Package ‘gjam’

March 18, 2020

Type Package
Title Generalized Joint Attribute Modeling
Version 2.3.1
Date 2020-3-18
Author James S. Clark, Daniel Taylor-Rodriquez
Maintainer James S. Clark <jimclark@duke.edu>
Description Analyzes joint attribute data (e.g., species abundance) that are combinations of continuous and discrete data with Gibbs sampling. Full model and computation details are described in Clark et al. (2018) <doi:10.1002/ecm.1241>.
License GPL (>= 2)
Imports Rcpp (>= 0.11.5), MASS, RANN
Depends R (>= 2.10)
LinkingTo Rcpp, RcppArmadillo
LazyData true
Suggests knitr
VignetteBuilder knitr
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-03-18 14:40:05 UTC

R topics documented:

gjam-package ................................................................. 2
gjam ................................................................. 4
gjamCensorY ................................................................. 7
gjamDeZero ................................................................. 9
gjamFillMissingTimes .................................................. 10
gjamIIE ................................................................. 12
gjamIIEplot ............................................................... 13
gjamOrdination ............................................................ 15
**gjam-package**

Description

Inference and prediction for jointly distributed responses that are combinations of continuous and discrete data. Functions begin with `gjam` to avoid conflicts with other packages.

Details

- **Package:** gjam
- **Type:** Package
- **Version:** 2.3.1
- **Date:** 2020-3-18
- **License:** GPL (>= 2)
- **URL:** http://sites.nicholas.duke.edu/clarklab/code/

The generalized joint attribute model (gjam) analyzes multivariate data that are combinations of presence-absence, ordinal, continuous, discrete, composition, zero-inflated, and censored. It does so as a joint distribution over response variables. gjam provides inference on sensitivity to input variables, correlations between responses on the data scale, model selection, and prediction.

Importantly, analysis is done on the observation scale. That is, coefficients and covariances are interpreted on the same scale as the data. Contrast this approach with standard Generalized Linear Models, where coefficients and covariances are difficult to interpret and cannot be compared across responses that are modeled on different scales.

gjam was motivated by species distribution and abundance data in ecology, but can provide an attractive alternative to traditional methods wherever observations are multivariate and combine multiple scales and mixtures of continuous and discrete data.

gjam can be used to model ecological trait data, where species traits are translated to locations as community-weighted means and modes.

Posterior simulation is done by Gibbs sampling. Analysis is done by these functions, roughly in order of how frequently they might be used:


**gjam-package**

- **gjam** fits model with Gibbs sampling.
- **gjamSimData** simulates data for analysis by gjam.
- **gjamPriorTemplate** sets up prior distribution for coefficients.
- **gjamSensitivity** evaluates sensitivity to predictors from gjam.
- **gjamCensorY** defines censored values and intervals.
- **gjamTrimY** trims the response matrix and aggregates rare types.
- **gjamPlot** plots output from gjam.
- **gjamSpec2Trait** constructs plot by trait matrix.
- **gjamPredict** does conditional prediction.
- **gjamOrdination** ordinates the response matrix.
- **gjamDeZero** de-zeros response matrix for storage.
- **gjamReZero** recovers response matrix from de-zeroed format.
- **gjamIIE** evaluates indirect effects and interactions.
- **gjamIIEplot** plots indirect effects and interactions.
- **gjamSpec2Trait** generates trait values.
- **gjamPoints2Grid** aggregates incidence data to counts on a lattice.

**Author(s)**

Author: James S Clark, <jimclark@duke.edu>, Daniel Taylor-Rodriquez

**References**


**See Also**


A more detailed vignette is can be obtained with:

browseVignettes('gjam')
gjam

**Gibbs sampler for gjam data**

**Description**

Analyzes joint attribute data (e.g., species abundance) with Gibbs sampling. Input can be output from `gjamSimData`. Returns a list of objects from Gibbs sampling that can be plotted by `gjamPlot`.

**Usage**

```
gjam(formula, xdata, ydata, modelList)
```

---

### S3 method for class 'gjam'

```
print(x, ...)
```

---

### S3 method for class 'gjam'

```
summary(object, ...)
```

**Arguments**

- **formula**
  R formula for model, e.g., `~ x1 + x2`.

- **xdata**
  `data.frame` containing predictors in `formula`. If not found in `xdata` variables, they must be available from the user’s workspace.

- **ydata**
  `n` by `S` response matrix or `data.frame`. Column names are unique labels, e.g., species names. All columns will be included in the analysis.

- **modelList**
  list specifying inputs, including `ng` (number of Gibbs steps), `burnin`, and `typeNames`. Can include the number of holdouts for out-of-sample prediction, `holdoutN`. See **Details**.

- **x**
  object of class `gjam`.

- **object**
  currently, also an object of class `gjam`.

- **...**
  further arguments not used here.

**Details**

Note that `formula` begins with `~`, not `y ~`. The response matrix is passed in the form of a `n` by `S` matrix or `data.frame` `ydata`.

Both predictors in `xdata` and responses in `ydata` can include missing values as `NA`. Factors in `xdata` should be declared using `factor`. For computational stability variables that are not factors are standardized by mean and variance, then transformed back to original scales in output. To retain a variable in its original scale during computation include it in the character string `notStandard` as part of the list `modelList`. (example shown in the vignette on traits).

`modelList` has these defaults and provides these options:

- `ng = 2000`, number of Gibbs steps.
- `burnin = 500`, no. initial steps, must be less than `ng`. 
typeNames can be 'PA' (presenceAbsence), 'CON' (continuous on (-Inf,Inf)), 'CA' (continuous abundance, zero censoring), 'DA' (discrete abundance), 'FC' (fractional composition), 'CC' (count composition), 'OC' (ordinal counts), 'CAT' (categorical classes). typeNames can be a single value that applies to all columns in ydata, or there can be one value for each column.

holdoutN = 0, number of observations to hold out for out-of-sample prediction.

holdoutIndex = numeric(0), numeric vector of observations (row numbers) to holdout for out-of-sample prediction.

censor = NULL, list specifying columns, values, and intervals for censoring, see gjamCensorY.

effort = NULL, list containing 'columns', a vector of length <= S giving the names of columns in in y, and 'values', a length-n vector of effort or a n by S matrix (see Examples). effort can be plot area, search time, etc. for discrete count data 'DA'.

