Package 'dalmatian'

September 13, 2020

Title Automating the Fitting of Double Linear Mixed Models in 'JAGS' and 'nimble'

Version 0.6.1

Description Automates fitting of double GLM in 'JAGS'. Includes automatic generation of 'JAGS' scripts, running 'JAGS' or 'nimble' via the 'rjags' and 'nimble' package, and summarizing the resulting output.

Depends R (>= 3.5.0)

License GPL-2

LazyData true

VignetteBuilder knitr

Imports coda, ggmcmc, dglm, tidyr, dplyr, rlang

Suggests rmarkdown, rjags, nimble, knitr, ggplot2, parallel, dclone

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

Author Simon Bonner [aut, cre] (<https://orcid.org/0000-0003-2063-4572>), Hanna Kim [aut]

Maintainer Simon Bonner <simon.bonner@uwo.ca>

Repository CRAN

Date/Publication 2020-09-13 21:30:10 UTC

R topics documented:

betabin_data_1 2
caterpillar 3
caterpillar.dalmatian 3
coef.dalmatian 4
convergence 5
convergence.dalmatian 5
dalmatian 6
gamma_data_1 9

nbinom_data_1 10
pfdata 10
plot.dalmatian 11
predict.dalmatian 12
print.dalmatian 13
print.dalmatian.summary 14
ranef 14
ranef.dalmatian 15
residuals.dalmatian 15
setJAGSInits 17
summary.dalmatian 18
terms.dalmatian 19
traceplots 20
traceplots.dalmatian 20
weights_data_1 21

Index **22**

`betabin_data_1` *Simulated data for illustrating the beta-binomial model*

Description

Simulated data to show how the beta-binomial model may be fit with fixed and random effects on both the mean and dispersion.

Usage

```
betabin_data_1
```

Format

A data frame containing 500 observations and 6 columns:

ID The individual ID.

Rep The replicate number.

x1 The value of the covariate for the mean.

x2 The value of the covariate for the dispersion.

m The number of Bernoulli trials for each observation.

y The number of successes.

caterpillar *Caterpillar (Generic)*

Description

Caterpillar (Generic)

Usage

```
caterpillar(object, ...)
```

Arguments

object	Object to assess.
...	Ignored

caterpillar.dalmatian *Caterpillar (dalmatian)*

Description

Construct caterpillar plots for key (or selected) parameters in a dalmatian object.

Usage

```
## S3 method for class 'dalmatian'
caterpillar(
  object,
  family = NULL,
  nstart = start(object$coda),
  nend = end(object$coda),
  nthin = thin(object$coda),
  show = TRUE,
  return_plots = TRUE,
  ...
)
```

Arguments

object	Object of class dalmatian created by dalmatian().
family	String defining selected family of variables (see help for ggs()).
nstart	Start point for computing summary statistics (relative to true start of chain).
nend	End point for computing summary statistics (relative to true start of chain).

nthin	Thinning factor for computing summary statsitics (relative to full chain and not previously thinned output).
show	If TRUE then plots are displayed on the computer screen and the session is paused between each plot.
return_plots	If TRUE then a named list of ggplot objects containing the plots will be returned as output.
...	Ignored

Value

A list of ggplot objects that can be used to later reproduce the plots via print.

---

coef.dalmatian *Coefficients function for* dalmatian *objects*

Description

coef (dalmatian)

Usage

```
## S3 method for class 'dalmatian'
coef(object, summary = NULL, ranef = NULL, ...)
```

Arguments

object	Object of class dalmatian created by dalmatian().
summary	Posterior summaries computed from the supplied dalmatian object (optional).
ranef	Random effects summary computed from the supplied dalmatian object (optional).
...	Ignored

Details

Extracts coefficients for the mean and dispersion components of a dalmatian model.

Value

List of three lists named mean, dispersion, and joint each containing the posterior means of the coefficients corresponding to the fixed and random terms of that model component (if present).

