in y and y_raw.
- yr. A number vector listing sequence of years, *1990 - 2013* included in the data set.
- yr_label. A numerical vector of length T = 278 with year labels for each monthly employment count in the cps data set.

---

fit_compare — *Side-by-side plot panels that compare latent function values to data for different estimation models*

Description

Uses as input the output object from the gpdpgrow() and gmrfdpgrow() functions.

Usage

```
fit_compare(objects, H = NULL, label.object = c("gp_rq", "gmrf_rw2"),
  units_name = "Observation_Unit", units_label = NULL, date_field = NULL,
  x.axis.label = NULL, y.axis.label = NULL)
```

Arguments

objects	A list input where each element is a returned object from estimation with either of gpdpgrow or gmrfdpgrow or an object that contains true N x T matrix of true latent function values, f . This latter input is only needed if want to compare estimated to true latent function values.
H	An N x 1 with entries in 1,...,M of cluster assignments for the N units of y under a known clustering.
label.object	A character vector of length equal to objects that contains labels for each element of objects to be used in rendering comparison plots. Defaults to label.object = c("gp_rq","gmrf_rw2").
units_name	A character input that provides a label for the set of N observation units. Defaults to units_name = "Observation_Unit".

units_label	A vector of labels to apply to the observation units with length equal to the number of unique units. Defaults to sequential numeric values as input with data, y.
date_field	A vector of Date values for labeling the x-axis tick marks. Defaults to 1:T .
x.axis.label	Text label for x-axis. Defaults to "time".
y.axis.label	Text label for y-axis. Defaults to "function values".

Value

A list object containing the plot of estimated functions, faceted by cluster, and the associated data.frame object.

p.t	A ggplot2 plot object
map	A data.frame object that contains clustering structure of observation units.

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow, gmrfdpgrow

Examples

```
{
library(growfunctions)

## load the monthly employment count data for a collection of
## U.S. states from the Current
## Population Survey (cps)
data(cps)
## subselect the columns of N x T, y, associated with
## the years 2009 - 2013
## to examine the state level employment levels
## during the "great recession"
y_short             <- cps$y[,(cps$yr_label %in% c(2009:2013))]

## run DP mixture of GP's to estimate posterior distributions
## for model parameters
## uses default setting of a single "rational quadratic"
## covariance formula
res_gp              <- gpdpgrow(y = y_short,
                                n.iter = 7,
                                n.burn = 3,
                                n.thin = 1,
                                n.tune = 0)
## 2 plots of estimated functions: 1. faceted by cluster and fit;
## 2.  data for experimental units.
## for a group of randomly-selected functions
fit_plots_gp        <- cluster_plot( object = res_gp,
```

```
                                    units_name = "state",
                                    units_label = cps$st,
                                    single_unit = FALSE,
                                    credible = TRUE )

## Run the DP mixture of iGMRF's to estimate posterior
## distributions for model parameters
## Under default RW2(kappa) = order 2 trend
## precision term
res_gmrf            <- gmrfdpgrow(y = y_short,
                                    n.iter = 35,
                                    n.burn = 15,
                                    n.thin = 1)

## 2 plots of estimated functions: 1. faceted by cluster and fit;
## 2.  data for experimental units.
## for a group of randomly-selected functions
fit_plots_gmrf      <- cluster_plot( object = res_gmrf,
                                    units_name = "state",
                                    units_label = cps$st,
                                    single_unit = FALSE,
                                    credible = TRUE )

## visual comparison of fit performance
## between gpdpgrow() and gmrfdpgrow()
## or any two objects returned from any
## combination of these estimation
## functions
objects                         <- vector("list",2)
objects[[1]]                    <- res_gmrf
objects[[2]]                    <- res_gp
label.object                    <- c("gmrf_tr2","gp_rq")
## the map data.frame object from fit_plots gp
## includes a field that
## identifies cluster assignments
## for each unit (or domain)
H                           <- fit_plots_gp$map$cluster
fit_plot_compare_facet      <- fit_compare( objects = objects,
                                 H = H,
                                 label.object = label.object,
                                 y.axis.label = "normalized y values",
                                 units_name = "state",
                                 units_label = cps$st)
}
```

---

gen_informative_sample

Generate a finite population and take an informative single or two-stage sample.

Description

Used to compare performance of sample design-weighted and unweighted estimation procedures.

Usage

```
gen_informative_sample(clustering = TRUE, two_stage = FALSE,
  theta = c(0.2, 0.7, 1), M = 3, theta_star = matrix(c(0.3, 0.3, 0.3,
  0.31, 0.72, 2.04, 0.58, 0.83, 1), 3, 3, byrow = TRUE), gp_type = "rq",
  N = 10000, T = 15, L = 10, R = 8, I = 4, n = 750,
  noise_to_signal = 0.05, incl_gradient = "medium")
```

Arguments

clustering	Boolean input on whether want population generated from clusters of covariance parameters. Defaults to clustering = FALSE
two_stage	Boolean input on whether want two stage sampling, with first stage defining set of L blocks, where membership in blocks determined by quantiles of observation unit variance functions. (They are structured like strata, though they are sub-sampled).
theta	A numeric vector of global covariance parameters in the case of clustering = FALSE. The length, P, of theta must be consistent with the selected gp_type. Defaults to theta = c(0.30.7,1.0) in the case of clustering = FALSE.
M	Scalar input denoting number of clusters to employ if clustering = TRUE. Defaults to M = 3
theta_star	An *P x M* matrix of cluster location values associated with the choice of M and the selected gp_type. Defaults to matrix(c(0.3,0.3,0.3,0.31,0.72,2.04,0.58,0.83,1.00),3,3,byrc
gp_type	Input of choice for covariance matrix formulation to be used to generate the functions for the N population units. Choices are c("se","rq"), where "se" denotes the squared exponential covariance function and "rq" denotes the rational quadratic. Defaults to gp_type = "se"
N	A scalar input denoting the number of population units (or establishments).
T	A scalar input denoting the number of time points in each of N, *T x 1* functions that contribute to the *N x T* population data matrix, y. Defaults to T = 15.
L	A scalar input that denotes the number of blocks in which to assign the population units to be sub-sampled in the first stage of sampling. Defaults to L = 10.
R	A scalar input that denotes the number of blocks to sample from L = 10 with probability proportional to the average variance of member functions in each block.
I	A scalar input denoting the number of strata to form within each block. Population units are divided into equally-sized strata based on variance quantiles. Defaults to I = 4.
n	Sample size to be generated. Both an informative sample under either single (two_stage = FALSE) or 2-stage (two_stage = TRUE) sample is taken, along with a non-informative, *iid* sample of the same size (n) from the finite population (generated with (clustering = TRUE) or without clustering). Defaults to n = 770.

`incl_gradient` A character input on whether stratum probabilities from lowest-to-highest is to `"high"`, in which case they are proportional to the exponential of the cluster number. If set to `"medium"` , the inclusion probabilities are proportional to the square of the cluster number. Note that population units are assigned to each stratum proportional to a progressively increasing quantile variance. The `incl_gradient` setting is used for both `two_stage = TRUE`, in which case it is applied to strata within block, as well as `two_stage = FALSE`, in which case a simple stratified random sample is conducted. Defaults to `incl_gradient = "medium"`

`noise_to_signal` A numeric input in the interval, `(0,1)`, denoting the ratio of noise variance to the average variance of the generated functions, `bb_i`. Defaults to `noise_to_signal = 0.05`

Value

A list object named `dat_sim` containing objects related to the generated sample finite population, the informative sample and the non-informative, *iid*, sample. Some important objects, include:

`H` A vector of length `N`, the population size, with cluster assignments for each establishment (unit) in `1,..M` clusters.