FULL = F in modelList will save full prediction chains in $chains$ygibbs.

notStandard = NULL, character vector of column names in xdata that should not be standardized.

reductList = list(N = 20,r = 3), list of dimension reduction parameters, invoked when reductList is included in modelList or automatically when ydata has too many columns. See vignette on Dimension Reduction.

random, character string giving the name of a column in xdata that will be used to specify random effects. The random group column should be declared as a factor. There should be replication, i.e., each group level occurs multiple times.

REDUCT = F in modelList overrides automatic dimension reduction.

FCgroups,CCgroups, are length-S vectors assigned to columns in ydata indicating composition 'FC' or 'CC' group membership. For example, if there are two 'CA' columns in ydata followed by two groups of fractional composition data, each having three columns, then typeNames = c('CA','CA','FC','FC','FC','FC','FC','FC') and FCgroups = c(0,0,1,1,1,2,2,2). note: gjamSimData is not currently set up to simulate multiple composition groups, but gjam will model it.

PREDICTX = T executes inverse prediction of x. Speed-up by setting PREDICTX = F.

ematAlpha = .5 is the probability assigned for conditional and marginal independence in the ematrix.

traitList = list(plotByTrait,traitTypes,specByTrait), list of trait objects. See vignette on Trait analysis.

More detailed vignettes can be obtained with:

browseVignettes('gjam')

Value

Returns an object of class "gjam", which is a list containing the following components:

call function call

chains list of MCMC matrices, each with ng rows; includes coefficients bgibbs(Q*S columns), bgibbsUn (unstandardized for x), sensitivity fgibbs (Q1 columns), and fbgibbs (Q1 columns, where Q1 = Q-1, unless there are multilevel factors); covariance sgibbs has S*(S+1)/2 columns (REDUCT == F) or N*r columns (REDUCT == T).
fit list of diagnostics (DIC, rmspeAll, rmspeBySpec, xscore, yscore).

inputs list of input summaries, including breakMat (partition matrix), classBySpec (interval assignment), designTable (summary of design matrix), [FactorBeta, interBeta, intMat, linFactor] (factor and interaction information), other, notOther (response columns to exclude and not). [standMat, standRows, standX] means and variances to standardize x, [x, xdata, y] cleaned versions of data.

missing list of missing objects, including locations for predictors xmiss and responses ymiss in xdata and ydata, respectively, predictor means xmissMu and standard errors xmissSe, response means ymissMu and standard errors ymissSe.

modelList list of model specifications from input modelList.

parameters list of parameter estimates, including coefficient matrices on standardized (betaMu, betaSe), unstandardized (betaMuUn, betaSeUn), and dimensionless (fBetaMu, fBetaSd) scales; correlation (corMu, corSe) and covariance (sigMu, sigSe) matrices; sensitivities to predictors (fMatrix, fMu, fSe); environmental response matrix (ematrix), with locations of zero elements, conditionally (whConZero) and marginally (whichZero), set at probability level modelList$ematAlpha; and latent variables (wMu, wSd).

prediction list of predicted values, including species richness (responses predicted > 0); inverse predicted x (xpredMu, xpredSd) and predicted y (ypredMu, ypredSd) matrices.

If traits are modeled, then parameters will additionally include betaTraitMu, betaTraitSe (coefficients), sigmaTraitMu, sigmaTraitSe (covariance). prediction will additionally include tMuOrd (ordinal trait means), tMu, tSe (trait predictions).

Author(s)
James S Clark, <jimclark@duke.edu>

References

See Also
gjamSimData simulates data
A more detailed vignette is can be obtained with:
browseVignettes('gjam')
website ‘http://sites.nicholas.duke.edu/clarklab/code/’.

Examples
## Not run:
## combinations of scales
types <- c('DA', 'DA', 'OC', 'OC', 'OC', 'CC', 'CC', 'CC', 'CC', 'CA', 'CA', 'PA', 'PA')
f <- gjamSimData(S = length(types), typeNames = types)
ml <- list(ng = 50, burnin = 5, typeNames = f$typeNames)
out <- gjam(f$formula, f$xdata, f$ydata, modelList = ml)
summary(out)

# repeat with ng = 5000, burnin = 500, then plot data:
pl <- list(trueValues = f$trueValues)
gjamPlot(out, plotPars = pl)

## discrete abundance with heterogeneous effort
S <- 5
n <- 1000
eff <- list( columns = 1:S, values = round(runif(n,.5,5),1) )
f <- gjamSimData(n, S, typeNames='DA', effort=eff)
ml <- list(ng = 500, burnin = 50, typeNames = f$typeNames, effort = eff)
out <- gjam(f$formula, f$xdata, f$ydata, modelList = ml)
summary(out)

# repeat with ng = 2000, burnin = 500, then plot data:
pl <- list(trueValues = f$trueValues)
gjamPlot(out, plotPars = pl)

## End(Not run)

---

**gjamCensorY**

*Censor gjam response data*

**Description**

Returns a list with censored values, intervals, and censored response matrix y.

**Usage**

```r
gjamCensorY(values, intervals, y, type='CA', whichcol = c(1:ncol(y)))
```

**Arguments**

- **values**
  Values in y that are censored, specified by intervals

- **intervals**
  Matrix having two rows and one column for each value in values. The first row holds lower bounds. The second row holds upper bounds. See **Examples**.

- **y**
  Response matrix, n rows by S columns. All values within intervals will be replaced with values

- **type**
  Response type, see typeNames in **gjam**

- **whichcol**
  Columns in y that are censored (often not all responses are censored)

**Details**

Any values in y that fall within censored intervals are replaced with censored values. The example below simulates data collected on an ‘octave scale’: 0, 1, 2, 4, 8, ... , an approach to accelerate data collection with approximate bins.
Value

Returns a list containing two elements.

<table>
<thead>
<tr>
<th>y</th>
<th>n by S matrix updated with censored values substituted for those falling within intervals.</th>
</tr>
</thead>
<tbody>
<tr>
<td>censor</td>
<td>list containing $columns that are censored and $partition, a matrix with 3 rows used in gjam and gjamPlot, one column per censor interval. Rows are values, followed by lower and upper bounds.</td>
</tr>
</tbody>
</table>

Author(s)

James S Clark, <jimclark@duke.edu>

References


See Also

gjamSimData simulates data gjam analyzes data

A more detailed vignette is can be obtained with:
browseVignettes('gjam')

website 'http://sites.nicholas.duke.edu/clarklab/code/'.