Author(s)

Simon Bonner

convergence *Convergence Diagnostics (S3 Generic)*

Description

Generic function for computing convergence diagnostics.

Usage

```
convergence(object, ...)
```

Arguments

object	Object to asses.
...	Ignored

convergence.dalmatian *Convergence*

Description

Compute convergence diagnostics for a dalmatian object.

Usage

```
## S3 method for class 'dalmatian'
convergence(
  object,
  pars = NULL,
  nstart = start(object$coda),
  nend = end(object$coda),
  nthin = coda::thin(object$coda),
  raftery = NULL,
  ...
)
```

Arguments

object	Object of class dalmatian created by dalmatian().
pars	List of parameters to assess. If NULL (default) then diagnostics are computed for the fixed effects and random effects standard deviations in the mean, dispersion, and joint components.
nstart	Start point for computing summary statistics (relative to true start of chain).
nend	End point for computing summary statistics (relative to true start of chain).

nthin	Thinning factor for computing summary statsitics (relative to full chain and not previously thinned output).
raftery	List of arguments to be passed to `raftery.diag()`. Any values not provided will be set to their defaults (see `help(raftery.diag())` for details).
...	Ignored

Value

List containing Gelman-Rubin and Raftery convergence diagnostics and effective sampel sizes for the selected parameters.

`dalmatian` *Run DGLM in* JAGS *via* `rjags`

Description

The primary function which automates the running of JAGS.

See vignettes included in the package for full documentation. The list of available vignettes can be generated with `vignette(package="dalmatian")`.

Usage

```
dalmatian(
  df,
  family = "gaussian",
  mean.model,
  dispersion.model,
  joint.model = NULL,
  jags.model.args,
  coda.samples.args,
  response = NULL,
  ntrials = NULL,
  rounding = FALSE,
  lower = NULL,
  upper = NULL,
  parameters = NULL,
  svd = FALSE,
  residuals = FALSE,
  gencode = NULL,
  run.model = TRUE,
  engine = "JAGS",
  n.cores = 1L,
  drop.levels = TRUE,
  drop.missing = TRUE,
  include.checks = TRUE,
  overwrite = FALSE,
```

```
  debug = FALSE,
  saveJAGSinput = NULL
)
```

Arguments

df	Data frame containing the response and predictor values for each individual. (data.frame)
family	Name of family of response distribution. Currently supported families include normal (gaussian) and negative binomial (nbinom). (character)
mean.model	Model list specifying the structure of the mean. (list)
dispersion.model	Model list specifying the structure of the dispersion. (list)
joint.model	Model list specifying structure with parameter shared between linear predictors of the mean and variance. (list)
jags.model.args	List containing named arguments of jags.model. (list)
coda.samples.args	List containing named arguments of coda.samples. (list)
response	Name of variable in the data frame representing the response. (character)
ntrials	Name of variable in the data frame representing the number of independent trials for each observation of the beta binomial model.
rounding	Specifies that response has been rounded if TRUE. (logical)
lower	Name of variable in the data frame representing the lower bound on the response if rounded. (character)
upper	Name of variable in the data frame representing the upper bound on the response if rounded. (character)
parameters	Names of parameters to monitor. If NULL then default values are selected. (character)
svd	Compute Singular Variable Decomposition of model matrices to improve convergence. (logical)
residuals	If TRUE then compute residuals in output. (logical)
gencode	If TRUE then generate code potentially overwriting existing model file. By default generate code if the file does not exist and prompt user if it does. (logical)
run.model	If TRUE then run sampler. Otherwise, stop once code and data have been created. (logical)
engine	Specifies the sampling software. Packages currently supported include JAGS (the default) and nimble. (character)
n.cores	Number of cores to use. If equal to 1 then chains will not be run in parallel. If greater than 1 then chains will be run in parallel using the designated number of cores.
drop.levels	If TRUE then drop unused levels from all factors in df. (logical)
drop.missing	If TRUE then remove records with missing response variable. (logical)

include.checks	If TRUE (default) then include extra Bernoulli variables in the model to ensure that the mean and dispersion parameters remain within their support. (logical)
overwrite	If TRUE then overwrite existing JAGS files (non-interactive sessions only). (logical)
debug	If TRUE then enter debug model. (logical)
saveJAGSinput	Directory to which jags.model input is saved prior to calling jags.model(). This is useful for debugging. No files saved if NULL. (character)

