Package ‘plgp’

February 20, 2015

Type Package

Title Particle Learning of Gaussian Processes

Version 1.1-7

Date 2014-11-27

Author Robert B. Gramacy <rbgramacy@chicagobooth.edu>

Maintainer Robert B. Gramacy <rbgramacy@chicagobooth.edu>

Description Sequential Monte Carlo inference for fully Bayesian Gaussian process (GP) regression and classification models by particle learning (PL). The sequential nature of inference and the active learning (AL) hooks provided facilitate thrifty sequential design (by entropy) and optimization (by improvement) for classification and regression models, respectively. This package essentially provides a generic PL interface, and functions (arguments to the interface) which implement the GP models and AL heuristics. Functions for a special, linked, regression/classification GP model and an integrated expected conditional improvement (IECI) statistic is provides for optimization in the presence of unknown constraints. Separable and isotropic Gaussian, and single-index correlation functions are supported. See the examples section of ?plgp and demo(package="plgp") for an index of demos

Depends R (>= 2.4), mvtnorm, tgp

Suggests ellipse, splancs, akima

License LGPL

URL http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-12-02 00:14:32
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>plgp-package</td>
<td>2</td>
</tr>
<tr>
<td>addpall.GP</td>
<td>3</td>
</tr>
<tr>
<td>data.GP</td>
<td>4</td>
</tr>
<tr>
<td>draw.GP</td>
<td>6</td>
</tr>
<tr>
<td>exp2d.C</td>
<td>8</td>
</tr>
<tr>
<td>init.GP</td>
<td>9</td>
</tr>
<tr>
<td>lpredprob.GP</td>
<td>10</td>
</tr>
<tr>
<td>papply</td>
<td>11</td>
</tr>
<tr>
<td>params.GP</td>
<td>12</td>
</tr>
<tr>
<td>PL</td>
<td>13</td>
</tr>
<tr>
<td>pred.GP</td>
<td>16</td>
</tr>
<tr>
<td>prior.GP</td>
<td>18</td>
</tr>
<tr>
<td>propagate.GP</td>
<td>19</td>
</tr>
<tr>
<td>rectsclae</td>
<td>20</td>
</tr>
</tbody>
</table>

### Description

Sequential Monte Carlo inference for fully Bayesian Gaussian process (GP) regression and classification models by particle learning (PL). The sequential nature of inference and the active learning (AL) hooks provided facilitate thrifty sequential design (by entropy) and optimization (by improvement) for classification and regression models, respectively. This package essentially provides a generic PL interface, and functions (arguments to the interface) which implement the GP models and AL heuristics. Functions for a special, linked, regression/classification GP model and an integrated expected conditional improvement (IECI) statistic is provides for optimization in the presence of unknown constraints. Separable and isotropic Gaussian, and single-index correlation functions are supported. See the examples section of `?plgp` and `demo(package="plgp")` for an index of demos.

### Details

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

### Author(s)

Robert B. Gramacy <rbgramacy@chicagobooth.edu>
addpall.GP

References

arXiv:0909.5262

9, J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith and M.
West (Eds.); Oxford University Press


http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

See Also

PL, tgp

```r
addpall.GP  Add data to pall
```

Description

Add sufficient data common to all particles to the global `pall` variable, a mnemonic for “particles-
all”, for Gaussian process (GP) regression, classification, or combined unknown constraint models

Usage

```r
addpall.GP(Z)
addpall.CGP(Z)
addpall.ConstGP(Z)
```

Arguments

- `Z` new observation(s) (usually the next one in “time”) to add to the `pall` global variable

Details

All three functions add new `Z$x` to `pall$x`; `addpall.GP` also adds `Z$y` to `pall$Y`, `addpall.CGP` also adds `Z$c` to `pall$Y`, and `addpall.ConstGP` does both

Value

nothing is returned, but global variables are modified

Author(s)

Robert B. Gramacy, <rbgramacy@chicagobooth.edu>
References


http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

See Also

PL

Examples

```r
## See the demos via demo(package="plgp") and the examples
## section of ?plgp
```

data.GP

Supply GP data to PL

Description

Functions to supply data to PL for Gaussian process (GP) regression, classification, or combined unknown constraint models

