Package 'FBMS'

July 21, 2025

Type Package

Title Flexible Bayesian Model Selection and Model Averaging

Version 1.1

Date 2025-02-26

Encoding UTF-8

Language en-US

Description Implements the Mode Jumping Markov Chain Monte Carlo algorithm described in <doi:10.1016/j.csda.2018.05.020> and its Genetically Modified counterpart described in <doi:10.1613/jair.1.13047> as well as the sub-sampling versions described in <doi:10.1016/j.ijar.2022.08.018> for flexible Bayesian model selection and model averaging.

License GPL-2

Depends R (>= 3.5.0), fastglm, GenSA, parallel, methods, stats, graphics, r2r

Imports Rcpp

LinkingTo Rcpp

Suggests testthat, knitr, rmarkdown, markdown

RoxygenNote 7.3.2

VignetteBuilder knitr

LazyData true

NeedsCompilation yes

Author Jon Lachmann [cre, aut], Aliaksandr Hubin [aut]

Maintainer Jon Lachmann <jon@lachmann.nu>

Repository CRAN

Date/Publication 2025-02-26 21:00:02 UTC

Contents

`FBMS-package` *Flexible Bayesian Model Selection and Model Averaging*

Description

Implements the Mode Jumping Markov Chain Monte Carlo algorithm described in <doi:10.1016/j.csda.2018.05.020> and its Genetically Modified counterpart described in <doi:10.1613/jair.1.13047> as well as the sub-sampling versions described in <doi:10.1016/j.ijar.2022.08.018> for flexible Bayesian model selection and model averaging.

Author(s)

Maintainer: Jon Lachmann <jon@lachmann.nu>

Authors:

- Jon Lachmann <jon@lachmann.nu>
- Aliaksandr Hubin <aliaksah@math.uio.no>

Other contributors:

- Florian Frommlet <florian.frommlet@meduniwien.ac.at> [contributor]
- Geir Storvik <geirs@math.uio.no> [contributor]

References

Lachmann, J., Storvik, G., Frommlet, F., & Hubin, A. (2022). A subsampling approach for Bayesian model selection. International Journal of Approximate Reasoning, 151, 33-63. Elsevier.

Hubin, A., Storvik, G., & Frommlet, F. (2021). Flexible Bayesian Nonlinear Model Configuration. Journal of Artificial Intelligence Research, 72, 901-942.

Hubin, A., Frommlet, F., & Storvik, G. (2021). Reversible Genetically Modified MJMCMC. Under review in EYSM 2021.

Hubin, A., & Storvik, G. (2018). Mode jumping MCMC for Bayesian variable selection in GLMM. Computational Statistics & Data Analysis, 127, 281-297. Elsevier.

`abalone` *Physical measurements of 4177 abalones, a species of sea snail.*

Description

%% ~~ A concise (1-5 lines) description of the dataset. ~~

Format

A data frame with 4177 observations on the following 9 variables.

Diameter Diameter Perpendicular to length, continuous

Height Height with with meat in shell, continuous.

Length Longest shell measurement, continuous

Rings +1.5 gives the age in years, integer

Sex Sex of the abalone, `F` is female, `M` male, and `I` infant, categorical.

Weight_S Grams after being dried, continuous.

Weight_Sh Grams weight of meat, continuous.

Weight_V Grams gut weight (after bleeding), continuous.

Weight_W Grams whole abalone, continuous.

Details

See the web page https://archive.ics.uci.edu/ml/datasets/Abalone for more information about the data set.

Source

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository https://archive.ics.uci.edu/ml/. Irvine, CA: University of California, School of Information and Computer Science.

Examples

```
data(abalone)
## maybe str(abalone) ; plot(abalone) ...
```

---

breastcancer *Breast Cancer Wisconsin (Diagnostic) Data Set*

Description

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Usage

```
data(breastcancer)
```

Format

A data frame with 569 rows and 32 variables

Details

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) (K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992), a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: (K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34).

The variables are as follows:

- ID number
- Diagnosis (1 = malignant, 0 = benign)
- Ten real-valued features are computed for each cell nucleus

Source

Dataset downloaded from the UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Creators:

1. Dr. William H. Wolberg, General Surgery Dept. University of Wisconsin, Clinical Sciences Center Madison, WI 53792 wolberg 'at' eagle.surgery.wisc.edu
2. W. Nick Street, Computer Sciences Dept. University of Wisconsin, 1210 West Dayton St., Madison, WI 53706 street 'at' cs.wisc.edu 608-262-6619
3. Olvi L. Mangasarian, Computer Sciences Dept. University of Wisconsin, 1210 West Dayton St., Madison, WI 53706 olvi 'at' cs.wisc.edu

Donor: Nick Street

References

W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.

Lichman, M. (2013). UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

---

compute_effects *Compute effects for specified in labels covariates using a fitted model.*

Description

This function computes model averaged effects for specified covariates using a fitted model object. The effects are expected change in the BMA linear predictor having an increase of the corresponding covariate by one unit, while other covariates are fixed to 0. Users can provide custom labels and specify quantiles for the computation of effects.

Usage

```
compute_effects(object, labels, quantiles = c(0.025, 0.5, 0.975))
```

Arguments

object	A fitted model object, typically the result of a regression or predictive modeling.
labels	A vector of labels for which effects are to be computed.
quantiles	A numeric vector specifying the quantiles to be calculated. Default is c(0.025, 0.5, 0.975).

Value

A matrix of treatment effects for the specified labels, with rows corresponding to labels and columns to quantiles.

See Also

predict

Examples

```
data <- data.frame(matrix(rnorm(600), 100))
result <- mjmcmc.parallel(runs = 2, cores = 1, data, gaussian.loglik)
compute_effects(result,labels = names(data)[-1])
```

---

cos_deg *Cosine function for degrees*

Description

Cosine function for degrees

Usage

```
cos_deg(x)
```

Arguments

x	The vector of values in degrees

Value

The cosine of x

Examples

```
cos_deg(0)
```

---

diagn_plot *Plot convergence of best/median/mean/other summary log posteriors in time*

Description

Plot convergence of best/median/mean/other summary log posteriors in time

Usage

```
diagn_plot(res, FUN = median, conf = 0.95, burnin = 0, window = 5, ylim = NULL)
```

Arguments

res	Object corresponding gmjmcmc output
FUN	The summary statistics to check convergence
conf	which confidence intervals to plot
burnin	how many first populations to skip
window	sliding window for computing the standard deviation
ylim	limits for the plotting range, if unspecified, min and max of confidence intervals will be used

Value

A list of summary statistics for checking convergence with given confidence intervals

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
diagnstats <- diagn_plot(result)
```

---

erf *erf function*

Description

erf function

Usage

```
erf(x)
```

Arguments

x	The vector of values

Value

2 * pnorm(x * sqrt(2)) - 1

Examples

```
erf(2)
```

`exoplanet` *Excerpt from the Open Exoplanet Catalogue data set*

Description

Data fields include planet and host star attributes.

Usage

```
data(exoplanet)
```

Format

A data frame with 223 rows and 11 variables

Details

The variables are as follows:

- TypeFlag: Flag indicating the type of data
- PlanetaryMassJpt: Mass of the planetary object in Jupiter masses
- RadiusJpt: Radius of the planetary object in Jupiter radii
- PeriodDays: Orbital period of the planetary object in days
- SemiMajorAxisAU: Semi-major axis of the planetary object's orbit in astronomical units
- Eccentricity: Eccentricity of the planetary object's orbit
- HostStarMassSlrMass: Mass of the host star in solar masses
- HostStarRadiusSlrRad: Radius of the host star in solar radii
- HostStarMetallicity: Metallicity of the host star
- HostStarTempK: Effective temperature of the host star in Kelvin
- PlanetaryDensJpt: Density of the planetary object up to a constant

Source

Dataset downloaded from the Open Exoplanet Catalogue Repository. https://github.com/OpenExoplanetCatalogue/oec_tables/

Creators:

1. Prof. Hanno Rein, Department for Physical and Environmental Sciences. University of Toronto at Scarborough Toronto, Ontario M1C 1A4 hanno.rein 'at' utoronto.ca

exp_dbl *Double exponential function*

Description

Double exponential function

Usage

