Package 'mlegp'

February 15, 2013

Type Package

Title Maximum Likelihood Estimates of Gaussian Processes

Version 3.1.3

Date 2012-12-10

Author Garrett M. Dancik

Maintainer Garrett M. Dancik <garrett.dancik@gmail.com>

Suggests snowfall

Description Maximum likelihood Gaussian process modeling for
univariate and multi-dimensional outputs with diagnostic plots.
Contact the maintainer for a package version that implements sensitivity analysis functionality.

License GPL (>= 2)

URL

Repository CRAN

Date/Publication 2012-12-20 19:11:33

NeedsCompilation yes

R topics documented:

mlegp-package 2
is.gp 3
mlegp 4
mlegp-nugget-related 8
plot.gp 9
plot.gp.list 10
plotObservedEffects 11
predict.gp 12
print.gp 14

print.gp.list 15
summary.gp 16
summary.gp.list 17
uniqueSummary 18

Index **19**

`mlegp-package` *mlegp package*

Description

Maximum likelihood Gaussian process modeling for univariate and multi-dimensional outputs with diagnostic plots and sensitivity analysis.

Details

Package:	mlegp
Type:	Package
Version:	2.0
Date:	2007-12-05
License:	Gnu General Public License (Version 3)

This package obtains maximum likelihood estimates of Gaussian processes (GPs) for univariate and multi-dimensional outputs, for Gaussian processes with product exponential correlation structure; a constant or linear regression mean function; no nugget term, constant nugget term, or a nugget matrix that can be specified up to a multiplicative constant. The latter provides some flexibility for using GPs to model heteroscedastic responses.

Multi-dimensional output can be modelled by fitting independent GPs to each output, or to the most important principle component weights following singular value decomposition of the output. Plotting of main effects for functional output is also implemented.

Contact the maintainer for a package version that implements sensitivity analysis including Functional Analysis of Variance (FANOVA) decomposition, plotting functions to obtain diagnostic plots, main effects, and two-way factor interactions.

For a complete list of functions, use 'library(help="mlegp")'.

Author(s)

Garrett M. Dancik < `garrett.dancik@gmail.com` >

References

Santner, T.J. Williams, B.J., Notz, W., 2003. The Design and Analysis of Computer Experiments (New York: Springer).

Schonlau, M. and Welch, W. 2006. Screening the Input Variables to a Computer Model Via Analysis of Variance and Visualization, in Screening: Methods for Experimentation in Industry, Drug Discovery, and Genetics. A Dean and S. Lewis, eds. (New York: Springer).

Heitmann, K., Higdon, D., Nakhleh, C., Habib, S., 2006. Cosmic Calibration. The Astrophysical Journal, 646, 2, L1-L4.

`is.gp` *Gaussian Process and Gaussian Process Lists*

Description

Test for a Gaussian process object or a Gaussian process list object

Usage

```
is.gp(x)
is.gp.list(x)
```

Arguments

`x`	object to be tested

Value

`is.gp` returns `TRUE` or `FALSE`, depending on whether its argument inherits the `gp` class or not

`is.gp.list` returns `TRUE` or `FALSE`, depending on whether its argument inherits the `gp.list` class or not

Author(s)

Garrett M. Dancik < `garrett.dancik@gmail.comu` >

Examples

```
## fit a single Gaussian process ##
x = -5:5; y1 = sin(x) + rnorm(length(x),sd=.1)
fit1 = mlegp(x, y1)

is.gp(fit1)       ## returns TRUE
is.gp.list(fit1)  ## returns FALSE
```

mlegp *mlegp: maximum likelihood estimation of Gaussian process parameters*

Description

Finds maximum likelihood estimates of Gaussian process parameters for a vector (or matrix) of one (or more) responses. For multiple responses, the user chooses between fitting independent Gaussian processes to the separate responses or fitting independent Gaussian processes to principle component weights obtained through singular value decomposition of the output. The latter is useful for functional output or data rich situations.

