Package: ggmix (via r-universe)

September 6, 2024

Type Package

Title Variable Selection in Linear Mixed Models for SNP Data

Version 0.0.2

Description Fit penalized multivariable linear mixed models with a single random effect to control for population structure in genetic association studies. The goal is to simultaneously fit many genetic variants at the same time, in order to select markers that are independently associated with the response. Can also handle prior annotation information, for example, rare variants, in the form of variable weights. For more information, see the website below and the accompanying paper: Bhatnagar et al., ``Simultaneous SNP selection and adjustment for population structure in high dimensional prediction models'', 2020, <DOI:10.1371/journal.pgen.1008766>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.4.0)

Imports glmnet, methods, stats, MASS, Matrix

Suggests RSpectra, popkin, bnpsd, testthat, covr, knitr, rmarkdown

BugReports `https://github.com/sahirbhatnagar/ggmix/issues`

URL `https://github.com/sahirbhatnagar/ggmix`

RoxygenNote 7.1.1

VignetteBuilder knitr

Repository https://sahirbhatnagar.r-universe.dev

RemoteUrl https://github.com/sahirbhatnagar/ggmix

RemoteRef HEAD

RemoteSha eb1c8a71ddc8f7e450dfbab68e7c562eac0ed487

Contents

`admixed` *Simulated Dataset with 1D Geography*

Description

A simulated dataset to show the utility of this package

Usage

```
admixed
```

Format

An object of class `list` of length 21.

Details

The code used to simulate the data is available at `https://github.com/sahirbhatnagar/ggmix/blob/master/data-raw/bnpsd-data.R`. See `gen_structured_model` for more details on the output and how the function used to simulate the data.

Value

A list with the following elements

ytrain simulated response vector for training set

ytune simulated response vector for tuning parameter selection set

ytest simulated response vector for test set

xtrain simulated design matrix for training set

xtune simulated design matrix for tuning parameter selection set

xtest simulated design matrix for testing set

xtrain_lasso simulated design matrix for training set for lasso model. This is the same as xtrain, but also includes the nPC principal components

xtune_lasso simulated design matrix for tuning parameter selection set for lasso model. This is the same as xtune, but also includes the nPC principal components

xtest simulated design matrix for testing set for lasso model. This is the same as xtest, but also includes the nPC principal components

causal character vector of the names of the causal SNPs

beta the vector of true regression coefficients

kin_train 2 times the estimated kinship for the training set individuals

kin_tune_train The covariance matrix between the tuning set and the training set individuals

kin_test_train The covariance matrix between the test set and training set individuals

Xkinship the matrix of SNPs used to estimate the kinship matrix

not_causal character vector of the non-causal SNPs

PC the principal components for population structure adjustment

References

Ochoa, Alejandro, and John D. Storey. 2016a. "FST And Kinship for Arbitrary Population Structures I: Generalized Definitions." bioRxiv doi:10.1101/083915.

Ochoa, Alejandro, and John D. Storey. 2016b. "FST And Kinship for Arbitrary Population Structures II: Method of Moments Estimators." bioRxiv doi:10.1101/083923.

Examples

```
data(admixed)
str(admixed)
```

`gen_structured_model` *Simulation Scenario from Bhatnagar et al. (2018+) ggmix paper*

Description

Function that generates data of the different simulation studies presented in the accompanying paper. This function requires the `popkin` and `bnpsd` package to be installed.

Usage

```
gen_structured_model(
  n,
  p_design,
  p_kinship,
  k,
  s,
  Fst,
  b0,
  nPC = 10,
  eta,
  sigma2,
  geography = c("ind", "1d", "circ"),
  percent_causal,
  percent_overlap,
  train_tune_test = c(0.6, 0.2, 0.2)
)
```

Arguments

`n` number of observations to simulate

`p_design` number of variables in X_test, i.e., the design matrix

`p_kinship` number of variable in X_kinship, i.e., matrix used to calculate kinship

`k` number of intermediate subpopulations.