Examples

## Not run:
# data in octaves
v <- up <- c(0, 2^c(0:4), Inf)
dn <- c(-Inf, v[-length(v)])
i <- rbind( dn, up ) # intervals

f <- gjamSimData(n = 2000, S = 15, Q = 3, typeNames='CA')
y <- f$y
cc <- c(3:6) # censored columns
g <- gjamCensorY(values = v, intervals = i, y = y, whichcol = cc)
y[,cc] <- g$y # replace columns
ml <- list(ng = 50, burnin = 10, censor = g$censor, typeNames = f$typeNames)
output <- gjam(f$formula, xdata = f$xdata, ydata = y, modelList = ml)

#repeat with ng = 2000, burnin = 500, then:
pl <- list(trueValues = f$trueValues, width = 3, height = 3)
gjamPlot(output, pl)

# upper detection limit
up <- 5
v <- up
i <- matrix(c(up,Inf),2)
rownames(i) <- c('down', 'up')

f <- gjamSimData(typeNames='CA')
g <- gjamCensorY(values = v, intervals = i, y = f$y)
ml <- list(ng = 50, burnin = 10, censor = g$censor, typeNames = f$typeNames)
out <- gjam(f$formula, xdata = f$xdata, ydata = g$y, modelList = ml)

# repeat with ng = 2000, burnin = 500, then:
pl <- list(trueValues = f$trueValues, width = 3, height = 3)
gjamPlot(out, pl)

# lower detection limit
lo <- .001
values <- upper <- lo
intervals <- matrix(c(-Inf, lo), 2)
rownames(intervals) <- c('lower', 'upper')

## End(Not run)

gjamDeZero

Compress (de-zero) gjam data

Description

Returns a de-zeroed (sparse matrix) version of matrix ymat with objects needed to re-zero it.

Usage

gjamDeZero(ymat)

Arguments

ymat n by S response matrix

Details

Many abundance data sets are mostly zeros. gjamDeZero extracts non-zero elements for storage.

Value

Returns a list containing the de-zeroed ymat as a vector yvec.

yvec non-zero elements of ymat
n no. rows of ymat
S no. cols of ymat
index index for non-zeros
ynames column names of ymat
gjamFillMissingTimes

Fill out data for time series (state-space) gjam

Description

Fills in predictor, response, and effort matrices for time series data where there are multiple multivariate time series. Time series gjam is still under development.

Usage

```r
gjamFillMissingTimes(xdata, ydata, edata, groupCol, timeCol, groupVars = groupCol,
                     FILLMEANS = FALSE, typeNames = NULL, missingEffort = .1)
```
Arguments

- **xd**
  - n by 0 data.frame holding predictor variables
- **ydata**
  - n by 5 matrix holding response variables
- **edata**
  - n by 5 matrix holding effort
- **groupCol**
  - column name in xdata for group variable, i.e., observations part of the same time series
- **timeCol**
  - column name in xdata for time index
- **groupVars**
  - character vector of column names in xdata having values that are fixed for a value of groupCol, i.e., they do not change with time index in timeCol
- **FILLMEANS**
  - fill new rows in ydata with mean for groupCol times missingEffort; otherwise NA
- **typeName**
  - typenames current limited to 'DA' for discrete counts
- **missingEffort**
  - effort assigned to missing values of edata and ydata

Details

Missing times in the data occur where there are gaps in timeCol column of xdata and the initial time 0 for each sequence. New versions of the data have NA (xdata) or prior values with appropriate weight (ydata). Missing times are filled in xdata, ydata, edata, including a time 0 which serves as a prior mean for ydata for time code1. The group and time indices in columns groupCol and timeCol of xdata reference the time for a given time series. Missing values in the columns groupVars of xdata are filled automatically filled in. This assumes that values for these variables are fixed for the group. If FILLMEANS, the missing values in ydata are filled with means for the group and given a low weight specified in missingEffort.

Value

A list containing the following:

- **xd**
  - filled version of xdata
- **ydata**
  - filled version of ydata
- **edata**
  - filled version of edata
- **timeList**
  - time indices used for computation, including, timeZero (row numbers in new data where each time series begins, with times = 0), timeLast (row numbers in new data where each time series ends), rowInserts (row numbers for all inserted rows), noEffort (rows for which effort in edata is filled with missingEffort)

Author(s)

James S Clark, <jimclark@duke.edu>

References

gjamIIE

Indirect effects and interactions for gjam data

Description

Evaluates direct, indirect, and interactions from a gjam object. Returns a list of objects that can be plotted by gjamIIEplot.

Usage

gjamIIE(output, xvector, MEAN = T, keepNames = NULL, omitY = NULL, sdScaleX = T, sdScaleY = F)

Arguments

output object of class inheriting from "gjam".
xvector vector of predictor values, with names, corresponding to columns in output$x.
MEAN logical, if false, then median used.
omitY character vector of columns in output$y to omit from calculations.
keepNames character vector of columns in output$y. If omitted, all columns used.

sdScaleX standardize coefficients to X scale.
sdScaleY standardize coefficients to correlation scale.

Details

For plotting or recovering effects. The list fit$IIE has matrices for main effects (mainEffect), interactions (intEffect), direct effects (dirEffect), indirect effects (indEffectTo), and standard deviations for each. The direct effects are the sum of main effects and interactions. The indirect effects include main effects and interactions that come through other species, determined by covariance matrix sigma.

If sdScaleX = T effects are standardized from the Y/X to Y scale. This is the typical standardization for predictor variables. If sdScaleY = T effects are given on the correlation scale. If both are true effects are dimensionless. See the gjam vignette on dimension reduction.

Value

A list of objects for plotting by gjamIIEplot.
Author(s)

James S Clark, <jimclark@duke.edu>

References


See Also

gjamIIEplot plots output from gjamIIE
A more detailed vignette is can be obtained with:
browseVignettes('gjam')
web site 'http://sites.nicholas.duke.edu/clarklab/code/'.

Examples

```r
## Not run:
sim <- gjamSimData(S = 12, Q = 5, typeNames = 'CA')
ml <- list(ng = 50, burnin = 5, typeNames = sim$typeNames)
out <- gjam(sim$formula, sim$xdata, sim$ydata, modelList = ml)
xvector <- colMeans(out$inputs$x) #predict at mean values for data
xvector[1] <- 1
fit <- gjamIIE(output = out, xvector)

gjamIIEplot(fit, response = 'S1', effectMu = c('main','ind'),
             effectSd = c('main','ind'), legLoc = 'topleft')

## End(Not run)
```

gjamIIEplot

Plots indirect effects and interactions for gjam data

Description

Using the object returned by gjamIIEplot generates a plot for a response variable.