Details

The primary function in the package, dalmatian automates the generation of code, data, and initial values. These are then passed as arguments to function from the rjags package which automates the generation of samplse from the posterior.

Value

An object of class dalmatian contaiining copies of the original data frame, the mean model, the dispersion model the arguments of jags.model and coda.samples. and the output of the MCMC sampler.

Author(s)

Simon Bonner

Examples

```
## Not run:
## Load pied flycatcher data
data(pied_flycatchers_1)

## Create variables bounding the true load
pfdata$lower=ifelse(pfdata$load==0,log(.001),log(pfdata$load-.049))
pfdata$upper=log(pfdata$load+.05)
## Mean model
mymean=list(fixed=list(name="alpha",
                       formula=~ log(IVI) + broodsize + sex,
                       priors=list(c("dnorm",0,.001))))

## Dispersion model
myvar=list(fixed=list(name="psi",
                       link="log",
                       formula=~broodsize + sex,
                       priors=list(c("dnorm",0,.001))))

## Set working directory
## By default uses a system temp directory. You probably want to change this.
workingDir <- tempdir()

## Define list of arguments for jags.model()
jm.args <- list(file=file.path(workingDir,"pied_flycatcher_1_jags.R"),n.adapt=1000)
```

```
## Define list of arguments for coda.samples()
cs.args <- list(n.iter=5000)

## Run the model using dalmatian
pfresults <- dalmatian(df=pfdata,
                         mean.model=mymean,
                         dispersion.model=myvar,
                         jags.model.args=jm.args,
                         coda.samples.args=cs.args,
                         rounding=TRUE,
                         lower="lower",
                         upper="upper",
                         debug=FALSE)

## End(Not run)
```

---

gamma_data_1 *Simulated data to show how the gamma model may be fit with fixed and random effects on both the mean and dispersion.*

Description

Simulated data to show how the gamma model may be fit with fixed and random effects on both the mean and dispersion.

Usage

```
gamma_data_1
```

Format

A data frame containing 1500 observations and 5 columns:

ID The individual ID.

Rep The replicate number.

x1 The value of the covariate for the mean.

x2 The value of the covariate for the dispersion.

y The response.

nbinom_data_1 — *Simulated data to show how the negative binomial model may be fit with fixed and random effects on both the mean and dispersion.*

Description

Simulated data to show how the negative binomial model may be fit with fixed and random effects on both the mean and dispersion.

Usage

```
nbinom_data_1
```

Format

A data frame containing 1500 observations and 5 columns:

ID The individual ID.

Rep The replicate number.

x1 The value of the covariate for the mean.

x2 The value of the covariate for the dispersion.

y The count.

pfdata — *Pied flycatcher feeding data*

Description

Dataset containing 5795 records of 60 pied flycatchers from 33 nest boxes feeding their nestlings during a brood manipulation experiment.

Usage

```
pfdata
```

Format

A data frame containing 5795 rows and 17 variables

plot.dalmatian *Plot Function for* dalmatian *objects*

Description

Create traceplots and caterpillar plots from output of the fitted model.