```
exp_dbl(x)
```

Arguments

x	The vector of values

Value

e^(-abs(x))

Examples

```
exp_dbl(2)
```

fbms *Fit a BGNLM model using Genetically Modified Mode Jumping Markov Chain Monte Carlo (MCMC) sampling. Or Fit a BGLM model using Modified Mode Jumping Markov Chain Monte Carlo (MCMC) sampling.*

Description

This function fits a model using the relevant MCMC sampling. The user can specify the formula, family, data, transforms, and other parameters to customize the model.

Usage

```
fbms(
  formula = NULL,
  family = "gaussian",
  data = NULL,
  impute = FALSE,
  loglik.pi = gaussian.loglik,
  method = "mjmcmc",
  verbose = TRUE,
  ...
)
```

Arguments

formula	A formula object specifying the model structure. Default is NULL.
family	The distribution family of the response variable. Currently supports "gaussian", "binomial" and "custom". Default is "gaussian".
data	A data frame containing the variables in the model. If NULL, the variables are taken from the environment of the formula. Default is NULL.
impute	TRUE means imputation combined with adding a dummy column with indicators of imputed values, FALSE (default) means only full data is used.
loglik.pi	Custom function to compute the logarithm of the posterior mode based on logarithm of marginal likelihood and logarithm of prior functions (needs specification only used if family = "custom")
method	Which fitting algorithm should be used, currently implemented options include "gmjmcmc", "gmjmcmc.parallel", "mjmcmc" and "mjmcmc.parallel" with "mjmcmc" being the default and 'mjmcmc' means that only linear models will be estimated
verbose	If TRUE, print detailed progress information during the fitting process. Default is TRUE.
...	Additional parameters to be passed to the underlying method.

Value

An object containing the results of the fitted model and MCMC sampling.

See Also

mjmcmc, gmjmcmc, gmjmcmc.parallel

Examples

```
# Fit a Gaussian multivariate time series model
fbms_result <- fbms(
 X1 ~ .,
 family = "gaussian",
 method = "gmjmcmc.parallel",
 data = data.frame(matrix(rnorm(600), 100)),
 transforms = c("sin","cos"),
 P = 10,
 runs = 1,
 cores = 1
)
summary(fbms_result)
plot(fbms_result)
```

gauss *Gaussian function*

Description

Gaussian function

Usage

```
gauss(x)
```

Arguments

x	The vector of values

Value

e^(-x^2)

Examples

```
gauss(2)
```

gaussian.loglik *Log likelihood function for gaussian regression with a prior p(m)=r*sum(total_width).*

Description

Log likelihood function for gaussian regression with a prior p(m)=r*sum(total_width).

Usage

```
gaussian.loglik(y, x, model, complex, params)
```

Arguments

y	A vector containing the dependent variable
x	The matrix containing the precalculated features
model	The model to estimate as a logical vector
complex	A list of complexity measures for the features
params	A list of parameters for the log likelihood, supplied by the user

Value

A list with the log marginal likelihood combined with the log prior (crit) and the posterior mode of the coefficients (coefs).

Examples

```
gaussian.loglik(rnorm(100), matrix(rnorm(100)), TRUE, list(oc = 1), NULL)
```

`gaussian.loglik.alpha` *Log likelihood function for gaussian regression for alpha calculation This function is just the bare likelihood function Note that it only gives a proportional value and is equivalent to least squares*

Description

Log likelihood function for gaussian regression for alpha calculation This function is just the bare likelihood function Note that it only gives a proportional value and is equivalent to least squares

Usage

```
gaussian.loglik.alpha(a, data, mu_func)
```

Arguments

`a`	A vector of the alphas to be used
`data`	The data to be used for calculation
`mu_func`	The function linking the mean to the covariates, as a string with the alphas as a[i].

Value

A numeric with the log likelihood.

Examples

```
## Not run:
gaussian.loglik.alpha(my_alpha,my_data,my_mu)

## End(Not run)
```

`gelu` *GELU function*

Description

GELU function

Usage

```
gelu(x)
```

Arguments

`x`	The vector of values

Value

x*pnorm(x)

Examples

```
gelu(2)
```

`gen.params.gmjmcmc` *Generate a parameter list for GMJMCMC (Genetically Modified MJMCMC)*

Description

This function generates the full list of parameters required for the Generalized Mode Jumping Markov Chain Monte Carlo (GMJMCMC) algorithm, building upon the parameters from `gen.params.mjmcmc`. The generated parameter list includes feature generation settings, population control parameters, and optimization controls for the search process.

Usage

```
gen.params.gmjmcmc(data)
```

Arguments

`data`	A data frame containing the dataset with covariates and response variable.

Value

A list of parameters for controlling GMJMCMC behavior:

Feature Generation Parameters (`feat`)

`feat$D` Maximum feature depth, default `5`. Limits the number of recursive feature transformations. For fractional polynomials, it is recommended to set `D = 1`.

`feat$L` Maximum number of features per model, default `15`. Increase for complex models.

`feat$alpha` Strategy for generating $alpha$ parameters in non-linear projections:

- `"unit"` (Default) Sets all components to 1.
- `"deep"` Optimizes $alpha$ across all feature layers.
- `"random"` Samples $alpha$ from the prior for a fully Bayesian approach.

`feat$pop.max` Maximum feature population size per iteration. Defaults to `min(100, as.integer(1.5 * p))`, where `p` is the number of covariates.

`feat$keep.org` Logical flag; if `TRUE`, original covariates remain in every population (default `FALSE`).

`feat$prel.filter` Threshold for pre-filtering covariates before the first population generation. Default `0` disables filtering.

`feat$prel.select` Indices of covariates to include initially. Default `NULL` includes all.

`feat$keep.min` Minimum proportion of features to retain during population updates. Default `0.8`.

`feat$eps` Threshold for feature inclusion probability during generation. Default `0.05`.

`feat$check.col` Logical; if `TRUE` (default), checks for collinearity during feature generation.

`feat$max.proj.size` Maximum number of existing features used to construct a new one. Default `15`.

Scaling Option

`rescale.large` Logical flag for rescaling large data values for numerical stability. Default `FALSE`.

MJMCMC Parameters

`burn_in` The burn-in period for the MJMCMC algorithm, which is set to 100 iterations by default.

`mh` A list containing parameters for the regular Metropolis-Hastings (MH) kernel:

- `neigh.size` The size of the neighborhood for MH proposals with fixed proposal size, default set to 1.
- `neigh.min` The minimum neighborhood size for random proposal size, default set to 1.
- `neigh.max` The maximum neighborhood size for random proposal size, default set to 2.

`large` A list containing parameters for the large jump kernel:

- `neigh.size` The size of the neighborhood for large jump proposals with fixed neighborhood size, default set to the smaller of `0.35 * p` and `35`, where p is the number of covariates.
- `neigh.min` The minimum neighborhood size for large jumps with random size of the neighborhood, default set to the smaller of `0.25 * p` and `25`.
- `neigh.max` The maximum neighborhood size for large jumps with random size of the neighborhood, default set to the smaller of `0.45 * p` and `45`.

`random` A list containing a parameter for the randomization kernel:

- `prob` The small probability of changing the component around the mode, default set to 0.01.

`sa` A list containing parameters for the simulated annealing kernel:

probs A numeric vector of length 6 specifying the probabilities for different types of proposals in the simulated annealing algorithm.

neigh.size The size of the neighborhood for the simulated annealing proposals, default set to 1.

neigh.min The minimum neighborhood size, default set to 1.

neigh.max The maximum neighborhood size, default set to 2.

t.init The initial temperature for simulated annealing, default set to 10.

t.min The minimum temperature for simulated annealing, default set to 0.0001.

dt The temperature decrement factor, default set to 3.

M The number of iterations in the simulated annealing process, default set to 12.

greedy A list containing parameters for the greedy algorithm:

probs A numeric vector of length 6 specifying the probabilities for different types of proposals in the greedy algorithm.

neigh.size The size of the neighborhood for greedy algorithm proposals, set to 1.

neigh.min The minimum neighborhood size for greedy proposals, set to 1.

neigh.max The maximum neighborhood size for greedy proposals, set to 2.

steps The number of steps for the greedy algorithm, set to 20.

tries The number of tries for the greedy algorithm, set to 3.

loglik A list to store log-likelihood values, which is by default empty.

See Also

gen.params.mjmcmc, gmjmcmc

Examples