`map.tot` A `data.frame` object including unit label identifiers (under `establishment`), the cluster assignment (if `clustering = TRUE`), the block (if`two_stage = TRUE`) and stratum assignments and the sample inclusion probabilities.

`map.obs` A `data.frame` object configured the same as `map.tot`, only confined to those establishments/units selected into the *informative* sample of size `n`.

`map.iid` A `data.frame` object configured the same as `map.tot`, only confined to those establishments/units selected into the *non-informative*, iid sample of size `n`.

`(y,bb)` *N x T* `matrix` objects containing data responses and de-noised ' functions, respectively, for each of the `N` population units. The order of the `N` units is consistent with `map`.

`(y_obs,bb_obs)` *N x T* `matrix` objects containing observed responses and de-noised ' functions, respectively, for each of the `n` units sampled under an informative sampling design. The order of the `n` units is consistent with `map_obs`.

`(y_iid,bb_iid)` *N x T* `matrix` objects containing observed responses and de-noised ' functions, respectively, for each of the `n` units sampled under a non-informative / iid sampling design. The order of the `n` units is consistent with `map_iid`.

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

`gpdpgrow`, `gmrfdpgrow`

Examples

```
## Not run:
library(growfunctions)
## use gen_informative_sample() to generate an
## N X T population drawn from a dependent GP
## By default, 3 clusters are used to generate
## the population.
## A single stage stratified random sample of size n
## is drawn from the population using I = 4 strata.
## The resulting sample is informative in that the
## distribution for this sample is
## different from the population from which
## it was drawn because the strata inclusion
## probabilities are proportional to a feature
## of the response, y (in the case, the variance.
## The stratified random sample over-samples
## large variance strata).
## (The user may also select a 2-stage
## sample with the first stage
## sampling "blocks" of the population and
## the second stage sampling strata within blocks).
dat_sim        <- gen_informative_sample(N = 10000,
                                n = 500, T = 10,
                                noise_to_signal = 0.1)

## extract n x T observed sample under informative
## stratified sampling design.
y_obs                        <- dat_sim$y_obs
T                            <- ncol(y_obs)

## End(Not run)
```

---

gmrfdpgrow *Bayesian instrinsic Gaussian Markov Random Field model for dependent time-indexed functions*

Description

Estimates a collection of time-indexed functions under intrinsic Gaussian Markov random field prior formulations where a Dirichlet process mixture allows sub-groupings of the functions to share the same iGMRF precision parameter. The iGMRF formulation supports any number of additive precision terms, expressing either or both of multiple trend and seasonality.

Usage

```
gmrfdpgrow(y, ipr, q_order, q_type, q_shape, q_rate, tau_shape, tau_rate,
  dp_shape, dp_rate, M_init, n.iter, n.burn, n.thin, progress, jitter,
  kappa_fast)
```

Arguments

y	A multivariate continuous response, specified as an $N \times T$ matrix, where N denotes the number of functions and T, the number of time points per function. Intermittent missing-at-random values are allowed and will be estimated from the posterior predictive distribution. Missing cells should be denoted with NA.
ipr	An optional input vector of inclusion probabilities for each observation unit in the case the observed data were acquired through an informative sampling design, so that unbiased inference about the population requires adjustments to the observed sample. Defaults to `ipr = rep(1,nrow(y))` indicating an iid sample.
q_order	An integer vector of length K to select the order for each iGMRF precision term. e.g. If the first term is a RW2 and there is a second is a 3-month seasonality term, where the time points are indexed by month, then `q_order = c(2,3)`. Defaults to `q_order = 2`
q_type	A vector of length K, the number of iGMRF precision terms, with each entry indicating whether the associated term is a trend (`"tr"`) or seasonality (`"sn"`) term. So all entries must be one of `c("tr","sn")`. Defaults to `q_type = "tr"`.
q_shape	The value (in (0,infty)) for the shape hyperparameter for the Gamma base distribution for the iGMRF scale parameters, `kappa_star(k,m)`, where k denotes the term and m, the cluster. Defaults to `q_shape = 0.3`.
q_rate	The rate parameter of the Gamma base distribution on kappa_star. Defaults to `q_rate = 0.0005`.
tau_shape	The value (in (0,infty)) for the shape hyperparameter for the Gamma prior on the error precision parameter. Defaults to `tau_shape = 1.0`.
tau_rate	The rate parameter of the Gamma prior distribution on tau_e. Defaults to `tau_rate = 1`.
dp_shape	The shape parameter for the Gamma prior on the DP concentration parameter, conc. Defaults to `dp_shape = 1`.
dp_rate	The rate parameter for the Gamma prior on the DP concentration parameter, conc. Defaults to `dp_rate = 1`.
M_init	Starting number of clusters of `nrow(y)` units to initialize sampler. Defaults to `M_init = nrow(y)`.
n.iter	Total number of MCMC iterations.
n.burn	Number of MCMC iterations to discard. gmrfdpgrow will return `(n.iter - n.burn)` posterior samples.
n.thin	Gap between successive sampling iterations to save.
progress	A boolean value denoting whether to display a progress bar during model execution. Defaults to `progress = true`.
jitter	A scalar double indicating amount of jitter to subract from the posterior rate and shape hyperparameters of tau_e to stabilize computation. Defaults to `jitter = 0.0`.
kappa_fast	Boolean for whether to generate rate hyperparameter from full conditionals versus joint Gaussian (on random effects, bb, given kappa. The former is faster, but numerically less stable. Defaults to `kappa_fast = FALSE`.

Value

S3 gmrfdpgrow object, for which many methods are available to return and view results. Generic functions applied to an object, res of class gmrfdpgrow, includes:

plot(res)	returns results plots, including fit functions versus data and allocation of fitted functions into clusters
samples(res)	contains (n.iter - n.burn) posterior sampling iterations for every model parameter
resid(res)	contains the model residuals.

Note

The intended focus for this package are data composed of observed noisy functions (each of length T) for a set of experimental units where the functions may express dependence among the experimental units

Author(s)

Terrance Savitsky <tds151@gmail.com> Daniell toth <danielltoth@yahoo.com>

References

T. D. Savitsky and D. Toth (2014) Bayesian Non-parametric Models for Collections of Time- indexed Functions. submitted to: JRSS Series A (Statistics in Society).

T. D. Savitsky (2014) Bayesian Non-parametric Functional Mixture Estimation for Time-indexed data. submitted to: Annals of Applied Statistics.

T. D. Savitsky (2014) Bayesian Non-Parametric Mixture Estimation for Time-Indexed Functional Data for R. Submitted to: Journal of Statistical Software.