Usage

```r
data.GP(begin, end = NULL, X, Y)
data.GP.improv(begin, end = NULL, f, rect, prior,
    adapt = ei.adapt, cands = 40,
    save = TRUE, oracle = TRUE, verb = 2,
    interp = interp.loess)
data.CGP(begin, end = NULL, X, C)
data.CGP.adapt(begin, end = NULL, f, rect, prior,
    cands = 40, verb = 2, interp=interp.loess)
data.ConstGP(begin, end = NULL, X, Y, C)
data.ConstGP.improv(begin, end = NULL, f, rect, prior,
    adapt = ieci.const.adapt, cands = 40,
    save = TRUE, oracle = TRUE, verb = 2,
    interp = interp.loess)
```
Arguments

begin  positive integer starting time for data to be returned
end   positive integer (end >= begin) ending time for data being returned; may be
       NULL if only data at time begin is needed
X     data.frame with at least end rows containing covariates
Y     vector of length at least end containing real-valued responses
C     vector of length at least end containing class labels
f     function returning a responses when called as f(X) for matrix X; for data.GP.improv
       the responses must be real-valued returned as a vector; for data.CGP.adapt
       they must be class labels returned as a vector; for data.ConstGP.improv they
       must be pairs of real-valued and in {0,1} (1 indicates constraint violation), re-
       turned as a 2-column data.frame
rect  bounding rectangle for the inputs X to f(X) with two columns and rows equalling
       nrow(X)
prior prior parameters passed from PL generated by one of the prior functions, e.g.,
       prior.GP
adapt function that evaluates a sequential design criterion on some candidate locations;
       the default ei.adapt EI about the minimum; ieci.adapt providing IECI is
       another possibility, which is hard coded into data.ConstGP.adapt
cands number of Latin Hypercube candidate locations used to choose the next adap-
       tively sampled input design point
save   scalar logical indicating if the improvment information for chosen candidate
       should be saved in the psave global variable
oracle scalar logical indicating if the candidates should be augmented with the point
       found to maximize the predictive surface (with a search starting at the most
       recently chosen input)
verb   verbosity level for printing the progress of improv and other adaptive sampling
       calculations
interp function for smoothing of 2-d image plots. The default comes from interp.loess,
       but what works best is interp which requires the akima package

Details

These functions provide data to PL for Gaussian progress regression and classification methods
in a variety of ways. The simplest, data.GP and data.CGP supply pre-recorded regression and
classification data stored in data frames and vectors; data.ConstGP is a hybrid that does joint
regression and classification. The other functions provide data by active learning/sequential design:
The data.GP.improv function uses expected improvement (EI); data.CGP.improv uses predictive
entropy; data.ConstGP.improv uses integrated expected conditional improvement (IECI). In these
cases, once the x-location(s) is/are chosen, the function f is used to provide the response(s)

Value

The output are vectors or data.frames.
Author(s)
Robert B. Gramacy, <rbgramacy@chicagobooth.edu>

References
arXiv:0909.5262
9, J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith and M.
West (Eds.); Oxford University Press
http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

See Also
PL

Examples
### See the demos via demo(package="plgp") and the examples
### section of ?plgp

---

**draw.GP**

*Metropolis-Hastings draw for GP parameters*

### Description
Functions for using Metropolis-Hastings (MH) to evolve a particle according to the posterior
distribution given by a Gaussian process (GP) for regression, classification, or combined unknown
constraint model

### Usage
```
draw.GP(Zt, prior, l = 3, h = 4, thin = 10, Y = NULL)
draw.CGP(Zt, prior, l = 3, h = 4, thin = 10)
draw.ConstGP(Zt, prior, l = 3, h = 4, thin = 10)
```

### Arguments
- **Zt**
  the particle describing model parameters and sufficient statistics that determines
  the predictive distribution
- **prior**
  prior parameters passed from PL generated by one of the prior functions, e.g.,
  prior.GP
- **l**
  positive uniform random walk parameter; for old parameter pold, a new param-
  eter is proposed as p = runif(1, p*l/h, p*h/l). Such proposals are then
  accepted (or rejected) via the MH acceptance ratio
**draw.GP**

- `h`: positive uniform random walk parameter; see above
- `thin`: thinning level in the MCMC; describes the number of MH rounds executed before the value is saved as a sample from the (marginal) posterior distribution
- `Y`: not for external use; used internally by CGP and ConstGP internal routines

**Details**

These functions are used in two important places in plgp. At the user level, they can be used to initialize the particles at time `start`; see PL and the demos. Internally, they are used in the PL propagate step, e.g., `propagate.GP`

draw.ConstGP is a combination of the draw.GP and draw.CGP methods, which are for regression and classification GPs, respectively

**Value**

These functions return an updated particle `Zt`

**Author(s)**

Robert B. Gramacy, <rbgramacy@chicagobooth.edu>

**References**


http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

**See Also**

`init.GP,propagate.GP,PL`

**Examples**

```r
## See the demos via demo(package="plgp") and the examples
## section of ?plgp
```
exp2d.C  2-d Exponential Hessian Data

Description
Generates 2-d classification data with two or three class labels, based on the Hessian data from a 2-d real-valued response

Usage
exp2d.C(x, threed = TRUE)

Arguments
X  a matrix or data.frame describing the design at which the response categories are desired
threed  a scalar logical indicating if the two or three-class version of the class labels should be returned.