Usage

```
mlegp(X, Z, constantMean = 1, nugget = NULL, nugget.known = 0, min.nugget = 0, param.names = NULL,
PC.UD = NULL, PC.num = NULL, PC.percent = NULL,
simplex.ntries = 5, simplex.maxiter = 500, simplex.reltol = 1e-8,
BFGS.maxiter = 500, BFGS.tol = 0.01, BFGS.h = 1e-10, seed = 0, verbose = 1, parallel = FALSE)
```

Arguments

X	the design matrix
Z	vector or matrix of observations; corresponding to the rows of X
constantMean	a value of 1 indicates that each Gaussian process will have a constant mean; otherwise the mean function will be a linear regression in X, plus an intercept term
nugget	if nugget.known is 1, a fixed value to use for the nugget or a vector corresponding to the fixed diagonal nugget matrix; otherwise, either a positive initial value for the nugget term which will be estimated, or a vector corresponding to the diagonal nugget matrix up to a multiplicative constant. If NULL (the default), mlegp estimates a nugget term only if there are replicates in the design matrix, see details
nugget.known	1 if a plug-in estimate of the nugget will be used; 0 otherwise
min.nugget	minimum value of the nugget term; 0 by default
param.names	a vector of parameter names, corresponding to the columns of X; parameter names are 'p1', 'p2', ... by default
gp.names	a vector of GP names, corresponding to the GPs fit to each column of Z or each PC weight
PC.UD	the UD matrix if Z is a matrix of principle component weights; see mlegp-svd-functions
PC.num	the number of principle component weights to keep in the singular value decomposition of Z
PC.percent	if not NULL the number of principle component weights kept is the minimum number that accounts for PC.percent of the total variance of the matrix Z
simplex.ntries	the number of simplexes to run

`simplex.maxiter` maximum number of evaluations / simplex

`simplex.reltol` relative tolerance for simplex method, defaulting to 1e-16

`BFGS.maxiter` maximum number of iterations for BFGS method

`BFGS.tol` stopping condition for BFGS method is when norm(gradient) < BFGS.tol * max(1, norm(x)), where x is the parameter vector and norm is the Euclidian norm

`BFGS.h` derivatives are approximated as [f(x+BFGS.h) - f(x)] / BFGS.h)

`seed` the random number seed

`verbose` a value of '1' or '2' will result in status updates being printed; a value of '2' results in more information

`parallel` if TRUE will fit GPs in parallel to each column of Z, or each set of PC weights; See details

Details

This function calls the C function fitGPFromR which in turn calls fitGP (both in the file fit_gp.h) to fit each Gaussian process.

Separate Gaussian processes are fit to the observations in each column of `Z`. Maximum likelihood estimates for correlation and nugget parameters are found through numerical methods (i.e., the Nelder-Mead Simplex and the L-BFGS method), while maximum likelihood estimates of the mean regression parameters and overall variance are calculated in closed form (given the correlation and (scaled) nugget parameters). Multiple simplexes are run, and estimates from the best simplex are used as initial values to the gradient (L-BFGS) method.

Gaussian processes are fit to principle component weights by utilizing the singular value decomposition (SVD) of `Z`, Z = UDVprime. Columns of `Z` should correspond to a single k-dimensional observation (e.g., functional output of a computer model, evaluated at a particular input)

In the complete SVD, `Z` is k x m, and r = min(k,m), `U` is k x r, `D` is r x r, containing the singular values along the diagonal, and `Vprime` is r x m. The output `Z` is approximated by keeping l < r singular values, keeping a UD matrix of dimension k x l, and the `Vprime` matrix of dimension l x m. Each column of `Vprime` now contains l principle component weights, which can be used to reconstruct the functional output.

If nugget.known = 1, nugget = NULL, and replicate observations are present, the nugget will be fixed at its best linear unbiased estimate (a weighted average of sample variances). For each column of Z, a GP will be fit to a collection of sample means rather than all observations. This is the recommended approach as it is more accurate and computationally more efficient.

Parallel support is provided through the package `snowfall` which allows multiple GPs to be fit in parallel. The user must set up the cluster using `sfInit` and call `sfLibrary(mlegp)` to load the library onto the slave nodes. Note: GP fitting is not recommended when the number of observations are large (> 100), in which case sequential GP fitting is faster.

Value

an object of class `gp.list` if Z has more than 1 column, otherwise an object of class `gp`

Note

The random number seed is 0 by default, but should be randomly set by the user

In some situations, especially for noiseless data, it may be desirable to force a nugget term in order to make the variance-covariance matrix of the Gaussian process more stable; this can be done by setting the argument `min.nugget`.