See Also

gmrfdpgrow

Examples

```
{
library(growfunctions)

## load the monthly employment count data for a collection of
## U.S. states from the Current
## Population Survey (cps)
data(cps)
## subselect the columns of N x T, y, associated
## with the years 2008 - 2013
## to examine the state level employment levels
## during the "great recession"
y_short   <- cps$y[,(cps$yr_label %in% c(2008:2013))]

## Run the DP mixture of iGMRF's to estimate posterior
```

```
## distributions for model parameters
## Under default RW2(kappa) = order 2 trend
## precision term
## Run for 1500 iterations, with half as burn-in for a
## more useful (converged) result.
res_gmrf            <- gmrfdpgrow(y = y_short,
                                  n.iter = 40,
                                  n.burn = 20,
                                  n.thin = 1)

## 2 plots of estimated functions: 1. faceted by cluster and fit;
## 2.  data for experimental units.
## for a group of randomly-selected functions
fit_plots_gmrf      <- cluster_plot( object = res_gmrf,
                                  units_name = "state",
                                  units_label = cps$st,
                                  single_unit = FALSE,
                                  credible = TRUE )
}
```

---

gmrfdpPost *Run a Bayesian functional data model under an instrinsic GMRF prior whose precision parameters employ a DP prior*

Description

An internal function to gmrfdpgrow

Usage

```
gmrfdpPost(y, ipr, C, D, q_order, q_type, n.iter, n.burn, n.thin, M_init,
  q_shape, q_rate, tau_shape, tau_rate, dp_shape, dp_rate, progress, jitter,
  kappa_fast)
```

Arguments

y	An *N x T* matrix of N observations of *T x 1* functions
ipr	An optional input vector of inclusion probabilities for each observation unit in the case the observed data were acquired through an informative sampling design, so that unbiased inference about the population requires adjustments to the observed sample. Defaults to `ipr = rep(1,nrow(y))` indicating an iid sample.
C	A list object of length, K, the number of iGMRF precision terms. Each entry contains a *T x T* normalized adjacency matrix. The diagonal entries are `0` and row `i` contains the weight for each entry !=i divided by the sum of the weights.
D	A *K x T* matrix, where K denotes the number of iGMRF terms. Row k contains the T elements of the diagonal of the term-k precision matrix, Q_k. Will increase with order and be equal, except for boundary corrections.

q_order	An integer vector where each entry contains the order of the associated K iGMRF precision terms matrix of Euclidean distances associated to each seasonal covariance term.
q_type	A vector of length K, the number of iGMRF precision terms, with each entry indicating whether the associated term is a trend ("tr") or seasonality ("sn") term.
tau_shape	The value (in (0,infty)) for the shape hyperparameter for the Gamma prior on the error precision parameter. Defaults to tau_shape = 1.0.
tau_rate	The rate parameter of the Gamma prior distribution on tau_e. Defaults to tau_rate = 1.
n.iter	The number of MCMC sampling iterations
n.burn	The number of warm-up iterations to discard
n.thin	The interval or step size of post-burn-in samples to return
M_init	Starting value of number of clusters for sampling cluster assignments.
q_shape	The shape parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
q_rate	The rate parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
dp_shape	The shape parameter for the Γ prior on the DP concentration parameter. The rate parameter is set of 1.
dp_rate	The rate parameter for the Γ prior on the DP concentration parameter. Default value is 1.
progress	An indicator in {0,1} denoting whether to display a progress bar during model execution. progress = 1 displays a progress bar. Defaults to progress = 1.
jitter	A scalar double indicating amount of jitter to subract from the posterior rate and shape hyperparameters of tau_e to stabilize computation. Defaults to jitter = 0.0.
kappa_fast	Boolean for whether to generate rate hyperparameter from full conditionals versus joint Gaussian (on random effects, bb, given kappa. The former is faster, but numerically less stable. Defaults to kappa_fast = FALSE.

Value

res A list object containing MCMC runs for all model parameters.

Note

Intended as an internal function for gmrfdpgrow

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow

gpBFixPost *Run a Bayesian functional data model under a GP prior with a fixed clustering structure that co-samples latent functions, bb_i.*

Description

An internal function to gpdpgrow

Usage

```
gpBFixPost(y, ipr, Omega_t, Omega_s, gp_mod, jitter, gp_shape, gp_rate,
  noise_shape, noise_rate, lower, upper, w, n_slice_iter, y_index, n.iter,
  n.burn, n.thin, n.tune, progress, s)
```

Arguments

y	An *N x T* matrix of N observations of *T x 1* functions. y may\ have intermittant missing values. They should be input with NA.
ipr	An optional input vector of inclusion probabilities for each observation unit in the case the observed data were acquired through an informative sampling design, so that unbiased inference about the population requires adjustments to the observed sample. Defaults to ipr = rep(1,nrow(y)) indicating an iid sample.
Omega_t	A *T x T* matrix of squared Eucidean distances for T time points
Omega_s	A list object of length L_s, where each contains the *T x T* matrix of Euclidean distances associated to each seasonal covariance term.
gp_mod	An *L x 1* numeric vector denoting the selected covariance function for each of L terms. gp_mod = 1 is "rq". gp_mod = 2 is "se". gp_mod = 3 is "sn".
jitter	Numeric value added to diagonals of GP covariance matrix to stabilize inversion.
gp_shape	The shape parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
gp_rate	The rate parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
noise_shape	The shape parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to noise_shape = 3.
noise_rate	The rate parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to noise_rate = 1.
lower	Minimum in range of support for GP covariance parameters, kappa.
upper	Maximum in range of support for GP covariance parameters, kappa.
w	Tuning parameter for slice sampling interval width used for GP covariance parameters, kappa.

n_slice_iter	Maximum number of steps to widen slice samplind width for GP covariance parameters, kappa.
y_index	List object where each contains index of time points to use in n progressively coarser distribution for sampling kappa in tempered update steps.
n.iter	The number of MCMC sampling iterations
n.burn	The number of warm-up iterations to discard
n.thin	The interval or step size of post-burn-in samples to return
n.tune	The number of tuning iterations to update the slice sampler width, w.
progress	An indicator in {0,1} denoting whether to display a progress bar during model execution. progress = 1 displays a progress bar. Defaults to progress = 1.
s	An integer vector inputting cluster membership structure if select fix == TRUE.

Value

res A list object containing MCMC runs for all model parameters.

Note

Intended as an internal function for gpdpgrow

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow

---

gpdpbPost *Run a Bayesian functional data model under a GP prior whose parameters employ a DP prior*

Description

An internal function to gpdpgrow

Usage

```
gpdpbPost(y, ipr, Omega_t, Omega_s, gp_mod, jitter, b_move, gp_shape, gp_rate,
  noise_shape, noise_rate, lower, upper, w_star, w, n_slice_iter, y_index,
  n.iter, n.burn, n.thin, n.tune, M_init, dp_shape, dp_rate, progress)
```

Arguments

y	An *N x T* matrix of N observations of *T x 1* functions. y may\ have intermittant missing values. They should be input with NA.
ipr	An optional input vector of inclusion probabilities for each observation unit in the case the observed data were acquired through an informative sampling design, so that unbiased inference about the population requires adjustments to the observed sample. Defaults to ipr = rep(1,nrow(y)) indicating an iid sample.
Omega_t	A *T x T* matrix of squared Eucidean distances for T time points
Omega_s	A list object of length L_s, where each contains the *T x T* matrix of Euclidean distances associated to each seasonal covariance term.
gp_mod	An *L x 1* numeric vector denoting the selected covariance function for each of L terms. gp_mod = 1 is "rq". gp_mod = 2 is "se". gp_mod = 3 is "sn".
jitter	Numeric value added to diagonals of GP covariance matrix to stabilize inversion.
b_move	An indicator in {0,1} denoting whether to sample GP functions, (bb_i) in a *T x 1* Gibbs step or through elliptical slice sampling. b_move = 1 samples in a Gibbs step. Defaults to b_move = 1.
gp_shape	The shape parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
gp_rate	The rate parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
noise_shape	The shape parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to noise_shape = 3.
noise_rate	The rate parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to noise_rate = 1.
lower	Minimum in range of support for GP covariance parameters, kappa.
upper	Maximum in range of support for GP covariance parameters, kappa.
w_star	Tuning parameter for number of locations to sample not linked to observations in the auxiliary Gibbs sampler for cluster assignments.
w	Tuning parameter for slice sampling interval width used for GP covariance parameters, kappa.
n_slice_iter	Maximum number of steps to widen slice samplind width for GP covariance parameters, kappa.
y_index	List object where each contains index of time points to use in n progressively coarser distribution for sampling kappa in tempered update steps.
n.iter	The number of MCMC sampling iterations
n.burn	The number of warm-up iterations to discard
n.thin	The interval or step size of post-burn-in samples to return
n.tune	The number of tuning iterations to update the slice sampler width, w.
M_init	Starting value of number of clusters for sampling cluster assignments.

dp_shape	The shape parameter for the Γ prior on the DP concentration parameter. The rate parameter is set of 1.
dp_rate	The rate parameter for the Γ prior on the DP concentration parameter. Default value is 1.
progress	An indicator in {0,1} denoting whether to display a progress bar during model execution. progress = 1 displays a progress bar. Defaults to progress = 1.