Details
The underlying real-valued response is governed by

\[ Z(X) = x_1 \exp(x_1^2 - x_2^2). \]

Two class labels are generated by inspecting the sign of the sum of the eigenvalues of the Hessian (Broderick & Gramacy, 2010). This generates the first (-) and second (+) classes in a three-class function. A third class label (the default) may created from the first one where \( X[:,1] > 0 \) (Gramacy & Polson, 2011)

Value
A vector of class labels of length nrow(X) is returned

Author(s)
Robert B. Gramacy, <rbgramacy@chicagobooth.edu>

References

http://faculty.chicagobooth.edu/robert.gramacy/plgp.html
Examples

```r
## The following demos use this data
## Not run:
## Illustrates classification GPs on a simple 2-d exponential
data generating mechanism
demo("plcgp_exp", ask=FALSE)

## Illustrates active learning via entropy with classification
## GPs on a simple 2-d exponential data generating mechanism
demo("plcgp_exp_entropy", ask=FALSE)

## End(Not run)
```

---

### init.GP

**Initialize particles for GPs**

---

**Description**

Functions for initializing particles for Gaussian process (GP) regression, classification, or combined unknown constraint models

**Usage**

```r
init.GP(prior, d = NULL, g = NULL, y = NULL)
init.CGP(prior, d = NULL, g = NULL)
init.ConstGP(prior)
```

**Arguments**

- **prior**
  - prior parameters passed from `pl` generated by one of the prior functions, e.g., `prior.GP`
- **d**
  - initial range (or length-scale) parameter(s) for the GP correlation function(s)
- **g**
  - initial nugget parameter for the GP correlation
- **y**
  - data used to update GP sufficient information in the case of `init.GP`; if NULL then `pall$Y` is used

**Value**

Returns a particle for internal use in the `pl` method

**Author(s)**

Robert B. Gramacy, <rbgramacy@chicagobooth.edu>
lpredprob.GP

References
http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

See Also
PL, draw.GP

Examples

## See the demos via demo(package="plgp") and the examples
## section of ?plgp

lpredprob.GP Log-Predictive Probability Calculation for GPs

Description
Log-predictive probability calculation for Gaussian process (GP) regression, classification, or combined unknown constraint models; primarily to be used particle learning (PL) re-sample step

Usage

lpredprob.GP(z, Zt, prior)
lpredprob.CGp(z, Zt, prior)
lpredprob.ConstGP(z, Zt, prior)

Arguments

z new observation whose (log) predictive probability is to be calculated given the particle Zt
Zt the particle describing model parameters and sufficient statistics that determines the predictive distribution
prior prior parameters passed from PL generated by one of the prior functions, e.g., prior.GP

Details
This is the workhorse of the PL re-sample step. For each new observation (in sequence), the PL function calls lpredprob and these values determine the weights used in the sample function to obtain the new particle set, which is then propagated, e.g., using propagate.GP
The lpredprob.ConstGP is essentially the combination (product) of lpredprob.GP and lpredprob.CGp for regression and classification GP models, respectively
papply

Value

Returns a real-valued scalar - the log predictive probability

Author(s)

Robert B. Gramacy, <rbgramacy@chicagobooth.edu>

References

arXiv:0909.5262

9, J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith and M.
West (Eds.); Oxford University Press

http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

See Also

PL, propagate.GP

Examples

```r
## See the demos via demo(package="plgp") and the examples
## section of ?plgp

papply

Extending apply to particles

Description

Applies a user-specified function to each particle contained in the global variables peach and pall,
collecting the output in a data.frame

Usage

papply(fun, verb = 1, pre = "", ...)
Extract parameters from GP particles

Extract parameters from particles for Gaussian process (GP) regression, classification, or combined unknown constraint models

Usage

params.GP()
params.CGP()
params.ConstGP()
Details

Collects the parameters from each of the particles (contained in the global variable `peach`) into a `data.frame` that can be used for quick summary and visualization, e.g., via `hist`. These functions are also called to make progress visualizations in `PL`.

Value

returns a `data.frame` containing summaries for each parameter in its columns.

Author(s)

Robert B. Gramacy, <rbgramacy@chicagobooth.edu>

References


http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

See Also

`PL`, `lpredprob.GP`, `propagate.GP`, `init.GP`, `pred.GP`

Examples