Usage

```
## S3 method for class 'dalmatian'
plot(
  x,
  trace = TRUE,
  caterpillar = TRUE,
  show = TRUE,
  return_plots = FALSE,
  ...
)
```

Arguments

x	Object of class dalmatian created by dalmatian().
trace	If TRUE (default) then generate traceplots.
caterpillar	If TRUE (default) then generate caterpillar plots
show	If TRUE (default) then display plots as they are generated.
return_plots	If TRUE (not default) return a list of ggplot objects representing the plots.
...	Ignored

Details

This function is a wrapper for the functions traceplots.dalmatian() and caterpillar.dalmatian() which create traceplots and caterpillar plots of all variables stored by the sampler. Further control is available by calling these functions directly.

Value

List of ggplot objects if return_plots is true.

Author(s)

Simon Bonner

Examples

```
## Not run:
## Plot results for pied-flycatcher model without random effects
plot(pfresults)

## Plot results for pied-flycatcher model with random effects
plot(pfresults2)

## End(Not run)
```

`predict.dalmatian` *Prediction method for dalmatian objects*

Description

Prediction method for dalmatian objects

Usage

```
## S3 method for class 'dalmatian'
predict(
  object,
  newdata = object$df,
  method = "mean",
  population = FALSE,
  se = TRUE,
  ci = TRUE,
  type = c("link", "response"),
  level = c(0.5, 0.95),
  ...
)
```

Arguments

`object`	Object of class `dalmatian` created by `dalmatian()`.
`newdata`	data frame containing predictor values to predict response variables. Defaults to data in `object` if not supplied. (data.frame)
`method`	Method to construct the fitted model. Either posterior mean (`"mean"`) or posterior mode (`"mode"`) (character)
`population`	If TRUE then generate predictions at the population level rather than the individual level. (logical)
`se`	if TRUE return the posterior standard devition (logical)
`ci`	returning credible intervals for predictions if TRUE (logical)

type	The type of prediction required (as in predict() for models generated by glm()). The default is on the scale of the linear predictors; the alternative '"response"' is on the scale of the response variable. E.g., if the link between the mean and its linear predictor is the logit function then the default prediction for the mean will be on the scale of the log-odds. If the link between the mean and its linear predictor is the log funciton then the defults prediction will be on the scale of the log.
level	vector of levels of credible intervals for predictions (numeric)
...	Ignored

Value

predictions (list)

---

print.dalmatian *Printed Summary of a* dalmatian *Object*

Description

Prints summary information about a fitted model of class dalmatian.

Usage

```
## S3 method for class 'dalmatian'
print(x, summary = TRUE, convergence = TRUE, ...)
```

Arguments

x	Object of class dalmatian created by dalmatian().
summary	If TRUE (default) compute posterior summary statistics via summary.dalmatian().
convergence	If TRUE (default) compute MCMC convergence diagnostics via convergence().
...	Ignored

Details

This function produces a description of the model's structure and (by default) computes and prints the summary statistics computed via summary.dalmatian() and the MCMC convergence diagnostics computed via convergence.dalmatian(). Further control is available by calling these functions directly.

Value

List of two elements containing posterior summary statstics and convergence diagnostics (if requested).

Author(s)

Simon Bonner

Examples

```
## Not run:
## Print summary of dalmatian objects
print(pfresults)
print(pfresults2)

## End(Not run)
```

---

print.dalmatian.summary *Print Summary (dalmatian)*

Description

Print Summary (dalmatian)

Usage

```
## S3 method for class 'dalmatian.summary'
print(x, digits = 2, ...)
```

Arguments

x	Object of class dalamtion.summary created by summary.dalmatian().
digits	Number of digits to display after decimal.
...	Ignored

---

ranef *Random Effects (S3 Generic)*

Description

Generic function for exporting summaries of random effects.

Usage

```
ranef(object, ...)
```

Arguments

object	Input object
...	Ignored

ranef.dalmatian *Random Effects (dalmatian)*

Description

Compute posterior summary statistics for the individual random effects in each part of the model.