If fitting multiple Gaussian processes, the arguments `min.nugget` and `nugget` apply to all Gaussian processes being fit.

In some cases, the variance-covariance matrix is stable in C but not stable in R. When this happens, this function will attempt to impose a minimum value for the nugget term, and this will be reported. However, the user is encouraged to refit the GP and manually setting the argument `min.nugget` in `mlegp`.

When fitting Gaussian processes to principle component weights, a minimum of two principle component weights must be used.

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

References

Santner, T.J. Williams, B.J., Notz, W., 2003. The Design and Analysis of Computer Experiments (New York: Springer).

Heitmann, K., Higdon, D., Nakhleh, C., Habib, S., 2006. Cosmic Calibration. The Astrophysical Journal, 646, 2, L1-L4.

See Also

`createGP` for details of the gp object; `gp.list` for details of the `gp.list` object; `mlegp-svd-functions` for details on fitting Gaussian processes to high-dimensional data using principle component weights; the L-BFGS method uses C code written by Naoaki Okazaki (http://www.chokkan.org/software/liblbfgs)

Examples

```
###### fit a single Gaussian process ######
x = -5:5; y1 = sin(x) + rnorm(length(x),sd=.1)
fit1 = mlegp(x, y1)

## summary and diagnostic plots ##
summary(fit1)
plot(fit1)

###### fit a single Gaussian process when replciates are present ######
x = kronecker(-5:5, rep(1,3))
y = x + rnorm(length(x))

## recommended approach: GP fit to sample means; nugget calcualted from sample variances ##
fit1 = mlegp(x,y, nugget.known = 1)
```

```
## original approach: GP fit to all observations; look for MLE of nugget ##
fit2 = mlegp(x,y)


###### fit multiple Gaussian processes to multiple observations ######
x = -5:5
y1 = sin(x) + rnorm(length(x),sd=.1)
y2 = sin(x) + 2*x + rnorm(length(x), sd = .1)
fitMulti = mlegp(x, cbind(y1,y2))

## summary and diagnostic plots ##
summary(fitMulti)
plot(fitMulti)


###### fit multiple Gaussian processes using principle component weights ######

## generate functional output ##
x = seq(-5,5,by=.2)
p = 1:50
y = matrix(0,length(p), length(x))
for (i in p) {
y[i,] = sin(x) + i + rnorm(length(x), sd  = .01)
}

## we now have 10 functional observations (each of length 100) ##
for (i in p) {
plot(x,y[i,], type = "l", col = i, ylim = c(min(y), max(y)))
par(new=TRUE)
}

## fit GPs to the two most important principle component weights ##
numPCs = 2
fitPC = mlegp(p, t(y), PC.num = numPCs)
plot(fitPC) ## diagnostics

## reconstruct the output Y = UDV'
Vprime = matrix(0,numPCs,length(p))
Vprime[1,] = predict(fitPC[[1]])
Vprime[2,] = predict(fitPC[[2]])

predY = fitPC$UD%*%Vprime
m1 = min(y[39,], predY[,39])
m2 = max(y[39,], predY[,39])

plot(x, y[39,], type="l", lty = 1, ylim = c(m1,m2), ylab = "original y" )
par(new=TRUE)
plot(x, predY[,39], type = "p", col = "red", ylim = c(m1,m2), ylab = "predicted y" )

## Not run:
### fit GPs in parallel ###
library(snowfall)
sfInit(parallel = TRUE, cpus = 2, slaveOutfile = "slave.out")
```

```
sfLibrary(mlegp)
fitPC = mlegp(p, t(y), PC.num = 2, parallel = TRUE)

## End(Not run)
```

---

mlegp-nugget-related *Gaussian Process Nugget Related Functions*

Description

Functions for detecting replicates and for calculating sample variance at specific design points

Usage

```
varPerReps(X, Y)
estimateNugget(X, Y)
anyReps(X)
```

Arguments

X	the design matrix
Y	a vector (or 1 column matrix) of observations

Value

varPerReps returns a 1-column matrix where element i corresponds to the sample variance in observations corresponding to design point X[i]

estimateNugget returns a double calculated by taking the mean of the matrix returned by varPerReps

anyReps returns TRUE if two or more rows of X are identical

Note

These functions are used by mlegp to set an initial value of the nugget when a constant nugget is being estimated. The function varPerReps may also be useful for specifying the form of the nugget matrix for use with mlegp.