```
data <- data.frame(y = rnorm(100), x1 = rnorm(100), x2 = rnorm(100))
params <- gen.params.gmjmcmc(data)
str(params)
```

---

gen.params.mjmcmc *Generate a parameter list for MJMCMC (Mode Jumping MCMC)*

Description

Generate a parameter list for MJMCMC (Mode Jumping MCMC)

Usage

```
gen.params.mjmcmc(data)
```

Arguments

data	The dataset that will be used in the algorithm

Value

A list of parameters to use when running the mjmcmc function.

The list contains the following elements:

`burn_in` The burn-in period for the MJMCMC algorithm, which is set to 100 iterations by default.

`mh` A list containing parameters for the regular Metropolis-Hastings (MH) kernel:

- `neigh.size` The size of the neighborhood for MH proposals with fixed proposal size, default set to 1.
- `neigh.min` The minimum neighborhood size for random proposal size, default set to 1.
- `neigh.max` The maximum neighborhood size for random proposal size, default set to 2.

`large` A list containing parameters for the large jump kernel:

- `neigh.size` The size of the neighborhood for large jump proposals with fixed neighborhood size, default set to the smaller of 0.35 $\times p$ and 35, where p is the number of covariates.
- `neigh.min` The minimum neighborhood size for large jumps with random size of the neighborhood, default set to the smaller of 0.25 $\times p$ and 25.
- `neigh.max` The maximum neighborhood size for large jumps with random size of the neighborhood, default set to the smaller of 0.45 $\times p$ and 45.

`random` A list containing a parameter for the randomization kernel:

- `prob` The small probability of changing the component around the mode, default set to 0.01.

`sa` A list containing parameters for the simulated annealing kernel:

- `probs` A numeric vector of length 6 specifying the probabilities for different types of proposals in the simulated annealing algorithm.
- `neigh.size` The size of the neighborhood for the simulated annealing proposals, default set to 1.
- `neigh.min` The minimum neighborhood size, default set to 1.
- `neigh.max` The maximum neighborhood size, default set to 2.
- `t.init` The initial temperature for simulated annealing, default set to 10.
- `t.min` The minimum temperature for simulated annealing, default set to 0.0001.
- `dt` The temperature decrement factor, default set to 3.
- `M` The number of iterations in the simulated annealing process, default set to 12.

`greedy` A list containing parameters for the greedy algorithm:

- `probs` A numeric vector of length 6 specifying the probabilities for different types of proposals in the greedy algorithm.
- `neigh.size` The size of the neighborhood for greedy algorithm proposals, set to 1.
- `neigh.min` The minimum neighborhood size for greedy proposals, set to 1.
- `neigh.max` The maximum neighborhood size for greedy proposals, set to 2.
- `steps` The number of steps for the greedy algorithm, set to 20.
- `tries` The number of tries for the greedy algorithm, set to 3.

`loglik` A list to store log-likelihood values, which is by default empty.

Note that the `$loglik` item is an empty list, which is passed to the log likelihood function of the model, intended to store parameters that the estimator function should use.

Examples

```
gen.params.mjmcmc(matrix(rnorm(600), 100))
```

`gen.probs.gmjmcmc` *Generate a probability list for GMJMCMC (Genetically Modified MJMCMC)*

Description

Generate a probability list for GMJMCMC (Genetically Modified MJMCMC)

Usage

```
gen.probs.gmjmcmc(transforms)
```

Arguments

`transforms`	A list of the transformations used (to get the count).

Value

A named list with eight elements:

`large` The probability of a large jump kernel in the MJMCMC algorithm. With this probability, a large jump proposal will be made; otherwise, a local Metropolis-Hastings proposal will be used. One needs to consider good mixing around and between modes when specifying this parameter.

`large.kern` A numeric vector of length 4 specifying the probabilities for different types of large jump kernels. The four components correspond to:

1. Random change with random neighborhood size
2. Random change with fixed neighborhood size
3. Swap with random neighborhood size
4. Swap with fixed neighborhood size

These probabilities will be automatically normalized if they do not sum to 1.

`localopt.kern` A numeric vector of length 2 specifying the probabilities for different local optimization methods during large jumps. The first value represents the probability of using simulated annealing, while the second corresponds to the greedy optimizer. These probabilities will be normalized if needed.

`random.kern` A numeric vector of length 2 specifying the probabilities of first two randomization kernels applied after local optimization. These correspond to the same kernel types as in `large.kern` but are used for local proposals where type and 2 only are allowed.

`mh` A numeric vector specifying the probabilities of different standard Metropolis-Hastings kernels, where the first four as the same as for other kernels, while fifths and sixes components are uniform addition/deletion of a covariate.

`filter` A numeric value controlling the filtering of features with low posterior probabilities in the current population. Features with posterior probabilities below this threshold will be removed with a probability proportional to $1 - P(\text{feature} \mid \text{population})$.

`gen` A numeric vector of length 4 specifying the probabilities of different feature generation operators. These determine how new nonlinear features are introduced. The first entry gives the probability for an interaction, followed by modification, nonlinear projection, and a mutation operator, which reintroduces discarded features. If these probabilities do not sum to 1, they are automatically normalized.

`trans` A numeric vector of length equal to the number of elements in `transforms`, specifying the probabilities of selecting each nonlinear transformation from $\mathcal{G}$. By default, a uniform distribution is assigned, but this can be modified by providing a specific `transforms` argument.

Examples

```
gen.probs.gmjmcmc(c("p0", "exp_dbl"))
```

`gen.probs.mjmcmc` *Generate a probability list for MJMCMC (Mode Jumping MCMC)*

Description

Generate a probability list for MJMCMC (Mode Jumping MCMC)

Usage

```
gen.probs.mjmcmc()
```

Value

A named list with five elements:

`large` A numeric value representing the probability of making a large jump. If a large jump is not made, a local MH (Metropolis-Hastings) proposal is used instead.

`large.kern` A numeric vector of length 4 specifying the probabilities for different types of large jump kernels. The four components correspond to:

1. Random change with random neighborhood size
2. Random change with fixed neighborhood size
3. Swap with random neighborhood size
4. Swap with fixed neighborhood size

These probabilities will be automatically normalized if they do not sum to 1.

`localopt.kern` A numeric vector of length 2 specifying the probabilities for different local optimization methods during large jumps. The first value represents the probability of using simulated annealing, while the second corresponds to the greedy optimizer. These probabilities will be normalized if needed.

`random.kern` A numeric vector of length 2 specifying the probabilities of different randomization kernels applied after local optimization of type one or two. These correspond to the first two kernel types as in `large.kern` but are used for local proposals with different neighborhood sizes.

`mh` A numeric vector specifying the probabilities of different standard Metropolis-Hastings kernels, where the first four as the same as for other kernels, while fifths and sixes components are uniform addition/deletion of a covariate.

Examples

```
gen.probs.mjmcmc()
```

`get.best.model` *Extract the Best Model from MJMCMC or GMJMCMC Results*

Description

This function retrieves the best model from the results of MJMCMC, MJMCMC parallel, GMJMCMC, or GMJMCMC merged runs based on the maximum criterion value (`crit`). The returned list includes the model probability, selected features, criterion value, intercept parameter, and named coefficients.

Usage