`s` the desired bias coefficient, which specifies sigma indirectly. Required if sigma is missing

`Fst` The desired final FST of the admixed individuals. Required if sigma is missing

`b0` the true intercept parameter

`nPC` number of principal components to include in the design matrix used for regression adjustment for population structure via principal components. This matrix is used as the input in a standard lasso regression routine, where there are no random effects.

`eta` the true eta parameter, which has to be `0 < eta < 1`

`sigma2` the true sigma2 parameter

`geography` the type of geography for simulation the kinship matrix. "ind" is independent populations where every individuals is actually unadmixed, "1d" is a 1D geography and "circ" is circular geography. Default: "ind". See the functions in the `bnpsd` for details on how this data is actually generated.

`percent_causal` percentage of `p_design` that is causal. must be $0 \leq percent_causal \leq 1$. The true regression coefficients are generated from a standard normal distribution.

`percent_overlap`
this represents the percentage of causal SNPs that will also be included in the calculation of the kinship matrix

`train_tune_test`
the proportion of sample size used for training tuning parameter selection and testing. default is 60/20/20 split

Details

The kinship is estimated using the `popkin` function from the `popkin` package. This function will multiple that kinship matrix by 2 to give the expected covariance matrix which is subsequently used in the linear mixed models

Value

A list with the following elements

ytrain simulated response vector for training set

ytune simulated response vector for tuning parameter selection set

ytest simulated response vector for test set

xtrain simulated design matrix for training set

xtune simulated design matrix for tuning parameter selection set

xtest simulated design matrix for testing set

xtrain_lasso simulated design matrix for training set for lasso model. This is the same as xtrain, but also includes the nPC principal components

xtune_lasso simulated design matrix for tuning parameter selection set for lasso model. This is the same as xtune, but also includes the nPC principal components

xtest simulated design matrix for testing set for lasso model. This is the same as xtest, but also includes the nPC principal components

causal character vector of the names of the causal SNPs

beta the vector of true regression coefficients

kin_train 2 times the estimated kinship for the training set individuals

kin_tune_train The covariance matrix between the tuning set and the training set individuals

kin_test_train The covariance matrix between the test set and training set individuals

Xkinship the matrix of SNPs used to estimate the kinship matrix

not_causal character vector of the non-causal SNPs

PC the principal components for population structure adjustment

See Also

admix_prop_1d_linear

Examples

```
admixed <- gen_structured_model(n = 100,
                                p_design = 50,
                                p_kinship = 5e2,
                                geography = "1d",
                                percent_causal = 0.10,
                                percent_overlap = "100",
                                k = 5, s = 0.5, Fst = 0.1,
                                b0 = 0, nPC = 10,
                                eta = 0.1, sigma2 = 1,
                                train_tune_test = c(0.8, 0.1, 0.1))
names(admixed)
```

---

ggmix *Fit Linear Mixed Model with Lasso or Group Lasso Regularization*

Description

Main function to fit the linear mixed model with lasso or group lasso penalty for a sequence of tuning parameters. This is a penalized regression method that accounts for population structure using either the kinship matrix or the factored realized relationship matrix

Usage

```
ggmix(
  x,
  y,
  U,
  D,
  kinship,
  K,
  n_nonzero_eigenvalues,
  n_zero_eigenvalues,
  estimation = c("full"),
  penalty = c("lasso"),
  group,
  penalty.factor = rep(1, p_design),
  lambda = NULL,
  lambda_min_ratio = ifelse(n_design < p_design, 0.01, 1e-04),
  nlambda = 100,
  eta_init = 0.5,
  maxit = 100,
  fdev = 1e-20,
  standardize = FALSE,
  alpha = 1,
  thresh_glmnet = 1e-08,
  epsilon = 1e-04,
  dfmax = p_design + 2,
  verbose = 0
)
```

Arguments

`x` input matrix, of dimension n x p; where n is the number of observations and p are the number of predictors.

`y` response variable. must be a quantitative variable

`U` left singular vectors corresponding to the non-zero eigenvalues provided in the `D` argument.