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

Examples

```
x = matrix(c(1,1,2,3,3))   # the design matrix
y = matrix(c(5,6,7,0,10))  # output

anyReps(x)
varPerReps(x,y)
estimateNugget(x,y)
```

---

plot.gp *Diagnostic Plots for Gaussian processes*

Description

Cross-Validated Diagnostic Plots for Gaussian Processes

Usage

```
## S3 method for class 'gp'
plot(x, type = 0, params = NULL, sds = 1, CI.at.point = FALSE, ...)
```

Arguments

x	an object of class gp
type	the type of graph to plot, 0 by default (see Details)
params	for graph types 2 and 3, a vector of parameter names (or parameter indices) to plot against. By default, all parameters are looked at
sds	the number of standard deviations to use for confidence bands/intervals, for graph types 0-3
CI.at.point	if TRUE, will plot confidence intervals at each predicted point, rather than bands, which is the default
...	additional arguments to plot, but cannot overwrite xlab or ylab

Details

All plots involve cross-validated predictions and/or cross-validated standardized residuals. The cross-validation is in the sense that for predictions made at design point x, all observations at design point x are removed from the training set.

Where relevant, open circles correspond to Gaussian process predictions, black lines correspond to the observations, and red lines correspond to confidence bands. The argument type determines the type of graph displayed, and is one of the following integers:

0 for observed vs. predicted AND observed vs. standardized residual (default),
1 for observed vs. predicted only,

2 for parameter vs. predicted for all parameters,
3 for parameter vs. standardized residual for all parameters,
4 for normal quantile plot and histogram of standardized residuals

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

See Also

CV for cross-validation, plot.gp.list for plotting gp.list objects

Examples

```
## fit the gp ##
x = seq(-5,5,by=.5)
y = sin(x) + rnorm(length(x), sd=.1)
fit = mlegp(x,y)

## plot diagnostics ##
plot(fit)
plot(fit, type = 2)
```

---

plot.gp.list *Diagnostics Plots for Gaussian Process Lists*

Description

Cross-validated Diagnostic Plots For Gaussian Process Lists

Usage

```
## S3 method for class 'gp.list'
plot(x, sds = 1, CI.at.point = FALSE, ...)
```

Arguments

x	an object of class gp.list
sds	the number of standard deviations to use for confidence bands / intervals
CI.at.point	if TRUE, will plot confidence intervals around each predicted point, rather than bands, which is the default
...	not used; for compatibility with plot.gp

Details

All plots involve cross-validated predictions and/or cross-validated standardized residuals. The cross-validation is in the sense that for predictions made at design point x, all observations at design point x are removed from the training set.

Where relevant, open circles correspond to Gaussian process output, black lines correspond to the observations, and red lines correspond to confidence bands.

For each Gaussian process in x, plot.gp is called using graph type 1, which plots cross-validated predictions vs. observed values.

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

See Also

plot.gp, CV

Examples

```
## create data for multiple responses ##
x = seq(-5,5)
z1 = 10 - 5*x + rnorm(length(x))
z2 = 4 - 5*x + rnorm(length(x))
z3 = 7*sin(x) + rnorm(length(x))

## fit multiple Gaussian processes ##
fitMulti = mlegp(x, cbind(z1,z2,z3))

## plot diagnostics ##
plot(fitMulti)
```

---

plotObservedEffects *Plot Observed Values Vs. Each Dimension of the Design Matrix*

Description

Constructs multiple graphs, plotting each parameter from the design matrix on the x-axis and observations on the y-axis

Usage

```
plotObservedEffects(x, ...)
```

Arguments

x	an object of class gp or a design matrix
...	if x is a design matrix, a vector of observations; if x is of class gp, a vector of parameter numbers or parameter names to plot (by default, all parameters will be graphed)

Details

if x is NOT of class gp (i.e., x is a design matrix), all columns of x will be plotted separately against the vector of observations

if x is of class gp, the specified columns of the design matrix of x will be plotted against the the observations

Note

It is often useful to use this function before fitting the gaussian process, to check that the observations are valid