```r
## See the demos via demo(package="plgp") and the examples
## section of ?plgp
```

---

**PL**  
*Particle Learning Skeleton Method*

Description

Implements the Particle Learning sequential Monte Carlo algorithm on the data sequence provided, using re-sample and propagate steps.

Usage

```r
PL(dstream, start, end, init, lpredprob, propagate, prior = NULL, 
addpall = NULL, params = NULL, save = NULL, P = 100, 
progress = 10, cont = FALSE, verb = 1)
```
Arguments

dstream
function generating the data stream; for examples see data.GP

start
a scalar integer specifying the starting “time”; the data entry/sample where PL will start

end
a scalar integer specifying the ending “time”; the data entry/sample where PL will stop

init
function used to initialize the particles at the start of PL; for examples see draw.GP

lpredprob
function used to calculate the predictive probability of an observation (usually the next one in “time”) given a particle. This is the primary function used in the PL re-sample step; for examples see lpredprob.GP

propagate
function used to propagate particles given an observation (usually the next one in “time”); for examples see propagate.GP

prior
function used to generate prior parameters that may be passed into the dstream, init, lpredprob and propagate functions as needed; for examples see prior.GP

addpall
an optional function that adds the new observation (usually the next one in “time”) to the pall variable in the pl.env environment (i.e., pl.env$pall), which stores the sufficient information shared by all particles; for examples see addpall.GP

params
an optional function called each progress rounds that collects parameters from the particles for summary and visualization; for examples see params.GP

save
an option function that is called every round to save some information about the particles

P
number of particles to use

progress
number of PL rounds after which to collect params and draws histograms; a non-positive value or params = NULL skips the progress meter

cont
if TRUE then PL will try to use the existing set of particles to “continue” where it left off; start and end should be specified appropriately when continuing

verb
if nonzero, then screen prints will indicate the proportion of PL updates finished so far; verb = 1 will cause PL to pause on progress drawings for inspection

Details

Uses the PL SMC algorithm via the functions provided. This function is just a skeleton framework. The hard work is in specifying the arguments/functions which execute the calculations needed in the re-sample and propagate steps.

PL and uses the variables stored in the pl.env environment: pall, containing sufficient information common to all particles, peach, containing sufficient information particular to each of the P particles, and psave containing any saved information. These variables may be accessed as pl.env$psave, for example.

Note that PL is designed to be fast for sequential updating (of GPs) when new data arrive. This facilitates efficient sequential design of experiments by active learning techniques, e.g., optimization by expected improvement and sequential exploration of classification label boundaries by the predictive entropy. PL is not optimized for static inference when all of the data arrive at once, in batch
Value

PL modifies the PL.env$speach variable, containing sufficient information particular to each (of the P) particles.

Author(s)

Robert B. Gramacy, <rbgramacy@chicagobooth.edu>

References


See Also

papply, draw.GP, data.GP, lpredprob.GP, propagate.GP, params.GP, pred.GP

Examples

```r
## See the demos via demo(package="plgp"); it is important to
## run them with the ask=FALSE argument so that the
## automatically generated plots may refresh automatically
## (without requiring the user to press RETURN)
## Not run:
## Illustrates regression GPs on a simple 1-d sinusoidal
## data generating mechanism
demo("plgp_sin1d", ask=FALSE)

## Illustrates classification GPs on a simple 2-d exponential
## data generating mechanism
demo("plcgp_exp", ask=FALSE)

## Illustrates classification GPs on Ripley's Cushings data
demo("plcgp_cush", ask=FALSE)

## Illustrates active learning via the expected improvement
## statistic on a simple 1-d data generating mechanism
demo("plgp_exp_ei", ask=FALSE)

## Illustrates active learning via entropy with classification
## GPs on a simple 2-d exponential data generating mechanism
demo("plcgp_exp_entropy", ask=FALSE)
```
Prediction for GPs

Description

Prediction on a per-particle basis for Gaussian process (GP) regression, classification, or combined unknown constraint models

Usage