`D` non-zero eigenvalues. This option is provided to the user should they decide or need to calculate the eigen decomposition of the kinship matrix or the singular value decomposition of the matrix of SNPs used to calculate the kinship outside of this function. This may occur, if for example, it is easier (e.g. because of memory issues, it's easier to calculate in plink). This should correspond to the non-zero eigenvalues only. Note that if you are doing an `svd` on the matrix of SNPs used to calculate the kinship matrix, then you must provide the square of the singular values so that they correspond to the eigenvalues of the kinship matrix. If you want to use the low rank estimation algorithm, you must provide the truncated eigenvalues and eigenvectors to the `D` and `U` arguments, respectively. If you want `ggmix` to truncate the eigenvectors and eigenvalues for low rank estimation, then provide either `K` or `kinship` instead and specify `n_nonzero_eigenvalues`.

`kinship` positive definite kinship matrix

`K` the matrix of SNPs used to determine the kinship matrix

`n_nonzero_eigenvalues` the number of nonzero eigenvalues. This argument is only used when `estimation="low"` and either `kinship` or `K` is provided. This argument will limit the function to finding the `n_nonzero_eigenvalues` largest eigenvalues. If `U` and `D` have been provided, then `n_nonzero_eigenvalues` defaults to the length of `D`.

`n_zero_eigenvalues` Currently not being used. Represents the number of zero eigenvalues. This argument must be specified when `U` and `D` are specified and `estimation="low"`. This is required for low rank estimation because the number of zero eigenvalues and their corresponding eigenvalues appears in the likelihood. In general this would be the rank of the matrix used to calculate the eigen or singular value decomposition. When `kinship` is provided and `estimation="low"` the default value will be `nrow(kinship) - n_nonzero_eigenvalues`. When `K` is provided and `estimation="low"`, the default value is `rank(K) - n_nonzero_eigenvalues`

`estimation` type of estimation. Currently only `type="full"` has been implemented.

`penalty` type of regularization penalty. Currently, only penalty="lasso" has been implemented.

`group` a vector of consecutive integers describing the grouping of the coefficients. Currently not implemented, but will be used when penalty="gglasso" is implemented.

`penalty.factor` Separate penalty factors can be applied to each coefficient. This is a number that multiplies lambda to allow differential shrinkage. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is 1 for all variables

`lambda` A user supplied lambda sequence (this is the tuning parameter). Typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Supplying a value of lambda overrides this. WARNING: use with care. Do not supply a single value for lambda (for predictions after CV use predict() instead). Supply instead a decreasing sequence of lambda values. glmnet relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.

`lambda_min_ratio`

Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default depends on the sample size nobs relative to the number of variables nvars. If nobs > nvars, the default is 0.0001, close to zero. If nobs < nvars, the default is 0.01. A very small value of lambda.min.ratio will lead to a saturated fit in the nobs < nvars case.

`nlambda` the number of lambda values - default is 100.

`eta_init` initial value for the eta parameter, with $0 < \eta < 1$ used in determining lambda.max and starting value for fitting algorithm.

`maxit` Maximum number of passes over the data for all lambda values; default is 10^2.

`fdev` Fractional deviance change threshold. If change in deviance between adjacent lambdas is less than fdev, the solution path stops early. factory default = 1.0e-5

`standardize` Logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is standardize=FALSE. If variables are in the same units already, you might not wish to standardize.

`alpha` The elasticnet mixing parameter, with $0 \leq \alpha \leq 1$. alpha=1 is the lasso penalty, and alpha=0 the ridge penalty.

`thresh_glmnet` Convergence threshold for coordinate descent for updating beta parameters. Each inner coordinate-descent loop continues until the maximum change in the objective after any coefficient update is less than thresh times the null deviance. Defaults value is 1E-7

`epsilon` Convergence threshold for block relaxation of the entire parameter vector $\Theta = (\beta, \eta, \sigma^2)$. The algorithm converges when

$$crossprod(\Theta_{j+1} - \Theta_j) < \epsilon$$

. Defaults value is 1E-4

`dfmax` limit the maximum number of variables in the model. Useful for very large `p` (the total number of predictors in the design matrix), if a partial path is desired. Default is the number of columns in the design matrix + 2 (for the variance components) `verbose` display progress. Can be either 0,1 or 2. 0 will not display any progress, 2 will display very detailed progress and 1 is somewhere in between. Default: 0.