Usage

```
## S3 method for class 'dalmatian'
ranef(
  object,
  nstart = start(object$coda),
  nend = end(object$coda),
  nthin = thin(object$coda),
  ...
)
```

Arguments

object	Object of class dalmatian created by dalmatian().
nstart	Start point for computing summary statistics (relative to true start of chain).
nend	End point for computing summary statistics (relative to true start of chain).
nthin	Thinning factor for computing summary statsitics (relative to full chain and not previously thinned output).
...	Ignored

Value

output (list)

residuals.dalmatian *Residuals method for dalmatian fitted objects*

Description

Computes posterior summaries of the residuals for each observation. Summary statistics include the posterior mean and the upper and lower bounds of the 95 If the response is not rounded then the residuals can either be sampled as part of the MCMC or computed during post-processing. If computed as part of the MCMC then residuals() will simply summarize the posterio distributions. Otherwise, residuals() will compute the residuals and their posterior summaries. If the response is rounded then the residuals must be sampled when the MCMC sampler is run.

Usage

```
## S3 method for class 'dalmatian'
residuals(object, ...)
```

Arguments

object	Object of class dalmatian created by dalmatian().
...	Ignored

Value

Data frame containing original data augmented with posterior mean and lower and upper bounds of the 95 residual for each observation.

Author(s)

Simon Bonner

Examples

```
## Not run:
## Here we rerun the first example in
## \code{vignettes(pied-flycatcher-1)} with \code{residuals = TRUE}
## in order to sample the residuals and then use the \code{residuals()}
## function to summarize the posterior distributions. This is necessary
## because the output is too large to store inside the package.

## Load pied flycatcher data
data(pied_flycatchers_1)


## Create variables bounding the true load
pfdata$lower=ifelse(pfdata$load==0,log(.001),log(pfdata$load-.049))
pfdata$upper=log(pfdata$load+.05)

##### Model 1 #####

## Mean model
mymean=list(fixed=list(name="alpha",
       formula=~ log(IVI) + broodsize + sex,
       priors=list(c("dnorm",0,.001))))

## Dispersion model
mydisp=list(fixed=list(name="psi",
      link="log",
      formula=~broodsize + sex,
      priors=list(c("dnorm",0,.001))))


## Set working directory
workingDir <- tempdir()
```

```
## Define list of arguments for jags.model()
jm.args <- list(file=file.path(workingDir,"pied_flycatcher_1_jags.R"),n.adapt=1000)

## Define list of arguments for coda.samples()
cs.args <- list(n.iter=5000,thin=20)

## Run the model using dalmatian
pfresults <- dalmatian(df=pfdata,
                       mean.model=mymean,
                       dispersion.model=mydisp,
                       jags.model.args=jm.args,
                       coda.samples.args=cs.args,
                       rounding=TRUE,
                       lower="lower",
                       upper="upper",
                       n.cores = 3,
                       residuals = TRUE,
                       overwrite = TRUE,
                       debug=FALSE)

## summarize residuals
res.pfresults <- residuals(object = pfresults)

## End(Not run)
```

`setJAGSInits` *Set initial values for* `dalmatian`

Description

Set initial values for `dalmatian`

Usage

```
setJAGSInits(
  mean.model,
  dispersion.model,
  fixed.mean = NULL,
  fixed.dispersion = NULL,
  y = NULL,
  random.mean = NULL,
  sd.mean = NULL,
  random.dispersion = NULL,
  sd.dispersion = NULL
)
```

Arguments

`mean.model`	Model list specifying the structure of the mean. (list)
`dispersion.model`	Model list specifyint the structure of the dispersion. (list)
`fixed.mean`	Initial values for the fixed effects of the mean. (numeric)
`fixed.dispersion`	Initial values for the fixed effects of the dispersion. (numeric)
`y`	Inital values for the true response. This should only be specified if the `rounding = TRUE` in the main call to dalmatian.
`random.mean`	Initial values for the random effects of the mean. (numeric)
`sd.mean`	Initial values for the standard deviation of the random effects of the mean. (numeric)
`random.dispersion`	Initial values for the random effects of the dispersion. (numeric
`sd.dispersion`	Initial values for the standard deviation of the random effects of the dispersion. (numeric)

Details

Allows the user to set initial values for `dalmatian`. Any values not specified will by initialized by `JAGS`.