Value

res A list object containing MCMC runs for all model parameters.

Note

Intended as an internal function for gpdpgrow

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow

---

gpdpgrow *Bayesian non-parametric dependent Gaussian process model for time-indexed functional data*

Description

Estimates a collection of time-indexed functions with Gaussian process (GP) formulations where a Dirichlet process mixture allows sub-groupings of the functions to share the same GP covariance parameters. The GP formulation supports any number of additive GP covariance terms, expressing either or both of multiple trend and seasonality.

Usage

```
gpdpgrow(y, ipr, time_points, gp_cov, sn_order, jitter, gp_shape, gp_rate,
  noise_shape, noise_rate, dp_shape, dp_rate, M_init, lower, upper, sub_size,
  w_star, w, n.iter, n.burn, n.thin, n.tune, progress, b_move, cluster, s)
```

Arguments

y	A multivariate continuous response, specified as an $N \times T$ matrix, where N denotes the number of functions and T, the number of time points per function. Intermittent missing-at-random values are allowed and will be estimated from the posterior predictive distribution. Missing cells should be denoted with NA. The sampling of missing values requires co-sampling the functions, bb, while these functions are marginalized out for sampling under the case of no missing data. So the sampler will run more slowly in the case of intermittent missingness.
ipr	An optional input vector of inclusion probabilities for each observation unit in the case the observed data were acquired through an informative sampling design, so that unbiased inference about the population requires adjustments to the observed sample. Defaults to `ipr = rep(1,nrow(y))` indicating an iid sample.
time_points	Inputs a vector of common time points at which the collections of functions were observed (with the possibility of intermittent missingness). The length of time_points should be equal to the number of columns in the data matrix, y. Defaults to `time_points = 1:ncol(y)`.
gp_cov	A vector of length L to select the covariance function for each of L terms. Allowed inputs are `c("rq","se","sn")`, where "rq" denotes the rational quadratic covariance function, "se", the square exponential, and "sn", seasonality. The seasonality covariance is a quasi-periodic multiplicative combination of a fixed length- scale periodic covariance kernel (where the period length is specified in 'sn_order') and a squared exponential kernel (of varying length-scale) to allow the periodicity in the seasonal term to evolve. e.g. If 4 terms are wanted - 2 trend terms with "se" and 2 seasonality terms, the input would be `gp_cov = c("se","se","sn","sn")`. Defaults to `gp_cov = "rq"`.
sn_order	A vector of length L_s, the number of terms in gp_cov where "sn" is selected, that denotes the seasonality order for each term; e.g. if the two "sn" terms above are for 3 and 12 month seasonality, respectively, for monthly data, then `sn_order = c(3,12)`. Defaults to `sn_order = NULL`.
jitter	A scalar numerical value added to the diagonal elements of the T x T GP covariance matrix to stabilize computation. Defaults to `jitter = 0.01`.
gp_shape	The shape parameter of the Gamma base distribution for the DP prior on the P x N matrix of GP covariance parameters (where P denotes the number of parameters for each of the N experimental units). Defaults to `gp_shape = 1`
gp_rate	The rate parameter of the Gamma base distribution on GP covariance parameters. Defaults to `gp_rate = 1`.
noise_shape	The shape parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to `noise_shape = 3`.
noise_rate	The rate parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to `noise_rate = 1`.
dp_shape	The shape parameter for the Gamma prior on the DP concentration parameter, conc. Defaults to `dp_shape = 1`.
dp_rate	The rate parameter for the Gamma prior on the DP concentration parameter, conc. Defaults to `dp_rate = 1`.

M_init — Starting number of clusters of nrow(y) units to initialize sampler. Defaults to M_init = nrow(y).

lower — The lower end of the range to be used in conditionally sampling the GP covariance parameters (kappa,tau_e) in the slice sampler. Defaults to lower = 0.

upper — The upper end of the range to be used in conditionally sampling the GP covariance parameters (kappa,tau_e) in the slice sampler. Defaults to upper = 1e10.

sub_size — Integer vector whose length, n, equals the number of progressively coarser GP covariance matrices to use for tempered sampling steps in an alternative space to sample the GP covariance parameters. Each entry denotes the number of sub-sample time points in T to draw under a latin hypercube design from the *N x T* data matrix, y to employ for that distribution; for example, suppose T = 300. sub_size = c(100,50), would randomly select first 100 and then 50 time points from y to use for each of the n = 2 distributions. Defaults to sub_size = c(floor(0.25*T),floor(0.1*T)).

w_star — Integer value denoting the number of cluster locations to sample ahead of observations in the auxiliary Gibbs sampler used to sample the number of clusters and associated cluster assignments. A higher value reduces samplin autocorrelation, but increases computational burden. Defaults to w_star = 2.

w — Numeric value denoting the step width used to construct the interval from which to draw a sample for each GP covariance parameter in the slice sampler. This value is adaptively updated in the sampler tuning stage for each parameter to be equal to the difference in the 0.95 and 0.05 sample quantiles for each of 5 block updates. Defaults to w = 1.5.

n.iter — Total number of MCMC iterations.

n.burn — Number of MCMC iterations to discard. gpdpgrow will return (n.iter - n.burn) posterior samples.

n.thin — Gap between successive sampling iterations to save.

n.tune — Number of iterations (before ergodic chain instantiated) to adapt w, separately, for each covariance term, p = 1,...,P. Sets each w_p to lie in the 90 percent credible interval computed from the tuning sample (that is divided into 5 blocks so that w_p is successively updated in each block of runs).

progress — A boolean value denoting whether to display a progress bar during model execution. Defaults to progress = TRUE

b_move — A boolean value denoting whether to sample the GP function, bb, in *T x 1* Gibbs steps b_move = TRUE or through elliptical slice sampling. Defaults to b_move = TRUE. Only used in the case there is any intermittent missingness; otherwise bb is marginalized out of the sampler and post-sampled from the its predictive distribution.

cluster — A boolean value denoting whether to employ DP mix model over set of GP functions or to just use GP model with no clustering of covariance function parameters. Defaults to cluster = TRUE

s — An *N x 1* integer vector that inputs a fixed clustering, rather than sampling it. Defaults to s = NULL

Value

S3 gpdpgrow object, for which many methods are available to return and view results. Generic functions applied to an object, res of class gpdpgrow, includes:

samples(res)	contains (n.iter - n.burn) posterior sampling iterations for every model parameter
resid(res)	contains the model residuals.

Note

The intended focus for this package are data composed of observed noisy functions (each of length T) for a set of experimental units where the functions may express dependence among the experimental units

Author(s)

Terrance Savitsky <tds151@gmail.com> Daniell Toth <danielltoth@yahoo.com>

References

T. D. Savitsky and D. Toth (2014) Bayesian Non-parametric Models for Collections of Time- indexed Functions. submitted to: JRSS Series A (Statistics in Society).