```
get.best.model(result, labels = FALSE)
```

Arguments

`result`	An object of class `"mjmcmc"`, `"mjmcmc_parallel"`, `"gmjmcmc"`, or `"gmjmcmc_merged"`, containing the results from the corresponding model search algorithms.
`labels`	Logical; if `TRUE`, uses labeled feature names when naming the model coefficients. Default is `FALSE`.

Details

The function identifies the best model by selecting the one with the highest `crit` value. Selection logic depends on the class of the `result` object:

`"mjmcmc"` Selects the top model from a single MJMCMC run.

`"mjmcmc_parallel"` Identifies the best chain, then selects the best model from that chain.

`"gmjmcmc"` Selects the best population and model within that population.

`"gmjmcmc_merged"` Finds the best chain and population before extracting the top model.

Value

A list containing the details of the best model:

`prob` A numeric value representing the model's probability.

`model` A logical vector indicating which features are included in the best model.

`crit` The criterion value used for model selection (e.g., marginal likelihood or posterior probability).

`alpha` The intercept parameter of the best model.

`coefs` A named numeric vector of model coefficients, including the intercept and selected features.

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
get.best.model(result)
```

`get.mpm.model` *Retrieve the Median Probability Model (MPM)*

Description

This function extracts the Median Probability Model (MPM) from a fitted model object. The MPM includes features with marginal posterior inclusion probabilities greater than 0.5. It constructs the corresponding model matrix and computes the model fit using the specified likelihood.

Usage

```
get.mpm.model(
  result,
  y,
  x,
  labels = F,
  family = "gaussian",
  loglik.pi = gaussian.loglik,
  params = NULL
)
```

Arguments

`result`	A fitted model object (e.g., from `mjmcmc`, `gmjmcmc`, or related classes) containing the summary statistics and marginal probabilities.
`y`	A numeric vector of response values. For `family = "binomial"`, it should contain binary (0/1) responses.
`x`	A `data.frame` of predictor variables. Columns must correspond to features considered during model fitting.

labels	If specified, custom labels of covariates can be used. Default is FALSE.
family	Character string specifying the model family. Supported options are: • "gaussian" (default) - for continuous outcomes. • "binomial" - for binary outcomes. • "custom" - for user-defined likelihood functions. If an unsupported family is provided, a warning is issued and the Gaussian likelihood is used by default.
loglik.pi	A function that computes the log-likelihood. Defaults to gaussian.loglik unless family = "binomial", in which case logistic.loglik is used. for custom family the user must specify the same likelihood that was used in the inference.
params	Parameters of loglik.pi, if not specified NULL will be used by default

Value

A bgnlm_model object containing:

prob The log marginal likelihood of the MPM.

model A logical vector indicating included features.

crit Criterion label set to "MPM".

coefs A named numeric vector of model coefficients, including the intercept.

Examples

```
## Not run:
# Simulate data
set.seed(42)
x <- data.frame(
  PlanetaryMassJpt = rnorm(100),
  RadiusJpt = rnorm(100),
  PeriodDays = rnorm(100)
)
y <- 1 + 0.5 * x$PlanetaryMassJpt - 0.3 * x$RadiusJpt + rnorm(100)

# Assume 'result' is a fitted object from gmjmcmc or mjmcmc
result <- mjmcmc(cbind(y,x))

# Get the MPM
mpm_model <- get.mpm.model(result, y, x, family = "gaussian")

# Access coefficients
mpm_model$coefs

## End(Not run)
```

gmjmcmc *Main algorithm for GMJMCMC (Genetically Modified MJMCMC)*

Description

Main algorithm for GMJMCMC (Genetically Modified MJMCMC)

Usage

```
gmjmcmc(
  data,
  loglik.pi = gaussian.loglik,
  loglik.alpha = gaussian.loglik.alpha,
  transforms,
  P = 10,
  N.init = 100,
  N.final = 100,
  probs = NULL,
  params = NULL,
  sub = FALSE,
  verbose = TRUE
)
```

Arguments

data	A matrix containing the data to use in the algorithm, first column should be the dependent variable, and the rest of the columns should be the independent variables.
loglik.pi	The (log) density to explore
loglik.alpha	The likelihood function to use for alpha calculation
transforms	A Character vector including the names of the non-linear functions to be used by the modification and the projection operator.
P	The number of generations for GMJMCMC (Genetically Modified MJMCMC). The default value is $P = 10$. A larger value like $P = 50$ might be more realistic for more complicated examples where one expects a lot of non-linear structures.
N.init	The number of iterations per population (total iterations = (T-1)*N.init+N.final)
N.final	The number of iterations for the final population (total iterations = (T-1)*N.init+N.final)
probs	A list of the various probability vectors to use
params	A list of the various parameters for all the parts of the algorithm
sub	An indicator that if the likelihood is inexact and should be improved each model visit (EXPERIMENTAL!)
verbose	A logical denoting if messages should be printed

Value

A list containing the following elements:

`models`	All models per population.
`lo.models`	All local optimization models per population.
`populations`	All features per population.
`marg.probs`	Marginal feature probabilities per population.
`model.probs`	Marginal feature probabilities per population.
`model.probs.idx`	Marginal feature probabilities per population.
`best.margs`	Best marginal model probability per population.
`accept`	Acceptance rate per population.
`accept.tot`	Overall acceptance rate.
`best`	Best marginal model probability throughout the run, represented as the maximum value in `unlist(best.margs)`.

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
summary(result)
plot(result)
```

`gmjmcmc.parallel` *Run multiple gmjmcmc (Genetically Modified MJMCMC) runs in parallel returning a list of all results.*

Description

Run multiple gmjmcmc (Genetically Modified MJMCMC) runs in parallel returning a list of all results.

Usage

```
gmjmcmc.parallel(
  runs = 2,
  cores = getOption("mc.cores", 2L),
  merge.options = list(populations = "best", complex.measure = 2, tol = 1e-07),
  data,
  loglik.pi = gaussian.loglik,
  loglik.alpha = gaussian.loglik.alpha,
  transforms,
  ...
)
```

Arguments

runs	The number of runs to run
cores	The number of cores to run on
merge.options	A list of options to pass to the merge_results() function run after the
data	A matrix containing the data to use in the algorithm, first column should be the dependent variable, and the rest of the columns should be the independent variables.
loglik.pi	The (log) density to explore
loglik.alpha	The likelihood function to use for alpha calculation
transforms	A Character vector including the names of the non-linear functions to be used by the modification and the projection operator.
...	Further parameters passed to mjmcmc.

Value

Results from multiple gmjmcmc runs

Examples

```
result <- gmjmcmc.parallel(
  runs = 1,
  cores = 1,
  list(populations = "best", complex.measure = 2, tol = 0.0000001),
  matrix(rnorm(600), 100),
  P = 2,
  gaussian.loglik,
  loglik.alpha = gaussian.loglik.alpha,
  c("p0", "exp_dbl")
)

summary(result)

plot(result)
```

---

hs *heavy side function*

Description

heavy side function

Usage

```
hs(x)
```

Arguments

x	The vector of values

Value

as.integer(x>0)

Examples

```
hs(2)
```

---

linear.g.prior.loglik *Log likelihood function for linear regression using Zellners g-prior*

Description

Log likelihood function for linear regression using Zellners g-prior

Usage

```
linear.g.prior.loglik(y, x, model, complex, params = list(g = 4))
```

Arguments

y	A vector containing the dependent variable
x	The matrix containing the precalculated features
model	The model to estimate as a logical vector
complex	A list of complexity measures for the features
params	A list of parameters for the log likelihood, supplied by the user

Value

A list with the log marginal likelihood combined with the log prior (crit) and the posterior mode of the coefficients (coefs).

Examples