```r
pred.GP(XX, Zt, prior, Y = NULL, quants = FALSE, Sigma = FALSE,
        sub = 1:Zt$t)
pred.CGP(XX, Zt, prior, mcreps = 100, cs = NULL)
pred.ConstGP(XX, Zt, prior, quants = TRUE)
```

Arguments

- **XX**: matrix or data.frame containing (a design of) predictive locations where `ncol(XX) = ncol(X)`, on which the data were trained and particle `Zt` thus obtained
- **Zt**: the particle describing model parameters and sufficient statistics that determines the predictive distribution
- **prior**: prior parameters passed from `PL` generated by one of the prior functions, e.g., `prior.GP`
- **Y**: not for external use; used internally by CGP and ConstGP internal routines
- **quants**: a scalar logical indicating if predictive quantiles should be desired
- **Sigma**: a scalar logical indicating if the full predictive variance-covariance matrix is desired; typically only used internally by CGP and ConstGP
pred.GP

sub not for external used; used internally by CGP and ConstGP internal routines
mcreps number of Monte Carlo iterations used in CGP prediction, integrating over the latent real-valued Y variables at the XX locations
cs indicates a class label at which the predictive probability is desired; the entire probability distribution over all class labels will be provided if not specified

Details
For pred.GP the predictive mean (and quantiles if quants = TRUE is provided. For pred.CGP the predictive distribution over the class labels is provided, unless only one class (cs) is desired. pred.ConstGP is a combination of the pred.GP and pred.CGP methods
It is suggested that this function is used in as an argument to papply to obtain many predictions - one for each particle in a cloud - which are combined into a data.frame
Some of the function arguments aren’t meant to be specified by the user, but are rather there to facilitate usage as a subroutine inside other PL functions, such as lpredprob.GP and others

Value
A single-row data.frame is returned with the desired predictive; these rows are automatically combined when used with papply

Author(s)
Robert B. Gramacy, <rbgramacy@chicagobooth.edu>

References
http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

See Also
papply, PL, lpredprob.GP

Examples
## See the demos via demo(package="plgp") and the examples
## section of ?plgp
Generate priors for Gaussian process (GP) regression, classification, or combined unknown constraint models

Usage

prior.GP(m, cov = c("isotropic", "separable", "sim"))
prior.CGP(m, cov = c("isotropic", "separable", "sim"))
prior.ConstGP(m, cov.GP = c("isotropic", "separable", "sim"),
              cov.CGP = cov.GP)

Arguments

m positive scalar integer specifying the dimensionality of the input space

cov whether to use an "isotropic" or "separable" power exponential correlation function with power 2 – nugget included; a single index model ("sim") capability is provided as “beta” functionality; applies to both regression and classification GPs

cov.GP specifies the covariance for the real-valued response in the combined unknown constraint GP model

cov.CGP specifies the covariance for the categorical response in the combined unknown constraint GP model

Details

These function generate a default prior object in the correct format for use with the other PL routines, e.g., init.GP and pred.GP. The object returned may be modified as necessary.

The prior.ConstGP is essentially the combination of prior.GP and prior.CGP for regression and classification GP models, respectively

Value

a valid prior object for the appropriate GP model;

By making the output $d_{rate}$ and/or $d_{rate}$ values negative causes the corresponding lengthscale $d$ parameter(s) and nugget $d$ parameter to be fixed at the reciprocal of their absolute values, respectively. This effectively turns off inference for these values, and allows one to study the GP predictive distribution as a function of fixed values. When both are fixed it is sensible to use only one particle ($P=1$, as an argument to PL)

Author(s)

Robert B. Gramacy, <rbgramacy@chicagobooth.edu>
References


http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

See Also

PL, lpredprob.GP, propagate.GP, init.GP, pred.GP

Examples

```r
## See the demos via demo(package="plgp") and the examples
## section of ?plgp
```

```
propagate.GP
```

**PL propagate rule for GPs**

Description

Incorporation of a new data point for Gaussian process (GP) regression, classification, or combined unknown constraint models; primarily to be used particle learning (PL) propagate step

Usage