T. D. Savitsky (2014) Bayesian Non-parametric Functional Mixture Estimation for Time-indexed data. submitted to: Annals of Applied Statistics.

T. D. Savitsky (2014) Bayesian Non-Parametric Mixture Estimation for Time-Indexed Functional Data for R. Submitted to: Journal of Statistical Software.

See Also

gmrfdpgrow

Examples

```
{
library(growfunctions)

## load the monthly employment count data for a collection of
## U.S. states from the Current
## Population Survey (cps)
data(cps)
## subselect the columns of N x T, y, associated with
## the years 2009 - 2013
## to examine the state level employment
## levels during the "great recession"
y_short   <- cps$y[,(cps$yr_label %in% c(2009:2013))]

## uses default setting of a single "rational quadratic" covariance
## run for 500 iterations, with half discarded as burn-in to
## obtain a more useful result.
res_gp              <- gpdpgrow(y = y_short,
```

```
                                n.iter = 8,
                                n.burn = 4,
                                n.thin = 1,
                                n.tune = 0)

## Two plots of estimated functions,
## 1. faceted by cluster
## 2. fitted functions vs noisy observations
## first plot will plot estimated denoised function,
## bb_i, for a single (randomly-selected) "state"
fit_plots_gp        <- cluster_plot( object = res_gp,
                           units_name = "state",
                           units_label = cps$st,
                           single_unit = TRUE,
                           credible = TRUE )
## second plot will randomly select 6 states
## and plot their estimated denoised functions, bb_i.
## with setting "single_unit = FALSE".
## (Option "num_plot" may be set to plot
## any integer number of
## randomly-selected units.)
fit_plots_gp        <- cluster_plot( object = res_gp,
                                     units_name = "state",
                                     units_label = cps$st,
                                     single_unit = FALSE,
                                     credible = TRUE )

}
```

---

gpdpPost *Run a Bayesian functional data model under a GP prior whose parameters employ a DP prior*

Description

An internal function to gpdpgrow

Usage

```
gpdpPost(y, ipr, Omega_t, Omega_s, gp_mod, jitter, gp_shape, gp_rate,
  noise_shape, noise_rate, lower, upper, w_star, w, n_slice_iter, y_index,
  n.iter, n.burn, n.thin, n.tune, M_init, dp_shape, dp_rate, progress)
```

Arguments

y	An *N x T* matrix of N observations of *T x 1* functions
ipr	An optional input vector of inclusion probabilities for each observation unit in the case the observed data were acquired through an informative sampling design, so that unbiased inference about the population requires adjustments to the observed sample. Defaults to `ipr = rep(1,nrow(y))` indicating an iid sample.

Omega_t	A *T x T* matrix of squared Eucidean distances for T time points
Omega_s	A list object of length L_s, where each contains the *T x T* matrix of Euclidean distances associated to each seasonal covariance term.
gp_mod	An *L x 1* numeric vector denoting the selected covariance function for each of L terms. gp_mod = 1 is "rq". gp_mod = 2 is "se". gp_mod = 3 is "sn".
jitter	Numeric value added to diagonals of GP covariance matrix to stabilize inversion.
gp_shape	The shape parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
gp_rate	The rate parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
noise_shape	The shape parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to noise_shape = 3.
noise_rate	The rate parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to noise_rate = 1.
lower	Minimum in range of support for GP covariance parameters, kappa.
upper	Maximum in range of support for GP covariance parameters, kappa.
w_star	Tuning parameter for number of locations to sample not linked to observations in the auxiliary Gibbs sampler for cluster assignments.
w	Tuning parameter for slice sampling interval width used for GP covariance parameters, kappa.
n_slice_iter	Maximum number of steps to widen slice samplind width for GP covariance parameters, kappa.
y_index	List object where each contains index of time points to use in n progressively coarser distribution for sampling kappa in tempered update steps.
n.iter	The number of MCMC sampling iterations
n.burn	The number of warm-up iterations to discard
n.thin	The interval or step size of post-burn-in samples to return
n.tune	The number of tuning iterations to update the slice sampler width, w.
M_init	Starting value of number of clusters for sampling cluster assignments.
dp_shape	The shape parameter for the Γ prior on the DP concentration parameter. The rate parameter is set of 1.
dp_rate	The rate parameter for the Γ prior on the DP concentration parameter. Default value is 1.
progress	An indicator in {0,1} denoting whether to display a progress bar during model execution. progress = 1 displays a progress bar. Defaults to progress = 1.

Value

res A list object containing MCMC runs for all model parameters.

Note

Intended as an internal function for gpdpgrow

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow

---

gpFixPost *Run a Bayesian functional data model under a GP prior whose parameters employ a DP prior*

Description

An internal function to gpdpgrow

Usage

```
gpFixPost(y, ipr, Omega_t, Omega_s, gp_mod, jitter, gp_shape, gp_rate,
  noise_shape, noise_rate, lower, upper, w, n_slice_iter, y_index, n.iter,
  n.burn, n.thin, n.tune, progress, s)
```

Arguments

y	An *N x T* matrix of N observations of *T x 1* functions
ipr	An optional input vector of inclusion probabilities for each observation unit in the case the observed data were acquired through an informative sampling design, so that unbiased inference about the population requires adjustments to the observed sample. Defaults to `ipr = rep(1,nrow(y))` indicating an iid sample.
Omega_t	A *T x T* matrix of squared Eucidean distances for T time points
Omega_s	A `list` object of length `L_s`, where each contains the *T x T* matrix of Euclidean distances associated to each seasonal covariance term.
gp_mod	An *L x 1* numeric vector denoting the selected covariance function for each of `L` terms. `gp_mod = 1` is `"rq"`. `gp_mod = 2` is `"se"`. `gp_mod = 3` is `"sn"`.
jitter	Numeric value added to diagonals of GP covariance matrix to stabilize inversion.
gp_shape	The shape parameter of the Gamma base distribution for the `kappa_star` locations used to sample the DP prior on the `P` GP covariance parameters, `kappa`, for each experimental unit.
gp_rate	The rate parameter of the Gamma base distribution for the `kappa_star` locations used to sample the DP prior on the `P` GP covariance parameters, `kappa`, for each experimental unit.

noise_shape	The shape parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to noise_shape = 3.
noise_rate	The rate parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to noise_rate = 1.
lower	Minimum in range of support for GP covariance parameters, kappa.
upper	Maximum in range of support for GP covariance parameters, kappa.
w	Tuning parameter for slice sampling interval width used for GP covariance parameters, kappa.
n_slice_iter	Maximum number of steps to widen slice samplind width for GP covariance parameters, kappa.
y_index	List object where each contains index of time points to use in n progressively coarser distribution for sampling kappa in tempered update steps.
n.iter	The number of MCMC sampling iterations
n.burn	The number of warm-up iterations to discard
n.thin	The interval or step size of post-burn-in samples to return
n.tune	The number of tuning iterations to update the slice sampler width, w.
progress	An indicator in {0,1} denoting whether to display a progress bar during model execution. progress = 1 displays a progress bar. Defaults to progress = 1.
s	An integer vector inputting cluster membership structure if select fix == TRUE.

Value

res A list object containing MCMC runs for all model parameters.