```
linear.g.prior.loglik(rnorm(100), matrix(rnorm(100)), TRUE, list(oc=1))
```

`logistic.loglik` *Log likelihood function for logistic regression with a prior p(m)=sum(total_width) This function is created as an example of how to create an estimator that is used to calculate the marginal likelihood of a model.*

Description

Log likelihood function for logistic regression with a prior p(m)=sum(total_width) This function is created as an example of how to create an estimator that is used to calculate the marginal likelihood of a model.

Usage

```
logistic.loglik(y, x, model, complex, params = list(r = exp(-0.5)))
```

Arguments

`y`	A vector containing the dependent variable
`x`	The matrix containing the precalculated features
`model`	The model to estimate as a logical vector
`complex`	A list of complexity measures for the features
`params`	A list of parameters for the log likelihood, supplied by the user

Value

A list with the log marginal likelihood combined with the log prior (crit) and the posterior mode of the coefficients (coefs).

Examples

```
logistic.loglik(as.integer(rnorm(100) > 0), matrix(rnorm(100)), TRUE, list(oc = 1))
```

`logistic.loglik.ala` *Log likelihood function for logistic regression with an approximate Laplace approximations used This function is created as an example of how to create an estimator that is used to calculate the marginal likelihood of a model.*

Description

Log likelihood function for logistic regression with an approximate Laplace approximations used This function is created as an example of how to create an estimator that is used to calculate the marginal likelihood of a model.

Usage

```
logistic.loglik.ala(y, x, model, complex, params = list(r = exp(-0.5)))
```

Arguments

y	A vector containing the dependent variable
x	The matrix containing the precalculated features
model	The model to estimate as a logical vector
complex	A list of complexity measures for the features
params	A list of parameters for the log likelihood, supplied by the user

Value

A list with the log marginal likelihood combined with the log prior (crit) and the posterior mode of the coefficients (coefs).

Examples

```
logistic.loglik.ala(as.integer(rnorm(100) > 0), matrix(rnorm(100)), TRUE, list(oc = 1))
```

---

logistic.loglik.alpha *Log likelihood function for logistic regression for alpha calculation This function is just the bare likelihood function*

Description

Log likelihood function for logistic regression for alpha calculation This function is just the bare likelihood function

Usage

```
logistic.loglik.alpha(a, data, mu_func)
```

Arguments

a	A vector of the alphas to be used
data	The data to be used for calculation
mu_func	The function linking the mean to the covariates, as a string with the alphas as a[i].

Value

A numeric with the log likelihood.

log_prior *Log model prior function*

Description

Log model prior function

Usage

```
log_prior(params, complex)
```

Arguments

params	list of passed parameters of the likelihood in GMJMCMC
complex	list of complexity measures of the features included into the model

Value

A numeric with the log model prior.

Examples

```
log_prior(params = list(r=2), complex = list(oc = 2))
```

marginal.probs *Function for calculating marginal inclusion probabilities of features given a list of models*

Description

Function for calculating marginal inclusion probabilities of features given a list of models

Usage

```
marginal.probs(models)
```

Arguments

models	The list of models to use.

Value

A numeric vector of marginal model probabilities based on relative frequencies of model visits in MCMC.

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
marginal.probs(result$models[[1]])
```

`merge_results` *Merge a list of multiple results from many runs This function will weight the features based on the best marginal posterior in that population and merge the results together, simplifying by merging equivalent features (having high correlation).*

Description

Merge a list of multiple results from many runs This function will weight the features based on the best marginal posterior in that population and merge the results together, simplifying by merging equivalent features (having high correlation).

Usage

```
merge_results(
  results,
  populations = NULL,
  complex.measure = NULL,
  tol = NULL,
  data = NULL
)
```

Arguments

`results`	A list containing multiple results from GMJMCMC (Genetically Modified MJMCMC).
`populations`	Which populations should be merged from the results, can be "all", "last" (default) or "best".
`complex.measure`	The complex measure to use when finding the simplest equivalent feature, 1=total width, 2=operation count and 3=depth.
`tol`	The tolerance to use for the correlation when finding equivalent features, default is 0.0000001
`data`	Data to use when comparing features, default is NULL meaning that mock data will be generated, if data is supplied it should be of the same form as is required by gmjmcmc, i.e. with both x, y and an intercept.

Value

An object of class "gmjmcmc_merged" containing the following elements:

`features`	The features where equivalent features are represented in their simplest form.
`marg.probs`	Importance of features.
`counts`	Counts of how many versions that were present of each feature.
`results`	Results as they were passed to the function.
`pop.best`	The population in the results which contained the model with the highest log marginal posterior.
`thread.best`	The thread in the results which contained the model with the highest log marginal posterior.
`crit.best`	The highest log marginal posterior for any model in the results.
`reported`	The highest log marginal likelihood for the reported populations as defined in the populations argument.
`rep.pop`	The index of the population which contains reported.
`best.log.posteriors`	A matrix where the first column contains the population indices and the second column contains the model with the highest log marginal posterior within that population.
`rep.thread`	The index of the thread which contains reported.

Examples

```
result <- gmjmcmc.parallel(
 runs = 1,
 cores = 1,
 list(populations = "best", complex.measure = 2, tol = 0.0000001),
 matrix(rnorm(600), 100),
 P = 2,
 gaussian.loglik,
 loglik.alpha = gaussian.loglik.alpha,
 c("p0", "exp_dbl")
)

summary(result)

plot(result)

merge_results(result$results)
```

`mjmcmc` *Main algorithm for MJMCMC (Genetically Modified MJMCMC)*

Description

Main algorithm for MJMCMC (Genetically Modified MJMCMC)

Usage

```
mjmcmc(
  data,
  loglik.pi = gaussian.loglik,
  N = 100,
  probs = NULL,
  params = NULL,
  sub = FALSE,
  verbose = TRUE
)
```

Arguments

`data`	A matrix containing the data to use in the algorithm, first column should be the dependent variable, and the rest of the columns should be the independent variables.
`loglik.pi`	The (log) density to explore
`N`	The number of iterations to run for
`probs`	A list of the various probability vectors to use
`params`	A list of the various parameters for all the parts of the algorithm
`sub`	An indicator that if the likelihood is inexact and should be improved each model visit (EXPERIMENTAL!)
`verbose`	A logical denoting if messages should be printed

Value

A list containing the following elements:

`models`	All visited models.
`accept`	Average acceptance rate of the chain.
`lo.models`	All models visited during local optimization.
`best.crit`	The highest log marginal probability of the visited models.
`marg.probs`	Marginal probabilities of the features.
`model.probs`	Marginal probabilities of all of the visited models.
`model.probs.idx`	Indices of unique visited models.
`populations`	The covariates represented as a list of features.

Examples

```
result <- mjmcmc(matrix(rnorm(600), 100), gaussian.loglik)
summary(result)
plot(result)
```

`mjmcmc.parallel` *Run multiple mjmcmc runs in parallel, merging the results before returning.*

Description

Run multiple mjmcmc runs in parallel, merging the results before returning.

Usage

```
mjmcmc.parallel(runs = 2, cores = getOption("mc.cores", 2L), ...)
```

Arguments

`runs`	The number of runs to run
`cores`	The number of cores to run on
`...`	Further parameters passed to mjmcmc.