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

Examples

```
## create the design and output matrices ##
x1 = kronecker(seq(0,1,by=.25), rep(1,5))
x2 = rep(seq(0,1,by=.25),5)
z = 4 * x1 - 2*x2 + x1 * x2 + rnorm(length(x1), sd = 0.001)

## look at the observed effects prior to fitting the GP ##
plotObservedEffects(cbind(x1,x2), z)

## fit the Gaussian process ##
fit = mlegp(cbind(x1,x2), z, param.names = c("x1", "x2"))

## look at the observed effects of the fitted GP (which are same as above)
plotObservedEffects(fit)
```

---

predict.gp *Gaussian Process Predictions*

Description

Gaussian Process Predictions

Usage

```
## S3 method for class 'gp'
predict(object, newData = object$X, se.fit = FALSE, ...)
```

Arguments

object	an object of class gp
newData	an optional data frame or matrix with rows corresponding to inputs for which to predict. If omitted, the design matrix X of object is used.
se.fit	a switch indicating if standard errors are desired
...	for compatibility with generic method predict

Details

The Gaussian process is used to predict output at the design points newData; if the logical se.fit is set to TRUE, standard errors (standard deviations of the predicted values) are also calculated. Note that if the Gaussian process contains a nugget term, these standard deviations correspond to standard deviations of predicted expected values, and NOT standard deviations of predicted observations. However, the latter can be obtained by noting that the variance of a predicted observation equals the variance of the predicted expected value plus a nugget term.

If newData is equal to the design matrix of object (the default), and there is no nugget term, the Gaussian process interpolates the observations and the predictions will be identical to component Z of object. For cross-validation, the function CV should be used.

Value

predict.gp produces a vector of predictions. If se.fit is TRUE, a list with the following components is returned:

fit	vector as above
se.fit	standard error of the predictions

Note

for predictions with gp.list objects, call predict.gp separately for each gp in the list

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

See Also

For cross-validated predictions, see CV

Examples

```
x = -5:5; y = sin(x) + rnorm(length(x), sd = 0.001)
fit = mlegp(x,y)
predict(fit, matrix(c(2.4, 3.2)))
round(predict(fit) - fit$Z, 6)   ## predictions at design points match the observations (because there is no n

# this is not necessarily true if there is a nugget
fit = mlegp(x,y, min.nugget = 0.01)
round(predict(fit) - fit$Z,6)
```

---

print.gp *Gaussian Process Summary Information*

Description

prints a summary of a Gaussian process object

Usage

```
## S3 method for class 'gp'
print(x, ...)
```

Arguments

x	an object of class gp
...	for compatibility with generic method print

Details

prints a summary of the Gaussian process object x, by calling summary.gp

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

See Also

summary.gp for more description of the output

Examples

```
x = -5:5; y1 = sin(x) + rnorm(length(x),sd=.1)
fit1 = mlegp(x, y1)
print(fit1)
```

print.gp.list *Gaussian Process List Summary Information*

Description

prints a summary of a Gaussian process list object, or (a subset) of its components

Usage

```
## S3 method for class 'gp.list'
print(x, nums = NULL, ...)
```

Arguments

x	an object of class gp.list
nums	optionally, a vector of integers corresponding to Gaussian processes in the list to summarize
...	for compatibility with generic method print

Details

if nums is NULL, prints summary information for the Gaussian process list object x, using summary.gp.list

if nums is not NULL, prints summary information for each Gaussian process specified by nums, using summary.gp

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

See Also

summary.gp.list, summary.gp

Examples

```
x = -5:5
y1 = sin(x) + rnorm(length(x),sd=.1)
y2 = sin(x) + 2*x + rnorm(length(x), sd = .1)
fitMulti = mlegp(x, cbind(y1,y2))
print(fitMulti)  ## summary of the Gaussian process list
print(fitMulti, nums = 1:2) ## summary of Gaussian processes 1 and 2
```

summary.gp *Gaussian Process Summary Information*

Description

prints a summary of a Gaussian process object

Usage

```
## S3 method for class 'gp'
summary(object, ...)
```

Arguments

object	an object of class gp
...	for compatibility with generic method summary

Details

prints a summary of the Gaussian process object object. Output should be self explanatory, except for possibly CV RMSE, the cross-validated root mean squared error (the average squared difference between the observations and cross-validated predictions); and CV RMaxSE, the maximum cross-validated root squared error. If the design in the Gaussian process object contains any replicates, the root mean pure error (RMPE), which is the square root of the average within replicate variance and the root max pure error (RMaxPE) are also reported.