Note

Intended as an internal function for gpdpgrow

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow

gpPost *Run a Bayesian functional data model under a GP prior whose parameters employ a DP prior*

Description

An internal function to gpdpgrow

Usage

```
gpPost(y, ipr, Omega_t, Omega_s, gp_mod, jitter, gp_shape, gp_rate, noise_shape,
  noise_rate, lower, upper, w, n_slice_iter, y_index, n.iter, n.burn, n.thin,
  n.tune, progress)
```

Arguments

y	An *N x T* matrix of N observations of *T x 1* functions
ipr	An optional input vector of inclusion probabilities for each observation unit in the case the observed data were acquired through an informative sampling design, so that unbiased inference about the population requires adjustments to the observed sample. Defaults to `ipr = rep(1,nrow(y))` indicating an iid sample.
Omega_t	A *T x T* matrix of squared Eucidean distances for T time points
Omega_s	A list object of length L_s, where each contains the *T x T* matrix of Euclidean distances associated to each seasonal covariance term.
gp_mod	An *L x 1* numeric vector denoting the selected covariance function for each of L terms. `gp_mod = 1` is "rq". `gp_mod = 2` is "se". `gp_mod = 3` is "sn".
jitter	Numeric value added to diagonals of GP covariance matrix to stabilize inversion.
gp_shape	The shape parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
gp_rate	The rate parameter of the Gamma base distribution for the kappa_star locations used to sample the DP prior on the P GP covariance parameters, kappa, for each experimental unit.
noise_shape	The shape parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to `noise_shape = 3`.
noise_rate	The rate parameter of the Gamma base distribution on tau_e, the model noise precision parameter. Defaults to `noise_rate = 1`.
lower	Minimum in range of support for GP covariance parameters, kappa.
upper	Maximum in range of support for GP covariance parameters, kappa.
w	Tuning parameter for slice sampling interval width used for GP covariance parameters, kappa.
n_slice_iter	Maximum number of steps to widen slice samplind width for GP covariance parameters, kappa.

y_index	List object where each contains index of time points to use in n progressively coarser distribution for sampling kappa in tempered update steps.
n.iter	The number of MCMC sampling iterations
n.burn	The number of warm-up iterations to discard
n.thin	The interval or step size of post-burn-in samples to return
n.tune	The number of tuning iterations to update the slice sampler width, w.
progress	An indicator in {0,1} denoting whether to display a progress bar during model execution. progress = 1 displays a progress bar. Defaults to progress = 1.

Value

res A list object containing MCMC runs for all model parameters.

Note

Intended as an internal function for gpdpgrow

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow

---

informative_plot *Plot credible intervals for parameters to compare ignoring with weighting an informative sample*

Description

Uses as input the output object from the gpdpgrow() and gmrfdpgrow() functions.

Usage

```
informative_plot(objects = NULL, objects_labels = c("ignore", "weight"),
  map = NULL, units_name = NULL, model = "gp", true_star = NULL,
  map_true = NULL)
```

Arguments

objects	A list of objects, either all outputs from gpdpgrow(), or all from gmrfdpgrow(). objects includes a model estimated under ignoring the informativeness of the sampling design and another that employs weighting to account for the informativeness. An additional object may be added to represent a separate "iid" (or non-informative) sample from the same population, which will typically be available if the dataset was generated as synthetic data using function, "gen_informative_sample()".
objects_labels	A character vector of length equal to objects that provides labels for each entry in objects. Allowed entries in objects_labels are c("ignore","weight","iid"), where "ignore" denotes a model that ignores the informativeness, while "weight" denotes a model that employs sampling weights and "iid" denotes a model run on a non-informative, iid sample from the same population. Defaults to objects_labels = c("ignore","weight").
map	A list matrices, where each entry is produced from cluster_plot(object)$map. It is comprised of unit labels and cluster assignments for each object in objects. The length of map must be equal to the length of objects.
units_name	The label in each "map" matrix for the observation units. Will be the same as the units_name entry for the previously run, cluster_plot() function.
model	A scalar character input indicating the estimation model for *all* of the entries in objects. Allowable values for model are c("gp","gmrf"). Defaults to model = "gp".
true_star	An optional, P *x* M matrix, of true parameter location values, where P denotes the number of parameters per cluster and M denotes the number of clusters. For example, in model = "gp" with a single, rational quadratic covariance, P = 3 and if there are 3 clusters, then M = 3. For a model = "gmrf", with a single covariance, P = 1.
map_true	An optional data.frame object with n rows, the size of the informative sample used for c("ignore","weight") objects that maps the units_name to a true cluster. map_true must have 2 columns (and the rest are ignored), one must be named the same value as input for units_name. The second column must be named, cluster. If the true values derive from running gen_informative_sample() as the source of the true values, one may just input the map_obs data.frame that is listed in the object returned by gen_informative_sample().

Value

A list object containing the plot of estimated functions, faceted by cluster, and the associated data.frame object.

p.compare	A ggplot2 plot object
dat.compare	A data.frame object used to generate p.compare.

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow, gmrfdpgrow

Examples

```
## Not run:
library(growfunctions)
## use gen_informative_sample() to generate an
## N X T population drawn from a dependent GP
## By default, 3 clusters are used to generate
## the population.
## A single stage stratified random sample of size n
## is drawn from the population using I = 4 strata.
## The resulting sample is informative in that the
## distribution for this sample is
## different from the population from which
## it was drawn because the strata inclusion
## probabilities are proportional to a feature
## of the response, y (in the case, the variance.
## The stratified random sample over-samples
## large variance strata).
## (The user may also select a 2-stage
## sample with the first stage
## sampling "blocks" of the population and
## the second stage sampling strata within blocks).
dat_sim        <- gen_informative_sample(N= 10000,
                                n = 500, T = 10,
                                noise_to_signal = 0.1)

y_obs                       <- dat_sim$y_obs
T                           <- ncol(y_obs)

an informative sampling design that inputs inclusion
probabilities, ipr
res_gp_w            <- gpdpgrow(y = y_obs,
                                ipr = dat_sim$map_obs$incl_prob,
                                n.iter = 10, n.burn = 4,
                                n.thin = 1, n.tune = 0)
and fit vs. data for experimental units
fit_plots_w         <- cluster_plot( object = res_gp_w,
                              units_name = "establishment",
                              units_label = dat_sim$map_obs$establishment,
                              single_unit = FALSE, credible = TRUE )

## estimate parameters ignoring sampling design
res_gp_i            <- gpdpgrow(y = y_obs,
                                n.iter = 10, n.burn = 4,
                                n.thin = 1, n.tune = 0)
## plots of estimated functions, faceted by cluster and fit vs.
## data for experimental units
fit_plots_i         <- cluster_plot( object = res_gp_i,
                                     units_name = "establishment",
                                     units_label = dat_sim$map_obs$establishment,
                                     single_unit = FALSE, credible = TRUE )

## We also draw an iid (non-informative, exchangeable)
```

```
## sample from the same population to
## compare estimation results to our weighted
## (w) and unweighted/ignoring (i) models

## estimate parameters under an iid sampling design
res_gp_iid          <- gpdpgrow(y = dat_sim$y_iid,
                                n.iter = 10, n.burn = 4,
                                n.thin = 1, n.tune = 0)
## plots of estimated functions, faceted by cluster and
## fit vs. data for experimental units
fit_plots_iid       <- cluster_plot( object = res_gp_iid,
                            units_name = "establishment",
                            units_label = dat_sim$map_iid$establishment,
                            single_unit = FALSE, credible = TRUE )

## compare estimations of covariance parameter credible
## intervals when ignoring informativeness vs.
## weighting to account for informativeness
objects                  <- map <- vector("list",3)
objects[[1]]             <- res_gp_i
objects[[2]]             <- res_gp_iid
objects[[3]]             <- res_gp_w
map[[1]]                 <- fit_plots_i$map
map[[2]]                 <- fit_plots_iid$map
map[[3]]                 <- fit_plots_w$map
objects_labels           <- c("ignore","iid","weight")

parms_plots_compare      <- informative_plot( objects = objects,
                                objects_labels = objects_labels,
                                map = map, units_name = "establishment",
                                model = "gp",
                                true_star = dat_sim$theta_star,
                                map_true = dat_sim$map_obs)


## End(Not run)
```

MSPE *Compute normalized mean squared prediction error based on accuracy to impute missing data values*

Description

Uses as input the output object from the gpdpgrow() and gmrfdpgrow() functions.