Value

Merged results from multiple mjmcmc runs

Examples

```
result <- mjmcmc.parallel(runs = 1, cores = 1, matrix(rnorm(600), 100), gaussian.loglik)
summary(result)
plot(result)
```

model.string *Function to generate a function string for a model consisting of features*

Description

Function to generate a function string for a model consisting of features

Usage

```
model.string(model, features, link = "I", round = 2)
```

Arguments

model	A logical vector indicating which features to include
features	The population of features
link	The link function to use, as a string
round	Rounding error for the features in the printed format

Value

A character representation of a model

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
summary(result)
plot(result)
model.string(c(TRUE, FALSE, TRUE, FALSE, TRUE), result$populations[[1]])
model.string(result$models[[1]][1][[1]]$model, result$populations[[1]])
```

ngelu *Negative GELU function*

Description

Negative GELU function

Usage

```
ngelu(x)
```

Arguments

x	The vector of values

Value

-x*pnorm(-x)

Examples

```
ngelu(2)
```

---

nhs *negative heavy side function*

Description

negative heavy side function

Usage

```
nhs(x)
```

Arguments

x	The vector of values

Value

as.integer(x<0)

Examples

```
nhs(2)
```

---

not *not x*

Description

not x

Usage

```
not(x)
```

Arguments

x	The vector of binary values

Value

1-x

Examples

```
not(TRUE)
```

`nrelu` *negative ReLu function*

Description

negative ReLu function

Usage

```
nrelu(x)
```

Arguments

`x`	The vector of values

Value

max(-x,0)

Examples

```
nrelu(2)
```

`p0` *p0 polynomial term*

Description

p0 polynomial term

Usage

```
p0(x)
```

Arguments

`x`	The vector of values

Value

log(abs(x) + .Machine$double.eps)

Examples

```
p0(2)
```

---

p05 *p05 polynomial term*

Description

p05 polynomial term

Usage

```
p05(x)
```

Arguments

x	The vector of values

Value

(abs(x)+.Machine$double.eps)^(0.5)

Examples

```
p05(2)
```

---

p0p0 *p0p0 polynomial term*

Description

p0p0 polynomial term

Usage

```
p0p0(x)
```

Arguments

x	The vector of values

Value

p0(x)*p0(x)

Examples

```
p0p0(2)
```

---

p0p05 *p0p05 polynomial term*

Description

p0p05 polynomial term

Usage

```
p0p05(x)
```

Arguments

x	The vector of values

Value

p0(x)*(abs(x)+.Machine$double.eps)^(0.5)

Examples

```
p0p05(2)
```

---

p0p1 *p0p1 polynomial term*

Description

p0p1 polynomial term

Usage

```
p0p1(x)
```

Arguments

x	The vector of values

Value

p0(x)*x

Examples

```
p0p1(2)
```

`p0p2` *p0p2 polynomial term*

Description

p0p2 polynomial term

Usage

```
p0p2(x)
```

Arguments

`x`	The vector of values

Value

p0(x)*x^(2)

Examples

```
p0p2(2)
```

`p0p3` *p0p3 polynomial term*

Description

p0p3 polynomial term

Usage

```
p0p3(x)
```

Arguments

`x`	The vector of values

Value

p0(x)*x^(3)

Examples

```
p0p3(2)
```

`p0pm05` *p0pm05 polynomial term*

Description

p0pm05 polynomial term

Usage

```
p0pm05(x)
```

Arguments

x	The vector of values

Value

p0(x)*sign(x)*(abs(x)+.Machine$double.eps)^(-0.5)

Examples

```
p0pm05(2)
```

`p0pm1` *p0pm1 polynomial terms*

Description

p0pm1 polynomial terms

Usage

```
p0pm1(x)
```

Arguments

x	The vector of values

Value

p0(x)*(x+.Machine$double.eps)^(-1)

Examples

```
p0pm1(2)
```

---

p0pm2 *p0pm2 polynomial term*

Description

p0pm2 polynomial term

Usage

```
p0pm2(x)
```

Arguments

x	The vector of values

Value

p0(x)*sign(x)*(abs(x)+.Machine$double.eps)^(-2)

Examples

```
p0pm2(2)
```

---

p2 *p2 polynomial term*

Description

p2 polynomial term

Usage

```
p2(x)
```

Arguments

x	The vector of values

Value

x^(2)

Examples

```
p2(2)
```

`p3` *p3 polynomial term*

Description

p3 polynomial term

Usage

```
p3(x)
```

Arguments

`x`	The vector of values

Value

x^(3)

Examples

```
p3(2)
```

`plot.gmjmcmc` *Function to plot the results, works both for results from gmjmcmc and merged results from merge.results*

Description

Function to plot the results, works both for results from gmjmcmc and merged results from merge.results

Usage

```
## S3 method for class 'gmjmcmc'
plot(x, count = "all", pop = "best", tol = 1e-07, data = NULL, ...)
```

Arguments

x	The results to use
count	The number of features to plot, defaults to all
pop	The population to plot, defaults to last
tol	The tolerance to use for the correlation when finding equivalent features, default is 0.0000001
data	Data to merge on, important if pre-filtering was used
...	Not used.

Value

No return value, just creates a plot

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
plot(result)
```

---

plot.gmjmcmc_merged *Plot a gmjmcmc_merged run*

Description

Plot a gmjmcmc_merged run

Usage

```
## S3 method for class 'gmjmcmc_merged'
plot(x, count = "all", pop = NULL, tol = 1e-07, data = NULL, ...)
```

Arguments

x	The results to use
count	The number of features to plot, defaults to all
pop	The population to plot, defaults to last
tol	The tolerance to use for the correlation when finding equivalent features, default is 0.0000001
data	Data to merge on, important if pre-filtering was used
...	Not used.

Value

No return value, just creates a plot

Examples

```
result <- gmjmcmc.parallel(
 runs = 1,
 cores = 1,
 list(populations = "best", complex.measure = 2, tol = 0.0000001),
 matrix(rnorm(600), 100),
 P = 2,
 gaussian.loglik,
 loglik.alpha = gaussian.loglik.alpha,
 c("p0", "exp_dbl")
)
plot(result)
```

`plot.mjmcmc` *Function to plot the results, works both for results from gmjmcmc and merged results from merge.results*

Description

Function to plot the results, works both for results from gmjmcmc and merged results from merge.results

Usage

```
## S3 method for class 'mjmcmc'
plot(x, count = "all", ...)
```

Arguments

`x`	The results to use
`count`	The number of features to plot, defaults to all
`...`	Not used.

Value

No return value, just creates a plot

Examples

```
result <- mjmcmc(matrix(rnorm(600), 100), gaussian.loglik)
plot(result)
```

plot.mjmcmc_parallel *Plot a mjmcmc_parallel run*

Description

Plot a mjmcmc_parallel run

Usage

```
## S3 method for class 'mjmcmc_parallel'
plot(x, count = "all", ...)
```

Arguments

x	The results to use
count	The number of features to plot, defaults to all
...	Not used.

Value

No return value, just creates a plot

Examples

```
result <- mjmcmc.parallel(runs = 1, cores = 1, matrix(rnorm(600), 100), gaussian.loglik)
plot(result)
```

pm05 *pm05 polynomial term*

Description

pm05 polynomial term

Usage

```
pm05(x)
```

Arguments

x	The vector of values

Value

(abs(x)+.Machine$double.eps)^(-0.5)

Examples

```
pm05(2)
```

`pm1` *pm1 polynomial term*

Description

pm1 polynomial term

Usage

```
pm1(x)
```

Arguments

x	The vector of values

Value

sign(x)*(abs(x)+.Machine$double.eps)^(-1)

Examples

```
pm1(2)
```

`pm2` *pm2 polynomial term*

Description

pm2 polynomial term

Usage

```
pm2(x)
```

Arguments

x	The vector of values

Value

sign(x)*(abs(x)+.Machine$double.eps)^(-2)

Examples

```
pm2(2)
```

`predict.bgnlm_model` *Predict responses from a BGNLM model*

Description

This function generates predictions from a fitted `bgnlm_model` object given a new dataset.

Usage

```
## S3 method for class 'bgnlm_model'
predict(
  object,
  x,
  link = function(x) {
     x
 },
  ...
)
```

Arguments

`object`	A fitted `bgnlm_model` object obtained from the BGNLM fitting procedure. It should contain the estimated coefficients in `model$coefs`.
`x`	A `data.frame` containing the new data for which predictions are to be made. The variables in `x` must match the features used in the model.
`link`	A link function to apply to the linear predictor. By default, it is the identity function `function(x){x}`, but it can be any function such as `plogis` for logistic regression models.
`...`	Additional arguments to pass to prediction function.