Usage

```r
gjamIIEplot(fit, response, effectMu, effectSd = NULL,
            ylim = NULL, col='black', legLoc = 'topleft', cex = 1)
```
Arguments

- fit: object from gjamIIE.
- response: name of a column in fit$y to plot.
- effectMu: character vector of mean effects to plot, can include 'main', 'int', 'direct', 'ind'.
- effectSd: character vector can include all or some of effectMu.
- ylim: vector of two values defines vertical axis range.
- col: vector of colors for barplot.
- legLoc: character for legend location.
- cex: font size.

Details

For plotting direct effects, interactions, and indirect effects from an object fit generated by gjamIIE. The character vector supplied as effectMu can include main effects ('main'), interactions ('int'), main effects plus interactions ('direct'), and/or indirect effects ('ind'). The list effectSd draws 0.95 predictive intervals for all or some of the effects listed in effectMu. Bars are contributions of each effect to the response.

For factors, effects are plotted relative to the mean over all factor levels.

Author(s)

James S Clark, <jimclark@duke.edu>

References


See Also

gjamIIE generates output for gjamIIEplot
A more detailed vignette is can be obtained with:
browseVignettes('gjam')
web site 'http://sites.nicholas.duke.edu/clarklab/code/'.

Examples

```r
## Not run:
sim <- gjamSimData(S = 10, Q = 6, typeNames = 'OC')
ml <- list(ng = 50, burnin = 5, typeNames = sim$typeNames)
out <- gjam(sim$formula, sim$xdata, sim$ydata, modellist = ml)
xvector <- colMeans(out$inputs$xStand) # predict at mean values for data, standardized x
xvector[1] <- 1
```
fit <- gjamIIE(out, xvector)

gjamIIEplot(fit, response = 'S1', effectMu = c('main','ind'),
             effectSd = c('main','ind'), legLoc = 'topleft')

## End(Not run)

gjamOrdination

Ordinate gjam data

Description
Ordinate data from a gjam object using correlation corresponding to response matrix E.

Usage

gjamOrdination(output, specLabs = NULL, col = NULL, cex = 1,
                 PLOT=T, method = 'PCA')

Arguments

output object of class "gjam".

specLabs character vector of variable names in colnames(output$y).

col character vector of columns in output$y to label in plots.

在意 text size in plot.

PLOT logical, if true, draw plots.

method character variable can specify 'NMDS'.

Details
Ordinates the response correlation ematrix contained in output$parameterTables. If method = 'PCA' returns eigenvalues and eigenvectors. If method = 'PCA' returns three NMDS dimensions. If PLOT, then plots will be generated. Uses principle components analysis or non-metric multidimensional scale (NMDS).

Value
eVecs S x S or, if there is an other response variable to be excluded, S-1 x S-1 matrix of eigenvectors for species (rows) by eigenvectors (columns).

eValues If method = 'PCA' returns length-S or, there is an other response variable to be excluded, length-S-1 vector of eigenvalues. If method = 'NMDS' this variable is NULL.

Author(s)

James S Clark, <jimclark@duke.edu>
References

See Also
gjam fits the data
A more detailed vignette is can be obtained with:
browseVignettes('gjam')
website 'http://sites.nicholas.duke.edu/clarklab/code/'.

Examples
## Not run:
f <- gjamSimData(S = 30, typeNames = 'CA')
ml <- list(ng = 30, burnin = 5, typeNames = f$typeNames, holdoutN = 10)
output <- gjam(f$formula, f$xdata, f$ydata, modelList = ml)
ePCA <- gjamOrdination(output, PLOT=FALSE)
eNMDS <- gjamOrdination(output, PLOT=FALSE, method='NMDS')
## End(Not run)

gjamPlot

Plot gjam analysis

Description
Constructs plots of posterior distributions, predictive distributions, and additional analysis from output of gjam.

Usage
gjamPlot(output, plotPars)

Arguments
output object of class "gjam"
plotPars list having default values described in Details
Details

plotPars a list that can contain the following, listed with default values:

- **PLOTY = T**  
  plot predicted y.

- **PLOTX = T**  
  plot inverse predicted x.

- **PREDICTX = T**  
  inverse prediction of x; does not work if PREDICTX = F in \texttt{link[gjam]}.  

- **ncluster**  
  number of clusters to highlight in cluster diagrams, default based on S.

- **CORLINES = T**  
  draw grid lines on grid plots of R and E.

- **cex = 1**  
  text size for grid plots, see \texttt{par}.

- **BETAGRID = T**  
  draw grid of beta coefficients.

- **PLOTALLY = F**  
  an individual plot for each column in y.

- **SMALLPLOTS = T**  
  avoids plot margin error on some devices, better appearance if FALSE.

- **GRIDPLOTS = F**  
  cluster and grid plots derived from parameters; matrices R and E are discussed in Clark et al. (2016).

- **SAVEPLOTS = F**  
  plots saved in pdf format.

- **outfolder = 'gjamOutput'**  
  folder for plot files if SAVEPLOTS = T.

- **width, height = 4**  
  can be small values, in inches, to avoid plot margin error on some devices.

- **specColor = 'black'**  
  color for posterior box-and-whisker plots.

- **ematAlpha = .95**  
  prob threshold used to infer that a covariance value in Emat is not zero.

- **ncluster = 4**  
  number of clusters to identify in ematrix.

The 'plot margin' errors mentioned above are device-dependent. They can be avoided by specifying small width,height (in inches) and by omitting the grid plots (GRIDPLOTS = F). If plotting does not produce a 'plot margin error', better appearance is obtained with SMALLPLOTS = F.

Names will not be legible for large numbers of species. Specify specLabs = F and use a character vector for specColor to identify species groups (see the gjam vignette on dimension reduction).

Box and whisker plots bound 0.68 and 0.95 credible and predictive intervals.

Value

Summary tables of parameter estimates are:

- **betaEstimates**  
  Posterior summary of beta coefficients.

- **clusterIndex**  
  cluster index for responses in grid/cluster plots.

- **clusterOrder**  
  order for responses in grid/cluster plots.

- **eComs**  
  groups based on clustering ematrix.

- **ematrix**  
  \( S \times S \) response correlation matrix for E.

- **eValues**  
  eigenvalues of ematrix.