Usage

```
MSPE(object, y_true, pos)
```

Arguments

object	A gpdpgrow or gmrfdpgrow object.
y_true	An N x T numeric matrix of test set values.
pos	An N x T matrix with all entries either 0 or 1, where a 1 indexes a missing entry or test point in y_true.

Value

A list object containing various MSPE fit statistics that measure the accuracy of of predicting the values in y_true indexed by pos.

SSE	Sum of squared errors based on *full* N x T, y_true - y_hat.
MSE	Mean squared error computed from SSE.
RMSE	Square root of MSE.
SSPE	Sum of squared prediction error based on missing values.
MSPE	Mean squared prediction error based on missing values.
nMSPE	Mean squared prediction error based on missing values that is normalized by the variance of the test observations to produce a value in [0,1].
RMSPE	Square root of MSPE.

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow, gmrfdpgrow

Examples

```
{
library(growfunctions)

## load the monthly employment count data for a collection of
## U.S. states from the Current
## Population Survey (cps)
data(cps)
## subselect the columns of N x T, y, associated with
## the years 2009 - 2013
## to examine the state level employment
## levels during the "great recession"
y_short   <- cps$y[,(cps$yr_label %in% c(2009:2013))]

## dimensions
T         <- ncol(y_short)  ## time points per unit
N         <- nrow(y_short)  ## number of units

## Demonstrate estimation of intermittent missing data
```

```
## from posterior predictive distribution by randomly
## selecting 10 percent of entries in y_short and
## setting them to NA.

## randomly assign missing positions in y.
## assume every unit has equal number of
## missing positions
## randomly select number of missing
## observations for each unit
m_factor  <- .1
M         <- floor(m_factor*N*T)
m_vec     <- rep(floor(M/N),N)
if( sum(m_vec) < M )
{
    m_left              <- M - sum(m_vec)
    pos_i               <- sample(1:N, m_left,
                                  replace = FALSE)
     m_vec[pos_i]        <- m_vec[pos_i] + 1
} # end conditional statement on whether all
  # missing cells allocated
  # randomly select missing
  # positions for each unit
pos       <- matrix(0,N,T)
for( i in 1:N )
{
    sel_ij              <- sample(3:(T-3), m_vec[i],
                           replace = FALSE)
                           ## avoid edge effects
    pos[i,sel_ij]       <- 1
}

## configure N x T matrix, y_obs, with 10 percent missing
## observations (filled with NA)
## to use for sampling.  Entries in y_short
## that are set to missing (NA) are
## determined by entries of "1" in the
## N x T matrix, pos.
y_obs               <- y_short
y_obs[pos == 1]     <- NA

## Conduct dependent GP model estimation under
## missing observations in y_obs.
## We illustrate the ability to have multiple
## function or covariance terms
## by seting gp_cov = c("se","sn"), which indicates
## the first term is a
## squared exponential ("se") trend covariance term
## and the "sn" is a seasonality
## term.  The setting, sn_order = 4, indicates the
## length scale of the seasonality
## term is 4 month.  The season term is actually
## "quasi" seasonal, in that the
## seasonal covariance kernel is multiplied by a
```

```
## squared exponential, which allows
## the pattern of seasonality to evolve over time.
res_gp_2              <- gpdpgrow(y = y_obs,
                                  gp_cov=c("se","sn"),
                                  sn_order = 4,
                                  n.iter = 10,
                                  n.burn = 4,
                                  n.thin = 1,
                                  n.tune = 0)

## 2 plots of estimated functions: 1. faceted by cluster and fit;
## 2.  data for experimental units.
## for a group of randomly-selected functions
fit_plots_gp_2        <- cluster_plot( object = res_gp_2,
                                       units_name = "state",
                                       units_label = cps$st,
                                       single_unit = TRUE,
                                       credible = TRUE )

## Compute out-of-sample fit statistic, normalized mean-square
## prediction error (MSPE)
## The normalized MSPE will take the predicted values
## for the entries in y_short purposefully
## set to NA and will subtract them from the known true
## values in y_short (and square them).
## This MSE on predicted (test) data is then
## divided by the variance of the test observations
## to output something akin to a percent error.
( nmspe_gp  <- MSPE(res_gp_2, y_short, pos)$nMSPE )
}
```

`plot_cluster` *Plot estimated functions, facetted by cluster numbers, for a known clustering*

Description

An internal function of the `growfunctions` package.

Usage

```
plot_cluster(y, H, sort = FALSE, sample_rate = 0.05, y.axis.label = NULL,
  smoother = TRUE, fade = 0.2, cluster_order = NULL, plot_render = TRUE)
```

Arguments

`y`	An `N x T` matrix for a collection of functions.
`H`	An `N x 1` with entries in `1,...,M` of cluster assignments for the `N` units of `y` under a known clustering.

sort	An optional boolean input on whether to sort the cluster-indexed plot panels of function by size of cluster. Defaults sort = FALSE.
sample_rate	An optional numeric value in (0,1] indicating percent of functions to randomly sample within each cluster to address over-plotting. Defaults to 1.
y.axis.label	An optional text label for y-axis. Defaults to "function values".
smoother	An optional scalar boolean input indicating whether to co-plot a smoother line through the functions in each cluster.
fade	An optional numeric input in (0,1) indicating the degree of fading to apply to the plots of functions in each cluster-indexed panel. Defaults to fade = 0.2.
cluster_order	An optional numeric vector of length M, the number of clusters, indicating the order, from-to-right, for plotting the cluster-indexed panels.
plot_render	An optional boolean input indicating whether to render the plot. Defaults to plot_render = TRUE.

Value

A list object containing the plot of estimated functions, faceted by cluster, and the associated data.frame object.

p.basis	A ggplot2 plot object
map	A data.frame object listing the unit and associated cluster membership.

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow, gmrfdpgrow

`predict_functions` *Use the model-estimated covariance parameters from gpdpgrow() or gmrdpgrow to predict the function at future time points.*

Description

This is the generic predict_functions method. See the following functions for the details about different data structures:

Usage

```
predict_functions(object, J, ...)
```

Arguments

object	Object of class gpdpgrow orgmrfdpgrow().
J	Scalar denoting number of draws to take from posterior predictive for each unit. Defaults to J = 500.
...	further arguments passed to or from other methods.

Details

- predict_functions.gpdpgrow for objects of class gpdpgrow
- predict_functions.gmrfdpgrow for objects of class gmrfdpgrow

---

predict_functions.gmrfdpgrow

Use the model-estimated iGMRF precision parameters from gmrfdpgrow() to predict the iGMRF function at future time points. Inputs the gmrfdpgrow *object of estimated parameters.*

Description

A companion function to gmrfdpgrow

Usage

```
## S3 method for class 'gmrfdpgrow'
predict_functions(object, J = 500, T_test, ...)
```

Arguments

object	Object of class gmrfdpgrow returned from model run of gmrfdpgrow()
J	Scalar denoting number of draws to take from posterior predictive for each unit. Defaults to J = 500.
T_test	The number of equally-spaced time points to predict the iGMRF functions ahead of of the functions estimated at T_train time points.
...	further arguments passed to or from other methods.