Value

list which includes the fitted object, lambda sequence, solution path, variance-covariance parameters, degrees of freedom, and singular values, vectors of kinship matrix

References

Bhatnagar, Sahir R, Yang, Yi, Lu, Tianyuan, Schurr, Erwin, Loredo-Osti, JC, Forest, Marie, Oualkacha, Karim, and Greenwood, Celia MT. (2020) *Simultaneous SNP selection and adjustment for population structure in high dimensional prediction models* doi:10.1101/408484

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*, `https://web.stanford.edu/~hastie/Papers/glmnet.pdf` *Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010* `https://www.jstatsoft.org/v33/i01/`

Yang, Y., & Zou, H. (2015). A fast unified algorithm for solving group-lasso penalize learning problems. *Statistics and Computing*, 25(6), 1129-1141. `https://www.math.mcgill.ca/yyang/resources/papers/STCO_gglasso.pdf`

Examples

```
data(admixed)
fitlmm <- ggmix(x = admixed$xtrain, y = admixed$ytrain,
                kinship = admixed$kin_train,
                estimation = "full")
gicfit <- gic(fitlmm)
coef(gicfit, type = "nonzero")
predict(gicfit, newx = admixed$xtest)[1:5,,drop=FALSE]
plot(gicfit)
plot(fitlmm)
```

`ggmix_data_object` *Constructor functions for the different ggmix objects*

Description

`new_fullrank_kinship`, `new_fullrank_K`, `new_fullrank_UD`, `new_lowrank_kinship`, `new_lowrank_K` and `new_lowrank_UD` create the ggmix objects from the provided data that are necessary to fit the penalized linear mixed model according to the user's parameters.

Usage

```
new_fullrank_kinship(x, y, kinship)

new_fullrank_K(x, y, K)
```

```
new_fullrank_UD(x, y, U, D)

new_lowrank_kinship(x, y, kinship, n_nonzero_eigenvalues, n_zero_eigenvalues)

new_lowrank_K(x, y, K, n_nonzero_eigenvalues, n_zero_eigenvalues)

new_lowrank_UD(x, y, U, D, n_nonzero_eigenvalues, n_zero_eigenvalues)
```

Arguments

`x` input matrix, of dimension n x p; where n is the number of observations and p are the number of predictors.

`y` response variable. must be a quantitative variable

`kinship` positive definite kinship matrix

`K` the matrix of SNPs used to determine the kinship matrix

`U` left singular vectors corresponding to the non-zero eigenvalues provided in the `D` argument.

`D` non-zero eigenvalues. This option is provided to the user should they decide or need to calculate the eigen decomposition of the kinship matrix or the singular value decomposition of the matrix of SNPs used to calculate the kinship outside of this function. This may occur, if for example, it is easier (e.g. because of memory issues, it's easier to calculate in plink). This should correspond to the non-zero eigenvalues only. Note that if you are doing an `svd` on the matrix of SNPs used to calculate the kinship matrix, then you must provide the square of the singular values so that they correspond to the eigenvalues of the kinship matrix. If you want to use the low rank estimation algorithm, you must provide the truncated eigenvalues and eigenvectors to the `D` and `U` arguments, respectively. If you want `ggmix` to truncate the eigenvectors and eigenvalues for low rank estimation, then provide either `K` or `kinship` instead and specify `n_nonzero_eigenvalues`.

`n_nonzero_eigenvalues`
the number of nonzero eigenvalues. This argument is only used when `estimation="low"` and either `kinship` or `K` is provided. This argument will limit the function to finding the `n_nonzero_eigenvalues` largest eigenvalues. If `U` and `D` have been provided, then `n_nonzero_eigenvalues` defaults to the length of `D`.