- **eVecs**  
  eigenvectors of ematrix.

- **fit**  
  list containing DIC, score, and rmspe.

Author(s)

James S Clark, <jimclark@duke.edu>
References


See Also
gjam A more detailed vignette is can be obtained with:
browseVignettes('gjam')
website 'http://sites.nicholas.duke.edu/clarklab/code/'.

Examples

## Not run:
## ordinal data
f <- gjamSimData(S = 15, Q = 3, typeNames = 'OC')
ml <- list(ng = 200, burnin = 50, typeNames = f$typeNames, holdoutN = 10)
out <- gjam(f$formula, f$xdata, f$ydata, modelList = ml)
# repeat with ng = 2000, burnin = 500, then plot data here:
pl <- list(trueValues = f$trueValues, width=3, height=2)
fit <- gjamPlot(output = out, plotPars = pl)
## End(Not run)

---

gjamPoints2Grid

**Incidence point pattern to grid counts**

Description

From point pattern data in (x, y) generates counts on a lattice supplied by the user or specified by lattice size or density. For analysis in gjam as counts (known effort) or count composition (unknown effort) data.

Usage

gjamPoints2Grid(specs, xy, nxy = NULL, dxy = NULL,
predGrid = NULL, effortOnly = TRUE)

Arguments

specs character vector of species names or codes.
xy matrix with rows = length(specs) and columns for (x, y).
nxy length-2 numeric vector with numbers of points evenly spaced on (x, y).
dxy length-2 numeric vector with distances for points evenly spaced on (x, y).
predGrid matrix with 2 columns for (x, y).
effortOnly logical to return only points where counts are positive (e.g., effort is unknown).
gjamPoints2Grid

Details

For incidence data with species names specs and locations (x,y) constructs a lattice based a prediction grid predGrid, at a density of (dx,dy), or with numbers of lattice points (nx,ny). If effortOnly = T, returns only points with non-zero values.

A prediction grid predGrid would be passed when counts by locations of known effort are required or where multiple groups should be assign to the same lattice points.

The returned gridBySpec can be analyzed in gjam with known effort as count data "DA" or with unknown effort as count composition data "CC".

Value

gridBySpec matrix with rows for grid locations, columns for counts by species.
predGrid matrix with columns for (x, y) and rows matching gridBySpec.

Author(s)

James S Clark, <jimclark@duke.edu>

References


See Also

gjam A more detailed vignette is can be obtained with: browseVignettes('gjam')

Examples

## Not run:
## random data
n <- 100
s <- sample( letters[1:3], n, replace = TRUE)
xy <- cbind( rnorm(n,0,.2), rnorm(n,10,2) )

nx <- ny <- 5 # uniform 5 X 5 lattice
f <- gjamPoints2Grid(s, xy, nxy = c(nx, ny))
plot(f$predGrid[,1], f$predGrid[,2], cex=.1, xlim=c(-1,1), ylim=c(0,20), xlab = 'x', ylab = 'y')
text(f$predGrid[,1], f$predGrid[,2], rowSums(f$gridBySpec))

dx <- .2 # uniform density
dy <- 1.5
g <- gjamPoints2Grid(s, xy, dxy = c(dx, dy))
text(g$predGrid[,1], g$predGrid[,2], rowSums(g$gridBySpec), col='brown')

p <- cbind( runif(30, -1, 1), runif(30, 0, 20) ) # irregular lattice
gjamPredict

Predict gjam data

Description
Predicts data from a gjam object, including conditional and out-of-sample prediction.

Usage
```
gjamPredict(output, newdata = NULL, y2plot = NULL, ylim = NULL,
             FULL = FALSE)
```

Arguments
- `output` object of class "gjam".
- `newdata` a list of data for prediction, see Details.
- `y2plot` character vector of columns in `output$y` to plot.
- `ylim` vector of lower and upper bounds for prediction plot
- `FULL` will return full chains for predictions as `output$ychains`

Details
If `newdata` is not specified, the response is predicted from `xdata` as an in-sample prediction. If `newdata` is specified, prediction is either conditional or out-of-sample.

Conditional prediction on a new set of `y` values is done if `newdata` includes the matrix `ycondData`, which holds columns to condition on. `ycondData` must be a matrix and have column names matching those in `y` that it will replace. `ycondData` must have at least one column, but fewer than `ncol(y)` columns. Columns not included in `ycondData` will be predicted conditionally.

Alternatively, the list `newdata` can include a new version of `xdata` for out-of-sample prediction. The version of `xdata` passed in `newdata` has the columns with the same names and variable types as `xdata` passed to `gjam`. Note that factor levels must also match those included when fitting the model. All columns in `y` will be predicted out-of-sample.

For count composition data the effort (total count) is 1000.

Because there is no out-of-sample effort for ‘CC’ data, values are predicted on the [0, 1] scale.

See examples below.
**gjamPredict**

**Value**

- **x**
  - design matrix.
- **sdList**
  - list of predictive means and standard errors includes \( y\mu, y\text{Pe} \) (predictive mean, SE), \( w\mu, w\text{Se} \) (mean latent states and SEs)
- **piList**
  - predictive intervals. only generated if \( \text{length}(y) < 10000 \), includes \( y\text{Lo}, y\text{Hi} \) (0.025, 0.975) prediction interval, \( w\text{Lo}, w\text{Hi} \) (0.025, 0.975) for latent states
- **prPresent**
  - \( n \times S \) matrix of probabilities of presence
- **ematrix**
  - effort
- **ychains**
  - full prediction chains if FULL = T

**Author(s)**

James S Clark, <jimclark@duke.edu>

**References**


**See Also**

- `gjamSimData` simulates data
- A more detailed vignette is can be obtained with:
  - `browseVignettes('gjam')`
  - web site 'http://sites.nicholas.duke.edu/clarklab/code/'