Value

A numeric vector of predicted values for the given data `x`. These predictions are calculated as $\hat{y} = \text{link}(X\beta)$, where X is the design matrix and β are the model coefficients.

Examples

```
## Not run:
# Example with simulated data
set.seed(123)
x_train <- data.frame(PlanetaryMassJpt = rnorm(100), RadiusJpt = rnorm(100))
model <- list(
  coefs = c(Intercept = -0.5, PlanetaryMassJpt = 0.2, RadiusJpt = -0.1),
```

```
  class = "bgnlm_model"
)
class(model) <- "bgnlm_model"

# New data for prediction
x_new <- data.frame(PlanetaryMassJpt = c(0.1, -0.3), RadiusJpt = c(0.2, -0.1))

# Predict using the identity link (default)
preds <- predict.bgnlm_model(model, x_new)

## End(Not run)
```

`predict.gmjmcmc` *Predict using a gmjmcmc result object.*

Description

Predict using a gmjmcmc result object.

Usage

```
## S3 method for class 'gmjmcmc'
predict(
  object,
  x,
  link = function(x) x,
  quantiles = c(0.025, 0.5, 0.975),
  pop = NULL,
  tol = 1e-07,
  ...
)
```

Arguments

`object`	The model to use.
`x`	The new data to use for the prediction, a matrix where each row is an observation.
`link`	The link function to use
`quantiles`	The quantiles to calculate credible intervals for the posterior modes (in model space).
`pop`	The population to plot, defaults to last
`tol`	The tolerance to use for the correlation when finding equivalent features, default is 0.0000001
`...`	Not used.

Value

A list containing aggregated predictions and per model predictions.

aggr	Aggregated predictions with mean and quantiles.
preds	A list of lists containing individual predictions per model per population in object.

Examples

```
result <- gmjmcmc(
 matrix(rnorm(600), 100),
 P = 2,
 gaussian.loglik,
 loglik.alpha = gaussian.loglik.alpha,
 c("p0", "exp_dbl")
)
preds <- predict(result, matrix(rnorm(600), 100))
```

---

predict.gmjmcmc_merged

Predict using a merged gmjmcmc result object.

Description

Predict using a merged gmjmcmc result object.

Usage

```
## S3 method for class 'gmjmcmc_merged'
predict(
  object,
  x,
  link = function(x) x,
  quantiles = c(0.025, 0.5, 0.975),
  pop = NULL,
  tol = 1e-07,
  ...
)
```

Arguments

object	The model to use.
x	The new data to use for the prediction, a matrix where each row is an observation.
link	The link function to use

quantiles	The quantiles to calculate credible intervals for the posterior modes (in model space).
pop	The population to plot, defaults to last
tol	The tolerance to use for the correlation when finding equivalent features, default is 0.0000001
...	Not used.

Value

A list containing aggregated predictions and per model predictions.

aggr	Aggregated predictions with mean and quantiles.
preds	A list of lists containing individual predictions per model per population in object.

Examples

```
result <- gmjmcmc.parallel(
 runs = 1,
 cores = 1,
 list(populations = "best", complex.measure = 2, tol = 0.0000001),
 matrix(rnorm(600), 100),
 P = 2,
 gaussian.loglik,
 loglik.alpha = gaussian.loglik.alpha,
 c("p0", "exp_dbl")
)
preds <- predict(result, matrix(rnorm(600), 100))
```

---

predict.gmjmcmc_parallel

Predict using a gmjmcmc result object from a parallel run.

Description

Predict using a gmjmcmc result object from a parallel run.

Usage

```
## S3 method for class 'gmjmcmc_parallel'
predict(object, x, link = function(x) x, quantiles = c(0.025, 0.5, 0.975), ...)
```

Arguments

object	The model to use.
x	The new data to use for the prediction, a matrix where each row is an observation.
link	The link function to use
quantiles	The quantiles to calculate credible intervals for the posterior modes (in model space).
...	Additional arguments to pass to merge_results.

Value

A list containing aggregated predictions and per model predictions.

aggr	Aggregated predictions with mean and quantiles.
preds	A list of lists containing individual predictions per model per population in object.

Examples

```
result <- gmjmcmc.parallel(
 runs = 1,
 cores = 1,
 list(populations = "best", complex.measure = 2, tol = 0.0000001),
 matrix(rnorm(600), 100),
 P = 2,
 gaussian.loglik,
 loglik.alpha = gaussian.loglik.alpha,
 c("p0", "exp_dbl")
)
preds <- predict(result$results, matrix(rnorm(600), 100))
```

---

predict.mjmcmc *Predict using a mjmcmc result object.*

Description

Predict using a mjmcmc result object.

Usage

```
## S3 method for class 'mjmcmc'
predict(object, x, link = function(x) x, quantiles = c(0.025, 0.5, 0.975), ...)
```

Arguments

object	The model to use.
x	The new data to use for the prediction, a matrix where each row is an observation.
link	The link function to use
quantiles	The quantiles to calculate credible intervals for the posterior modes (in model space).
...	Not used.

Value

A list containing aggregated predictions.

mean	Mean of aggregated predictions.
quantiles	Quantiles of aggregated predictions.

Examples

```
result <- mjmcmc(matrix(rnorm(600), 100), gaussian.loglik)
preds <- predict(result, matrix(rnorm(500), 100))
```

---

predict.mjmcmc_parallel

Predict using a mjmcmc result object from a parallel run.

Description

Predict using a mjmcmc result object from a parallel run.

Usage

```
## S3 method for class 'mjmcmc_parallel'
predict(object, x, link = function(x) x, quantiles = c(0.025, 0.5, 0.975), ...)
```

Arguments

object	The model to use.
x	The new data to use for the prediction, a matrix where each row is an observation.
link	The link function to use
quantiles	The quantiles to calculate credible intervals for the posterior modes (in model space).
...	Not used.

Value

A list containing aggregated predictions.

mean	Mean of aggregated predictions.
quantiles	Quantiles of aggregated predictions.

Examples

```
result <- mjmcmc.parallel(runs = 1, cores = 1, matrix(rnorm(600), 100), gaussian.loglik)
preds <- predict(result, matrix(rnorm(500), 100))
```

---

print.feature *Print method for "feature" class*

Description

Print method for "feature" class

Usage

```
## S3 method for class 'feature'
print(x, dataset = FALSE, alphas = FALSE, labels = FALSE, round = FALSE, ...)
```

Arguments

x	An object of class "feature"
dataset	Set the regular covariates as columns in a dataset
alphas	Print a "?" instead of actual alphas to prepare the output for alpha estimation
labels	Should the covariates be named, or just referred to as their place in the data.frame.
round	Should numbers be rounded when printing? Default is FALSE, otherwise it can be set to the number of decimal places.
...	Not used.