Value

inits (list)

Author(s)

Simon Bonner

`summary.dalmatian` *Summary (dalmatian)*

Description

Summary (dalmatian)

Usage

```
## S3 method for class 'dalmatian'
summary(
  object,
  nstart = start(object$coda),
  nend = end(object$coda),
  nthin = thin(object$coda),
  ...
)
```

Arguments

object	Object of class dalmatian created by dalmatian().
nstart	Start point for computing summary statistics (relative to true start of chain).
nend	End point for computing summary statistics (relative to true start of chain).
nthin	Thinning factor for computing summary statsitics (relative to full chain and not previously thinned output).
...	Ignored

Value

output (list)

---

terms.dalmatian *Terms function for* dalmatian *objects*

Description

terms (dalmatian)

Usage

```
## S3 method for class 'dalmatian'
terms(x, ...)
```

Arguments

x	Object of class dalmatian created by dalmatian().
...	Further object passed directly to terms. Recycled for each model component.

Details

Constructs a list of terms objects for each component of the model specified in the input object.

Value

List of with two lists named mean and dispersion each containing terms objects corresponding to the fixed and random components of that model component (if present).

Author(s)

Simon Bonner

traceplots *Traceplots (Generic)*

Description

Traceplots (Generic)

Usage

```
traceplots(object, ...)
```

Arguments

object	Object to assess.
...	Ignored

traceplots.dalmatian *Traceplots (dalmatian)*

Description

Construct traceplots for key (or selected) parameters in a dalmatian object.

Usage

```
## S3 method for class 'dalmatian'
traceplots(
  object,
  family = NULL,
  nstart = start(object$coda),
  nend = end(object$coda),
  nthin = thin(object$coda),
  show = TRUE,
  return_plots = TRUE,
  ...
)
```

Arguments

object	Object of class dalmatian created by dalmatian().
family	String defining selected family of variables (see help for ggs()).
nstart	Start point for computing summary statistics (relative to true start of chain).
nend	End point for computing summary statistics (relative to true start of chain).

nthin	Thinning factor for computing summary statsitics (relative to full chain and not previously thinned output).
show	If TRUE then plots are displayed on the computer screen and the session is paused between each plot.
return_plots	If TRUE then return list of ggplot objects.
...	Ignored

Value

A list of ggplot objects that can be used to later reproduce the plots via print.

---

weights_data_1 *Simulated data for illustrating the use of weights*

Description

Simulated data for illustrating the use of weights in the particular case when the responses are averages of observed with different denominators

Usage

```
weights_data_1
```

Format

An object of class data.frame with 100 rows and 3 columns.

Details

@format A data frame with 100 rows and 3 columns:

n The number of observations.

x The common predictor value.

y The mean response value.

Index

∗ **datasets**
betabin_data_1, 2
gamma_data_1, 9
nbinom_data_1, 10
pfdata, 10
weights_data_1, 21

betabin_data_1, 2

caterpillar, 3
caterpillar.dalmatian, 3
coef.dalmatian, 4
convergence, 5
convergence.dalmatian, 5

dalmatian, 6

gamma_data_1, 9

nbinom_data_1, 10

pfdata, 10
plot.dalmatian, 11
predict.dalmatian, 12
print.dalmatian, 13
print.dalmatian.summary, 14

ranef, 14
ranef.dalmatian, 15
residuals.dalmatian, 15

setJAGSInits, 17
summary.dalmatian, 18

terms.dalmatian, 19
traceplots, 20
traceplots.dalmatian, 20

weights_data_1, 21