**Examples**

```r
## Not run:
S <- 5
f <- gjamSimData(n = 200, S = S, Q = 3, typeNames = 'CC')
ml <- list(ng = 50, burnin = 5, typeNames = f$typeNames, holdoutN = 10)
out <- gjam(f$formula, f$xdata, f$ydata, modelList = ml)

# predict data
par(mfrow=c(1,3),bty='n')
gjamPredict(out, y2plot = colnames(f$ydata)) #predict the data in-sample
title('full sample')

# out-of-sample prediction
xdata <- f$xdata[1:20,]
xdata[,3] <- mean(f$xdata[,3]) # mean for x[,3]
xdata[,2] <- seq(-2,2,length=20) # gradient x[,2]
newdata <- list(xdata = xdata, nsim = 50 )
p1 <- gjamPredict(out, newdata = newdata)

# plus/minus 1 prediction SE, default effort = 1000
```

x2 <- p1$x[,2]
ylim <- c(0, max(p1$sdList$yMu[,1] + p1$sdList$yPe[,1]))
plot(x2, p1$sdList$yMu[,1], type='l', lwd=2, ylim=ylim, xlab='x2', ylab = 'Predicted')
lines(x2, p1$sdList$yMu[,1] + p1$sdList$yPe[,1], lty=2)
lines(x2, p1$sdList$yMu[,1] - p1$sdList$yPe[,1], lty=2)

# .95 prediction error
lines(x2, p1$piList$yLo[,1], lty=3)
lines(x2, p1$piList$yHi[,1], lty=3)
title('SE and prediction, Sp 1')

# conditional prediction
ydataCond <- out$inputs$y[,1,drop=FALSE]*0 # set first column to zero
newdata <- list(ydataCond = ydataCond, nsim=50)
p0 <- gjamPredict(output = out, newdata = newdata)
ydataCond <- ydataCond + 20 # first column is 20
newdata <- list(ydataCond = ydataCond, nsim=50)
p1 <- gjamPredict(output = out, newdata = newdata)

plot(out$inputs$y[,4], p0$sdList$yMu[,4], cex=.4, col='orange'); abline(0,1,lty=2)
points(out$inputs$y[,4], p1$sdList$yMu[,4], cex=.4, col='blue')
title('Cond. on 1st Sp')

# conditional, out-of-sample prediction compared with unconditional, in-sample
n <- 1000
S <- 5
f <- gjamSimData(n = n, S = S, Q = 3, typeNames = 'CA')

holdOuts <- sort( sample(n, 50) )
xdata <- f$xdata[-holdOuts,] # fitted data
ydata <- f$ydata[-holdOuts,]

xx <- f$xdata[holdOuts,] # holdout for prediction
yy <- f$ydata[holdOuts,]

ml <- list(ng = 2000, burnin = 50, typeNames = f$typeNames) # fit the non-holdouts
out <- gjam(f$formula, xdata, ydata, modelList = ml)

cdex <- 1:2 # condition on 2 species
ndex <- c(1:S)[-cdex] # conditionally predict others

newdata <- list(xdata = xx, ydataCond = yy[,cdex], nsim = 200) # conditionally predict out-of-sample
p2 <- gjamPredict(output = out, newdata = newdata)

plot(as.matrix(yy[,ndex]), p2$sdList$yMu[,ndex]); abline(0,1,lty=2)
title('Conditional, out-of-sample')
mspeC <- sqrt( mean( (as.matrix(yy[,ndex]) - p2$sdList$yMu[,ndex])^2 ) )

# predict unconditionally, out-of-sample
newdata  <- list(xdata = xx, nsim = 200)
p1  <- gjamPredict(out, newdata = newdata)
points( as.matrix(yy[,ndex]), p1$sdList$yMu[,ndex], col=2)
mspeU  <- sqrt( mean( (as.matrix(yy[,ndex]) - p1$sdList$yMu[,ndex])^2 ) )
e1  <- paste( 'cond, out-of-sample =', round(mspeC, 2) )
e2  <- paste( 'uncond, out-of-sample =', round(mspeU, 2) )
legend('topleft', c(e1, e2), text.col=c(1,2))

## End(Not run)

gjamPriorTemplate

Prior coefficients for gjam analysis

Description

Constructs coefficient matrices for low and high limits on the uniform prior distribution for beta.

Usage

gjamPriorTemplate(formula, xdata, ydata, lo = NULL, hi = NULL)

Arguments

formula object of class formula, starting with ~, matches the formula passed to gjam
xdata  n x Q observation by predictor data.frame
ydata  n x Q observation by response data.frame
lo list of lower limits
hi list of upper limits

Details

The prior distribution for a coefficient beta[q, s] for predictor q and response s, is dunif(lo[q, s], hi[q, s]).
gjamPriorTemplate generates these matrices. The default values are (-Inf, Inf), i.e., all values in lo equal to -Inf and hi equal to Inf. These templates can be modified by changing specific values in lo and/or hi.

Alternatively, desired lower limits can be passed as the list lo, assigned to names in xdata (same limit for all species in ydata), in ydata (same limit for all predictors in xdata), or both, separating names in xdata and ydata by ".". The same convention is used for upper limits in hi.

These matrices are supplied in as list betaPrior, which is included in modelList passed to gjam. See examples and browseVignettes('gjam').

Note that the informative prior slows computation.
Value

A list containing two matrices. lo is a $Q \times S$ matrix of lower coefficient limits. hi is a $Q \times S$ matrix of upper coefficient limits. Unless specified in lo, all values in lo = -Inf. Likewise, unless specified in hi, all values in hiBeta = -Inf.

Author(s)

James S Clark, <jimclark@duke.edu>

References


See Also

gjam

Examples

## Not run:
library(repmis)
source_data("https://github.com/jimclarkatduke/gjam/blob/master/forestTraits.RData?raw=True")

xdata <- forestTraits$xdata
plotByTree <- gjamReZero(forestTraits$treesDeZero) # re-zero
traitTypes <- forestTraits$traitTypes
specByTrait <- forestTraits$specByTrait

tmp <- gjamSpec2Trait(phbs = plotByTree, sbyt = specByTrait,
tTypes = traitTypes)
tTypes <- tmp$tTypes
traity <- tmp$plotByCWM
censor <- tmp$censor

formula <- as.formula(~ temp + deficit)
lo <- list(temp_ring = 0, deficit_drought = 0) # positive combinations
b <- gjamPriorTemplate(formula, xdata, ydata = traity, lo = lo, hi = hi)
bp <- list(loBeta = b$lo, hiBeta = b$hi)

ml <- list(ng=4000, burnin=1000, typeNames = tTypes, censor = censor,
betaPrior = bp)
out <- gjam(formula, xdata, ydata = traity, modelList = ml)

S <- ncol(traity)
sc <- rep('black',S)
sc[colnames(traity)]
pl <- list(SMALLPLOTS=F, specColor=sc)
gjamPlot(output = out, plotPars = pl)
Description
Returns a re-zeroed matrix \( y \) from the de-zeroed vector, a sparse matrix.