Value

out A list object containing containing two matrices; the first is a P x (N*T) matrix of predicted function values for each of P sampled iterations. N is slow index and denotes the number of experimental units. The second matrix is an N x T average over the P sampled draws, composed in Rao-Blackwellized fashion.

Note

Intended as a companion function for gmrfdpgrow for prediction

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gmrfdpgrow

Examples

```
## Not run:
library(growfunctions)
data(cps)
y_short   <- cps$y[,(cps$yr_label %in% c(2010:2013))]
t_train   <- ncol(y_short)
N         <- nrow(y_short)
t_test    <- 4

## Model Runs

res_gmrf            = gmrfdpgrow(y = y_short,
                                 q_order = c(2,4),
                                 q_type = c("tr","sn"),
                                 n.iter = 100,
                                 n.burn = 50,
                                 n.thin = 1)
## Prediction Model Runs
T_test             <- 4

pred_gmrf          <- predict_functions( object = res_gmrf,
                                         J = 1000,
                                         T_test = T_test )

## plot estimated and predicted functions
plot_gmrf       <- predict_plot(object = pred_gmrf,
                                units_label = cps$st,
                                single_unit = TRUE,
                                credible = FALSE)

## End(Not run)
```

---

predict_functions.gpdpgrow

Use the model-estimated GP covariance parameters from gpdpgrow() to predict the GP function at future time points. Inputs the gpdpgrow *object of estimated parameters.*

Description

A companion function to gpdpgrow

Usage

```
## S3 method for class 'gpdpgrow'
predict_functions(object, J = 500, test_times,
  time_points = NULL, sn_order = NULL, ...)
```

Arguments

object	Object of class gpdpgrow returned from model run of gpdpgrow()
J	Scalar denoting number of draws to take from posterior predictive for each unit. Defaults to J = 500.
test_times	A numeric vector holding test times at which to predict GP function values Will use the estimated covariance parameters from the training data to predict functions at the test_times for the N observation units.
time_points	Inputs a vector of common time points at which the collections of functions were observed (with the possibility of intermittent missingness). The length of time_points should be equal to the number of columns in the data matrix, y. Defaults to time_points = 1:ncol(y).
sn_order	An integer vector of length, L_s, equal to the number of seasonal terms. Conveys the order of the seasonality for each term on the scale of T; for example, if T is dimensioned in months, and one wishes to model quarterly seasonality, then the applicable seasonality term would be of order 3.
...	further arguments passed to or from other methods.

Value

out A list object containing containing two matrices; the first is a K x (N*T) matrix of predicted function values for each of K sampled iterations. N is slow index and denotes the number of experimental units. The second matrix is an N x T average over the K sampled draws, composed in Rao-Blackwellized fashion.

Note

Intended as a companion function for gpdpgrow for prediction

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow

Examples

```
## Not run:
library(growfunctions)
data(cps)
y_short   <- cps$y[,(cps$yr_label %in% c(2010:2013))]
```

```
t_train   <- ncol(y_short)
N         <- nrow(y_short)
t_test    <- 4

## Model Runs
res_gp              = gpdpgrow(y = y_short
                               n.iter = 50,
                               n.burn = 25,
                               n.thin = 1,
                               n.tune = 0)

## Prediction Model Runs
T_test             <- 4
T_yshort           <- ncol(y_short)
pred_gp            <- predict_functions( object = res_gp,
                        test_times = (T_yshort+1):(T_yshort+T_test) )

## plot estimated and predicted functions
plot_gp         <- predict_plot(object = pred_gp,
                                units_label = cps$st,
                                single_unit = FALSE,
                                credible = TRUE)

## End(Not run)
```

`predict_plot` *Plot estimated functions both at estimated and predicted time points with 95% credible intervals.*

Description

Uses as input the output object from the gpdpgrow.predict() and gmrfdpgrow.predict() methods.

Usage

```
predict_plot(object = NULL, units_label = NULL, type_label = c("fitted",
  "predicted"), time_points = NULL, date_label = NULL,
  x.axis.label = NULL, y.axis.label = NULL, single_unit = FALSE,
  credible = TRUE)
```

Arguments

`object`	A `gpdpgrow.predict` or `gmrfdpgrow.predict` object, obtained from `predict_functions(object,...`
`units_label`	A vector of labels to apply to experimental units with length equal to the number of unique units. Defaults to sequential numeric values as input with data, `y`.
`type_label`	A character vector assigning a "fitted" or "predicted label for the `time_points` input. Defaults to `type_label = c("fitted","predicted")`.

time_points	A list input of length 2 with each entry containing a numeric vector of times - one for the observed times for the set of "fitted" functions and the other denotes time values at which "predicted" values were rendered for the functions. This input variable only applies to gpdpgrow objects and not gmrfdpgrow objects because the latter covariance structure is based on adjacency for equally-spaced time points. Defaults to 1:T_train for the list entry pointed to "fitted" and (T_train+1):(T_train + T_test) for the list entry pointed to "predicted".
date_label	A vector of Date values for labeling the x-axis tick marks. Defaults to 1:T .
x.axis.label	Text label for x-axis. Defaults to "time".
y.axis.label	Text label for y-axis. Defaults to "function values".
single_unit	A scalar boolean indicating whether to plot the fitted vs data curve for only a single experimental units (versus a random sample of 6). Defaults to FALSE.
credible	A scalar boolean indicating whether to plot 95 percent credible intervals for estimated functions, bb, when plotting fitted functions versus data. Defaults to credible = TRUE .

Value

A list object containing the plot of estimated functions, faceted by cluster, and the associated data.frame object.

p.cluster	A ggplot2 plot object
dat.cluster	A data.frame object used to generate p.cluster.

Author(s)

Terrance Savitsky <tds151@gmail.com>

See Also

gpdpgrow, gmrfdpgrow

Examples

```
## Not run:
library(growfunctions)
data(cps)
y_short             <- cps$y[,(cps$yr_label %in% c(2008:2013))]
t_train             <- ncol(y_short)
N                   <- nrow(y_short)
t_test              <- 4

## Model Runs

res_gmrf            <- gmrfdpgrow(y = y_short,
                                  q_order = c(2,4),
                                  q_type = c("tr","sn"),
                                  n.iter = 40,
                                  n.burn = 20,
```

```
                                n.thin = 1)

res_gp              <- gpdpgrow(y = y_short
                                n.iter = 10,
                                n.burn = 4,
                                n.thin = 1,
                                n.tune = 0)

## Prediction Model Runs
T_test              <- 4

pred_gmrf           <- predict_functions( object = res_gmrf,
                                          J = 1000,
                                          T_test = T_test )

T_yshort            <- ncol(y_short)
pred_gp             <- predict_functions( object = res_gp,
                      test_times = (T_yshort+1):(T_yshort+T_test) )

## plot estimated and predicted functions
plot_gmrf       <- predict_plot(object = pred_gmrf,
                                units_label = cps$st,
                                single_unit = TRUE,
                                credible = FALSE)

plot_gp         <- predict_plot(object = pred_gp,
                                units_label = cps$st,
                                single_unit = FALSE,
                                credible = TRUE)

## End(Not run)
```

`samples` *Produce samples of MCMC output*

Description

provides posterior sampled values for every model parameter of a `gpdpgrow` object

Usage

```
samples(object, ...)
```

Arguments

`object`	A `gpdpgrow` object
`...`	Ignored

`samples.gmrfdpgrow` *Produce samples of MCMC output*

Description

provides posterior sampled values for every model parameter of a `gmrfdpgrow` object

Usage

```
## S3 method for class 'gmrfdpgrow'
samples(object, ...)
```

Arguments

`object`	A `gmrfdpgrow` object
`...`	Ignored

Index