Value

String representation of a feature

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
print(result$populations[[1]][1])
```

`relu` *ReLu function*

Description

ReLu function

Usage

```
relu(x)
```

Arguments

`x`	The vector of values

Value

max(x,0)

Examples

```
relu(2)
```

`rmclapply` *rmclapply: Cross-platform mclapply/forking hack for Windows*

Description

This function applies a function in parallel to a list or vector (`X`) using multiple cores. On Linux/macOS, it uses `mclapply`, while on Windows it uses a hackish version of parallelism. The Windows version is based on `parLapply` to mimic forking following Nathan VanHoudnos.

Usage

```
rmclapply(runs, args, fun, mc.cores = NULL)
```

Arguments

`runs`	The runs to run
`args`	The arguments to pass to fun
`fun`	The function to run
`mc.cores`	Number of cores to use for parallel processing. Defaults to `detectCores()`.

Value

A list of results, with one element for each element of `X`.

SangerData2 *Gene expression data lymphoblastoid cell lines of all 210 unrelated HapMap individuals from four populations*

Description

A 210 times 3221 matrix with indviduals along the rows and expression data along the columns

Usage

```
data(SangerData2)
```

Format

A data frame with 210 rows and 3221 variables

Details

The first column corresponds to column number 24266 (with name GI_6005726-S) in the original data. Column names give the name of the variables, row names the "name" of the individuals. This is a subset of SangerData where the 3220 last rows are select among all original rows following the same pre-processing procedure as in (section 1.6.1). See also the file Read_sanger_data.R

Source

Dataset downloaded from https://ftp.sanger.ac.uk/pub/genevar/

References:

Stranger, BE et al (2007): Relative impact of nucleotide and copy number variation on gene expression phenotypes Science, 2007•science.org

Bogdan et al (2020): Handbook of Multiple Comparisons, https://arxiv.org/pdf/2011.12154

set.transforms *Set the transformations option for GMJMCMC (Genetically Modified MJMCMC), this is also done when running the algorithm, but this function allows for it to be done manually.*

Description

Set the transformations option for GMJMCMC (Genetically Modified MJMCMC), this is also done when running the algorithm, but this function allows for it to be done manually.

Usage

```
set.transforms(transforms)
```

Arguments

transforms	The vector of non-linear transformations

Value

No return value, just sets the gmjmcmc-transformations option

Examples

```
set.transforms(c("p0","p1"))
```

---

sigmoid *Sigmoid function*

Description

Sigmoid function

Usage

```
sigmoid(x)
```

Arguments

x	The vector of values

Value

The sigmoid of x

Examples

```
sigmoid(2)
```

sin_deg *Sine function for degrees*

Description

Sine function for degrees

Usage

```
sin_deg(x)
```

Arguments

x	The vector of values in degrees

Value

The sine of x

Examples

```
sin_deg(0)
```

sqroot *Square root function*

Description

Square root function

Usage

```
sqroot(x)
```

Arguments

x	The vector of values

Value

The square root of the absolute value of x

Examples

```
sqroot(4)
```

string.population *Function to get a character representation of a list of features*

Description

Function to get a character representation of a list of features

Usage

```
string.population(x, round = 2)
```

Arguments

x	A list of feature objects
round	Rounding precision for parameters of the features

Value

A matrix of character representations of the features of a model.

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
string.population(result$populations[[1]])
```

string.population.models *Function to get a character representation of a list of models*

Description

Function to get a character representation of a list of models

Usage

```
string.population.models(features, models, round = 2, link = "I")
```

Arguments

features	A list of feature objects on which the models are build
models	A list of model objects
round	Rounding precision for parameters of the features
link	The link function to use, as a string

Value

A matrix of character representations of a list of models.

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
string.population.models(result$populations[[2]], result$models[[2]])
```

---

summary.gmjmcmc *Function to print a quick summary of the results*

Description

Function to print a quick summary of the results

Usage

```
## S3 method for class 'gmjmcmc'
summary(
  object,
  pop = "best",
  tol = 1e-04,
  labels = FALSE,
  effects = NULL,
  data = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	The results to use
pop	The population to print for, defaults to last
tol	The tolerance to use as a threshold when reporting the results.
labels	Should the covariates be named, or just referred to as their place in the data.frame.
effects	Quantiles for posterior modes of the effects across models to be reported, if either effects are NULL or if labels are NULL, no effects are reported.
data	Data to merge on, important if pre-filtering was used
verbose	If the summary should be printed to the console or just returned, defaults to TRUE
...	Not used.

Value

A data frame containing the following columns:

`feats.strings`	Character representation of the features ordered by marginal probabilities.
`marg.probs`	Marginal probabilities corresponding to the ordered feature strings.

Examples

```
result <- gmjmcmc(matrix(rnorm(600), 100), P = 2, gaussian.loglik, NULL, c("p0", "exp_dbl"))
summary(result, pop = "best")
```

---

summary.gmjmcmc_merged

Function to print a quick summary of the results

Description

Function to print a quick summary of the results

Usage

```
## S3 method for class 'gmjmcmc_merged'
summary(
  object,
  tol = 1e-04,
  labels = FALSE,
  effects = NULL,
  pop = NULL,
  data = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	The results to use
`tol`	The tolerance to use as a threshold when reporting the results.
`labels`	Should the covariates be named, or just referred to as their place in the data.frame.
`effects`	Quantiles for posterior modes of the effects across models to be reported, if either effects are NULL or if labels are NULL, no effects are reported.
`pop`	If null same as in merge.options for running parallel gmjmcmc otherwise results will be re-merged according to pop that can be "all", "last", "best"
`data`	Data to merge on, important if pre-filtering was used
`verbose`	If the summary should be printed to the console or just returned, defaults to TRUE
`...`	Not used.

Value

A data frame containing the following columns:

feats.strings	Character representation of the features ordered by marginal probabilities.
marg.probs	Marginal probabilities corresponding to the ordered feature strings.

Examples

```
result <- gmjmcmc.parallel(
 runs = 1,
 cores = 1,
 list(populations = "best", complex.measure = 2, tol = 0.0000001),
 matrix(rnorm(600), 100),
 P = 2,
 gaussian.loglik,
 loglik.alpha = gaussian.loglik.alpha,
 c("p0", "exp_dbl")
)
summary(result)
```

---

summary.mjmcmc *Function to print a quick summary of the results*

Description

Function to print a quick summary of the results

Usage

```
## S3 method for class 'mjmcmc'
summary(
  object,
  tol = 1e-04,
  labels = FALSE,
  effects = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	The results to use
tol	The tolerance to use as a threshold when reporting the results.
labels	Should the covariates be named, or just referred to as their place in the data.frame.
effects	Quantiles for posterior modes of the effects across models to be reported, if either effects are NULL or if labels are NULL, no effects are reported.

verbose	If the summary should be printed to the console or just returned, defaults to TRUE
...	Not used.

Value

A data frame containing the following columns:

feats.strings	Character representation of the covariates ordered by marginal probabilities.
marg.probs	Marginal probabilities corresponding to the ordered feature strings.

Examples

```
result <- mjmcmc(matrix(rnorm(600), 100), gaussian.loglik)
summary(result)
```

---

summary.mjmcmc_parallel

Function to print a quick summary of the results

Description

Function to print a quick summary of the results

Usage

```
## S3 method for class 'mjmcmc_parallel'
summary(
  object,
  tol = 1e-04,
  labels = FALSE,
  effects = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	The results to use
tol	The tolerance to use as a threshold when reporting the results.
labels	Should the covariates be named, or just referred to as their place in the data.frame.
effects	Quantiles for posterior modes of the effects across models to be reported, if either effects are NULL or if labels are NULL, no effects are reported.
verbose	If the summary should be printed to the console or just returned, defaults to TRUE
...	Not used.

Value

A data frame containing the following columns:

`feats.strings`	Character representation of the covariates ordered by marginal probabilities.
`marg.probs`	Marginal probabilities corresponding to the ordered feature strings.

Examples

```
result <- mjmcmc.parallel(runs = 1, cores = 1, matrix(rnorm(600), 100), gaussian.loglik)
summary(result)
```

---

to23 *To the 2.3 power function*

Description

To the 2.3 power function

Usage

```
to23(x)
```

Arguments

`x`	The vector of values

Value

x^2.3

Examples

```
to23(2)
```

to25 *To 2.5 power*

Description

To 2.5 power

Usage

```
to25(x)
```

Arguments

x	The vector of values

Value

x^(2.5)

Examples

```
to25(2)
```

to35 *To 3.5 power*

Description

To 3.5 power

Usage

```
to35(x)
```

Arguments

x	The vector of values

Value

x^(3.5)

Examples

```
to35(2)
```

to72 *To the 7/2 power function*

Description

To the 7/2 power function

Usage

```
to72(x)
```

Arguments

x	The vector of values

Value

x^(7/2)

Examples

```
to72(2)
```

troot *Cube root function*

Description

Cube root function

Usage

```
troot(x)
```

Arguments

x	The vector of values

Value

The cube root of x

Examples

```
troot(27)
```

Index