Usage

\[
gjamReZero( yDeZero )
\]

Arguments

- **yDeZero**
  - list created by \( gjamReZero \) containing number of rows \( n \), number of columns \( S \), index for non-zeros \( \text{index} \), the vector of non-zero values \( \text{yvec} \), and the column names \( \text{ynames} \).

Details
Many abundance data sets are mostly zeros. \( gjamReZero \) recovers the full matrix from de-zeroed list \( yDeZero \) written by \( gjamDeZero \).

Value

- **ymat**
  - re-zeroed \( n \) by \( S \) matrix.

Author(s)

James S Clark, <jimclark@duke.edu>

References

See Also
- \( gjamDeZero \) to de-zero \( ymat \)
- `browseVignettes('gjam')`
- website: 'http://sites.nicholas.duke.edu/clarklab/code/'.
## Examples

```r
## Not run:
library(repmis)
source_data("https://github.com/jimclarkatduke/gjam/blob/master/fungEnd.RData?raw=True")
ymat <- gjamReZero(fungEnd$yDeZero) # OTUs stored without zeros
length(fungEnd$yDeZero$yvec) # size of stored version
length(ymat) # full size

## End(Not run)
```

### gjamSensitivity

#### Sensitivity coefficients for gjam

**Description**

Evaluates sensitivity coefficients for full response matrix or subsets of it. Uses output from `gjam`. Returns a matrix of samples by predictors.

**Usage**

```r
gjamSensitivity(output, group=NULL, nsim=100)
```

**Arguments**

- `output`: object fitted with `gjam`.
- `group`: character vector of response-variable names from `output$inputs$y`.
- `nsim`: number of samples from posterior distribution.

**Details**

Sensitivity to predictors of entire response matrix or a subset of it, identified by the character string `group`. The equations for sensitivity are given here:

```r
browseVignettes('gjam')
```

**Value**

Returns a `nsim` by predictor matrix of sensitivities to predictor variables.

**Author(s)**

James S Clark. <jimclark@duke.edu>
**References**


**See Also**

`gjamSimData` simulates data

A more detailed vignette is can be obtained with:

```r
browseVignettes('gjam')
```

website 'http://sites.nicholas.duke.edu/clarklab/code/'.

**Examples**

```r
## Not run:
## combinations of scales
types <- c('DA', 'DA', 'OC', 'OC', 'OC', 'CC', 'CC', 'CC', 'CA', 'CA', 'PA', 'PA')
f <- gjamSimData(S = length(types), typeNames = types)
ml <- list(ng = 50, burnin = 5, typeNames = f$typeNames)
out <- gjam(f$formula, f$xdata, f$ydata, modelList = ml)

ynames <- colnames(f$y)
group <- ynames[types == 'OC']

full <- gjamSensitivity(out)
cc <- gjamSensitivity(out, group)

nt <- ncol(full)

ylim <- range(rbind(full, cc))

boxplot( full, boxwex = 0.25, at = 1:nt - .21, col='blue', log='y', ylim = ylim, xaxt = 'n', xlab = 'Predictors', ylab='Sensitivity')
boxplot( cc, boxwex = 0.25, at = 1:nt + .2, col='forestgreen', add=T, xaxt = 'n')
axis(1,at=1:nt,labels=colnames(full))
legend('bottomleft',c('full response','CC data'),
text.col=c('blue','forestgreen'))

## End(Not run)
```

---

### gjamSimData

**Simulated data for gjam analysis**

**Description**

Simulates data for analysis by `gjam`.  

---
Usage

gjamSimData(n = 1000, S = 10, Q = 5, x = NULL, nmiss = 0, typeNames, effort = NULL)

Arguments

- **n**: Sample size
- **S**: Number of response variables (columns) in `y`, typically less than `n`
- **Q**: Number of predictors (columns) in design matrix `x` of `n` or less
- **x**: design matrix, if supplied `n` and `Q` will be set to `nrow(x)` and `ncol(x)`, respectively
- **nmiss**: Number of missing values to in `x` of `n` or less
- **typeNames**: Character vector of data types, see Details
- **effort**: List containing 'columns' specifying columns to which effort applies, and 'values', a length-`n` vector of effort per observation.

Details

Generates simulated data and parameters for analysis by `gjam`. Because both parameters and data are stochastic, not all simulations will give good results.

- `typeNames` can be 'PA' (presenceAbsence), 'CA' (continuous), 'DA' (discrete), 'FC' (fractional composition), 'CC' (count composition), 'OC' (ordinal counts), and 'CAT' (categorical levels). If more than one 'CAT' is included, each defines a multilevel categorical response. One additional type, 'CON' (continuous), is not censored at zero by default.

If defined as a single character value `typeNames` applies to all columns in `y`. If not, `typeNames` is length-`S` character vector, identifying each response by column in `y`. If a column 'CAT' is included, a random number of levels will be generated, `a, b, c, ...`. A more detailed vignette is can be obtained with:

browseVignettes('gjam')

website 'http://sites.nicholas.duke.edu/clarklab/code/'.

Value

- **formula**: R formula for model, e.g., `~ x1 + x2`
- **xdata**: data.frame includes columns for predictors in the design matrix
- **ydata**: data.frame for the simulated response
- **y**: response as a `n` by `S` matrix as assembled in `gjam`.
- **w**: `n` by `S` latent states
- **typeY**: vector of data types corresponding to columns in `y`, see Details
- **typeNames**: vector of data types corresponding to columns in `ydata`
- **trueValues**: list containing true parameter values beta (regression coefficients), sigma (covariance matrix), corSpec (correlation matrix corresponding to sigma), and `cuts` (partition matrix for ordinal data).
- **effort**: see Arguments.
**gjamSimData**