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

See Also

createGP for details of the Gaussian process object

Examples

```
## no replicates in the design matrix ##
x1 = -5:5; y1 = sin(x1) + rnorm(length(x1),sd=.1)
fit1 = mlegp(x1, y1)
summary(fit1)

## with replicates in the design matrix ##
x2 = kronecker(x1, rep(1,3))
y2 = sin(x2) + rnorm(length(x2), sd = .1)
fit2 = mlegp(x2,y2)
summary(fit2)
```

summary.gp.list *Gaussian Process List Summary Information*

Description

prints a summary of a Gaussian process list object, or (a subset) of its components

Usage

```
## S3 method for class 'gp.list'
summary(object, nums = NULL, ...)
```

Arguments

object	an object of type gp.list
nums	optionally, a vector of integers corresponding to Gaussian processes in the list to summarize
...	for compatibility with generic method summary

Details

if nums is NULL, prints out a summary of the Gaussian process list

if nums is not NULL, displays a summary of the Gaussian processes specified by nums by calling summary.gp for each Gaussian process

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

See Also

summary.gp

Examples

```
x = -5:5
y1 = sin(x) + rnorm(length(x),sd=.1)
y2 = sin(x) + 2*x + rnorm(length(x), sd = .1)
fitMulti = mlegp(x, cbind(y1,y2))
summary(fitMulti)  ## summary of the Gaussian process list
summary(fitMulti, nums = 1:2) ## summary of Gaussian processes 1 and 2
```

uniqueSummary *Summary of outputs for each unique input*

Description

Finds sample means and variances for a matrix of observations when replicates are present

Usage

```
uniqueSummary(X,Y)
```

Arguments

X	matrix of inputs
Y	matrix of outputs corresponding to x

Details

uniqueSummary calculates sample means and variances for each unique input. For input values with no replicates, the sample variance will be 'NA'. This function is used by mlegp to fit a GP to a collection of means observations at each design point.

Value

A list with the following components:

reps	the number of reps for each unique design point
uniqueX	the input matrix with duplicate inputs removed
uniqueMeans	sample means corresponding to each unique input
uniqueVar	sample variances corresponding to each unique input

Author(s)

Garrett M. Dancik < garrett.dancik@gmail.com >

Examples

```
x = c(1,1,2,2,3)
y = x + rnorm(length(x))
uniqueSummary(x,y)
```

Index

∗Topic **hplot**
plot.gp, 9
plot.gp.list, 10
plotObservedEffects, 11
∗Topic **htest**
mlegp-nugget-related, 8
∗Topic **methods**
is.gp, 3
plot.gp, 9
plot.gp.list, 10
plotObservedEffects, 11
print.gp, 14
print.gp.list, 15
summary.gp, 16
summary.gp.list, 17
uniqueSummary, 18
∗Topic **models**
mlegp, 4
predict.gp, 12
∗Topic **package**
mlegp-package, 2
∗Topic **print**
print.gp, 14
print.gp.list, 15
summary.gp, 16
summary.gp.list, 17
∗Topic **smooth**
mlegp, 4
predict.gp, 12
∗Topic **ts**
mlegp, 4

anyReps *(*mlegp-nugget-related*)*, 8

calcPredictionError *(*predict.gp*)*, 12
createGP, *6*, *16*
CV, *10*, *11*, *13*

estimateNugget *(*mlegp-nugget-related*)*, 8

gp.list, *6*

is.gp, 3

mlegp, 4
mlegp-nugget-related, 8
mlegp-package, 2
mlegp2 *(*mlegp*)*, 4

nugget *(*mlegp-nugget-related*)*, 8

plot.gp, 9, *11*
plot.gp.list, *10*, 10
plotObservedEffects, 11
plotObservedEffectsDefault *(*plotObservedEffects*)*, 11
predict.gp, 12
predictMu *(*predict.gp*)*, 12
predictNewZ *(*predict.gp*)*, 12
print.gp, 14
print.gp.list, 15

summary.gp, *14*, *15*, 16, *17*
summary.gp.list, *15*, 17

uniqueSummary, 18

varPerReps *(*mlegp-nugget-related*)*, 8