```r
propagate.GP(z, Zt, prior)
propagate.CGP(z, Zt, prior)
propagate.ConstGP(z, Zt, prior)
```

Arguments

- `z` new observation whose to be incorporate into the particle `Zt`
- `Zt` the particle describing model parameters and sufficient statistics that the new data is being incorporated into
- `prior` prior parameters passed from PL generated by one of the prior functions, e.g., `prior.GP`

Details

This is the workhorse of the PL propagate step. After re-sampling the particles, PL calls propagate on each of the particles to obtain the set used in the next round/time-step

The `propagate.ConstGP` is essentially the combination of `propagate.GP` and `propagate.CGP` for regression and classification GP models, respectively
Value

These functions return a new particle with the new observation incorporated

Author(s)

Robert B. Gramacy, <rbgramacy@chicagobooth.edu>

References

arXiv:0909.5262
9, J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith and M.
West (Eds.); Oxford University Press
http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

See Also

PL, lpredprob.GP

Examples

```r
## See the demos via demo(package="plgp") and the examples
## section of ?plgp
```

rectscale

Un/Scale data in a bounding rectangle

Description

Scale data lying in an arbitrary rectangle to lie in the unit rectangle, and back again

Usage

rectscale(X, rect)
rectunscale(X, rect)

Arguments

X a matrix or data.frame of real-valued covariates
rect a matrix describing a bounding rectangle for X with 2 columns and ncol(X)
rows

Value

a matrix or data.frame with the same dimensions as X scaled or un-scaled as appropriate
rectscale

Author(s)
Robert B. Gramacy, <rbgramacy@chicagobooth.edu>

References
http://faculty.chicagobooth.edu/robert.gramacy/plgp.html

Examples
X <- matrix(runif(10, 1, 3), ncol=2)
rect <- rbind(c(1,3), c(1,3))
Xs <- rectscale(X, rect)
rectunscale(Xs, rect)
Index

*Topic **classif**
  draw.GP, 6
  init.GP, 9
  params.GP, 12
  pred.GP, 16
  prior.GP, 18
  propagate.GP, 19

*Topic **datagen**
  data.GP, 4
  exp2d.C, 8

*Topic **iterations**
  PL, 13

*Topic **iteration**
  papply, 11

*Topic **methods**
  data.GP, 4
  draw.GP, 6
  init.GP, 9
  lpredprob.GP, 10
  papply, 11
  params.GP, 12
  PL, 13
  pred.GP, 16
  prior.GP, 18
  propagate.GP, 19

*Topic **models**
  draw.GP, 6
  init.GP, 9
  lpredprob.GP, 10
  params.GP, 12
  pred.GP, 16
  prior.GP, 18
  propagate.GP, 19

*Topic **package**
  plgp-package, 2

*Topic **regression**
  draw.GP, 6
  init.GP, 9
  lpredprob.GP, 10
  params.GP, 12
  pred.GP, 16
  prior.GP, 18
  propagate.GP, 19

*Topic **utilities**
  addpall.GP, 3
  rectscale, 20
  addpall.CGP (addpall.GP), 3
  addpall.ConstGP (addpall.GP), 3
  addpall.GP, 3, 14
  apply, 12
  data.CGP (data.GP), 4
  data.ConstGP (data.GP), 4
  data.frame, 11, 13, 17
  data.GP, 4, 14, 15
  draw.CGP (draw.GP), 6
  draw.ConstGP (draw.GP), 6
  draw.GP, 6, 10, 14, 15
  exp2d.C, 8
  hist, 13
  init.CGP (init.GP), 9
  init.ConstGP (init.GP), 9
  init.GP, 7, 9, 13, 18, 19
  interp, 5
  interp.loess, 5
  lpredprob.CGP, 10
  lpredprob.CGP (lpredprob.GP), 10
  lpredprob.ConstGP, 10
  lpredprob.ConstGP (lpredprob.GP), 10
  lpredprob.GP, 10, 10, 13–15, 17, 19, 20
  papply, 11, 15, 17
  params.CGP (params.GP), 12
  params.ConstGP (params.GP), 12
  params.GP, 12, 15
INDEX

PL, 3–7, 9–13, 13, 16–20
plgp (PL), 13
plgp-package, 2
pred.CGP (pred.GP), 16
pred.ConstGP (pred.GP), 16
pred.GP, 12, 13, 15, 16, 18, 19
prior.CGP, 18
prior.CGP (prior.GP), 18
prior.ConstGP, 18
prior.ConstGP (prior.GP), 18
prior.GP, 5, 6, 9, 10, 14, 16, 18, 18, 19
propagate.CGP, 19
propagate.CGP (propagate.GP), 19
propagate.ConstGP, 19
propagate.ConstGP (propagate.GP), 19
propagate.GP, 7, 10, 11, 13–15, 19, 19
rectscale, 20
rectunscale (rectscale), 20

sample, 10
summary, 13