**Author(s)**

James S Clark, <jimclark@duke.edu>

**References**


**See Also**

`gjam`

**Examples**

```r
## Not run:
## ordinal data, show true parameter values
sim <- gjamSimData(S = 5, typeNames = 'OC')
sim$ydata[1:5,]  # example data
sim$trueValues$cuts  # simulated partition
sim$trueValues$beta  # coefficient matrix

## continuous data censored at zero, note latent w for obs y = 0
sim <- gjamSimData(n = 5, S = 5, typeNames = 'CA')
sim$w
sim$y

## continuous and discrete data
types <- c(rep('DA',5), rep('CA',4))
sim <- gjamSimData(n = 10, S = length(types), Q = 4, typeNames = types)
sim$typeNames
sim$ydata

## composition count data
sim <- gjamSimData(n = 10, S = 8, typeNames = 'CC')
totalCount <- rowSums(sim$ydata)
cbind(sim$ydata, totalCount)  # data with sample effort

## multiple categorical responses - compare matrix y and data.frqme ydata
types <- rep('CAT',2)
sim <- gjamSimData(S = length(types), typeNames = types)
head(sim$ydata)
head(sim$y)

## discrete abundance, heterogeneous effort
S <- 5
n <- 1000
ef <- list( columns = 1:S, values = round(runif(n,.5,5),1) )
sim <- gjamSimData(n, S, typeNames = 'DA', effort = ef)
sim$effort$values[1:20]

## combinations of scales, partition only for 'OC' columns
```
gjamSpec2Trait

Ecological traits for gjam analysis

Description

Constructs community-weighted mean-mode (CWMM) trait matrix for analysis with gjam for \( n \) observations, \( S \) species, \( P \) traits, and \( M \) total trait levels.

Usage

```r
gjamSpec2Trait(pbys, sbyt, tTypes)
```

Arguments

- `pbys`: \( n \times S \) plot by species matrix (presence-absence, abundance)
- `sbyt`: \( S \times P \) species by trait matrix
- `tTypes`: \( P \) data types for trait columns

Details

Generates the objects needed for a trait response model (TRM). As inputs the \( sbyt \\) data.frame has \( P \) columns containing numeric values, ordinal scores, and categorical variables, identified by data type in `tTypes`. Additional trait columns can appear in the \( n \times M \) output matrix `plotByCWMM`, because each level of a category becomes a new ‘FC’ column as a CWMM. Thus, \( M \) can exceed \( P \), depending on the number of factors in \( sbyt \). The exception is for categorical traits with only two levels, which can be treated as (0, 1) censored ‘CA’ data.

As output, the CWMM data types are given in `traitTypes`.

The list `censor = NULL` unless some data types are censored. In the example below there are two censored columns.

A detailed vignette on trait analysis is obtained with:
```r
browseVignettes('gjam')
```

Value

- `plotByCWM`: \( n \times M \) matrix of community-weight means (numeric) or modes (ordinal)
- `traitTypes`: character vector of data types for traits
- `specByTrait`: \( S \times M \) matrix translates species to traits
- `censor`: list of censored columns, values, and intervals; see `gjamCensorY`
**gjamTrimY**

Trim gjam response data

---

**Description**

Returns a list that includes a subset of columns in y. Rare species can be aggregated into a single class.

---

**Author(s)**

James S Clark, <jimclark@duke.edu>

**References**


**See Also**

gjam, gjamCensorY

**Examples**

```r
## Not run:
library(repmis)
source_data("https://github.com/jimclarkatduke/gjam/blob/master/forestTraits.RData?raw=True")

xdata <- forestTraits$xdata
plotByTree <- gjamReZero(forestTraits$treesDeZero) # re-zero
traitTypes <- forestTraits$traitTypes
specByTrait <- forestTraits$specByTrait
tmp <- gjamSpec2Trait(pbys = plotByTree, sbyt = specByTrait,
                       tTypes = traitTypes)
tTypes <- tmp$ttraitTypes
traity <- tmp$plotByCWM
censor <- tmp$censor

ml <- list(ng=2000, burnin=500, typeNames = tTypes, censor = censor)
out <- gjam(~ temp + stdage + deficit, xdata, ydata = traity, modelList = ml)
pl <- list(SMALLPLOTS=F)
gjamPlot(output = out, plotPars = pl)
## End(Not run)
```
Usage

```r
gjamTrimY(y, minObs = 2, maxCols = NULL, OTHER = TRUE)
```

Arguments

- **y**: n by S numeric response matrix
- **minObs**: minimum number of non-zero observations
- **maxCols**: maximum number of response variables
- **OTHER**: logical or character string. If \( \text{OTHER} = \text{TRUE} \), rare species are aggregated in a new column 'other'. A character vector contains the names of columns in \( y \) to be aggregated with rare species in the new column 'other'.

Details

Data sets commonly have many responses that are mostly zeros, large numbers of rare species, even singletons. Response matrix \( y \) can be trimmed to include only taxa having \( > \text{minObs} \) non-zero observations or to \( \leq \text{maxCol} \) total columns. The option \( \text{OTHER} \) is recommended for composition data ('CC', 'FC'), where the 'other' column is taken as the reference class. If there are unidentified species they might be included in this class. [See \texttt{gjamSimData} for typeName codes].

Value

Returns a list containing three elements.

- **y**: trimmed version of \( y \).
- **colIndex**: length-S vector of indices for new columns in \( y \).
- **nobs**: number of non-zero observations by column in \( y \).

Author(s)

James S Clark. <jimclark@duke.edu>

References


See Also

- \texttt{gjamSimData} simulates data \texttt{gjam} analyzes data
- A more detailed vignette is can be obtained with:
  - `browseVignettes('gjam')`
  - web site 'http://sites.nicholas.duke.edu/clarklab/code'.
Examples

## Not run:
library(repmis)
source_data("https://github.com/jimclarkatduke/gjam/blob/master/forestTraits.RData?raw=True")

y <- gjamReZero(fungEnd$yDeZero)  # re-zero data
dim(y)
y <- gjamTrimY(y, minObs = 200)$y  # species in >= 200 observations
dim(y)
tail(colnames(y))  # last column is 'other'

## End(Not run)
Index

*Topic package
  gjam-package, 2

class, 5, 12, 15, 16, 20

GJAM (gjam-package), 2
gjam, 3, 4, 7, 8, 12, 16, 18–20, 24, 26–32
gjam-package, 2
gjamCensorY, 3, 5, 7, 30, 31
gjamDeZero, 3, 9, 9, 25
gjamFillMissingTimes, 10
gjamIIE, 3, 12, 14
gjamIIEplot, 3, 12, 13, 13
gjamOrdination, 3, 15
gjamPlot, 3, 4, 8, 16
gjamPoints2Grid, 3, 18
gjamPredict, 3, 20
gjamPriorTemplate, 3, 23
gjamReZero, 3, 10, 25, 25
gjamSensitivity, 3, 26
gjamSimData, 3, 4, 6, 8, 21, 27, 27, 32
gjamSpec2Trait, 3, 30
gjamTrimY, 3, 31

par, 17
print.gjam (gjam), 4

summary.gjam (gjam), 4