`n_zero_eigenvalues`
the number of desired or specified zero eigenvalues. This is only needed when `estimation="lowrank"`, and is calculated internally by the `ggmix` function. It is equal to the number of observations minus `n_nonzero_eigenvalues`

Value

A ggmix object, of the class that corresponds to the estimation method. These objects are lists that contain the data necessary for computation. These functions are not meant to be called directly by the user

See Also

ggmix

`gic` *Generalised Information Criterion*

Description

Calculates the generalised information criterion for each value of the tuning parameter lambda

Usage

```
gic(ggmix_fit, ...)

## Default S3 method:
gic(ggmix_fit, ...)

## S3 method for class 'ggmix_fit'
gic(ggmix_fit, ..., an = log(log(n)) * log(p))
```

Arguments

`ggmix_fit`	An object of class `ggmix_fit` which is outputted by the `ggmix` function
`...`	other parameters. currently ignored.
`an`	numeric, the penalty per parameter to be used; the default is an = log(log(n))*log(p) where n is the number of subjects and p is the number of parameters

Details

the generalised information criterion used for gaussian response is given by

$$-2 * loglikelihood(\hat{\Theta}) + an * df$$

where df is the number of non-zero estimated parameters, including variance components

Value

an object with S3 class `"ggmix_gic"`, `"ggmix_fit"`, `"*"` and `"**"` where `"*"` is "lasso" or "gglasso" and `"**"` is fullrank or lowrank. Results are provided for converged values of lambda only.

ggmix_fit the ggmix_fit object

lambda the sequence of converged tuning parameters

nzero the number of non-zero estimated coefficients including the 2 variance parameters which are not penalized and therefore always included

gic gic value. a numeric vector with length equal to `length(lambda)`

lambda.min.name a character corresponding to the name of the tuning parameter lambda which minimizes the gic

lambda.min the value of lambda which minimizes the gic

References

Fan Y, Tang CY. Tuning parameter selection in high dimensional penalized likelihood. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2013 Jun 1;75(3):531-52.

Nishii R. Asymptotic properties of criteria for selection of variables in multiple regression. The Annals of Statistics. 1984;12(2):758-65.

See Also

`ggmix`

`gr_eta_lasso_fullrank` *Functions related to eta parameter used in optim and kkt checks*

Description

Used for gradient of eta. Currently being passed to optim in `lmmlasso` and used in `kkt_check`

Usage

```
gr_eta_lasso_fullrank(eta, sigma2, beta, eigenvalues, x, y, nt)

fn_eta_lasso_fullrank(eta, sigma2, beta, eigenvalues, x, y, nt)
```

Arguments

`eta`	current estimate of the eta parameter
`sigma2`	current estimate of the sigma2 parameter
`beta`	current estimate of the beta parameter including the intercept. this should be of length p+1, where p is the number of variables.
`eigenvalues`	non-zero eigenvalues of the kinship matrix, or the square of the singular values of the matrix used to construct the kinship matrix
`x`	input matrix, of dimension n x p; where n is the number of observations and p are the number of predictors.
`y`	response variable. must be a quantitative variable
`nt`	total number of observations

Value

returns the value of the function and gradient for the eta variance parameter

See Also

`logliklasso`, `kkt_check`, `lmmlasso`

karim *Karim's Simulated Data*

Description

A simulated dataset with a kinship matrix

Usage

```
karim
```

Format

A list with 6 elements:

- **b** vector of length 1000 representing the true regression coefficients. 10 non-zero coefficients, the rest are 0.
- **kin1** the true kinship matrix
- **s.g** polygenic variance, set to be 1.26
- **s.e** error variance, set to be 1
- **h.tot** the total trait heritability. Set to be 60 of genotypes of dimension 600 x 1000 SNPs, with approximately 800 common and 200 rare SNPs

Details

If you simulate data using the scenario provided in the example, then the QTL heritability of y will be 8 of the trait's total heritability), and the trait total heritability is set to be 60

Examples

```
data(karim)
# Simulate a response using the genotype matrix and the kinship matrix
Phi <- 2 * karim$kin1
intercept <- 1
P <- MASS::mvrnorm(1, rep(0,600), karim$s.g * Phi)
y <- intercept + karim$G %*% karim$b + P + rnorm(600,0,karim$s.e)
```

kkt_check *Check of KKT Conditions for Linear Mixed Model*

Description

This function checks the KKT conditions

Usage

```
kkt_check(eta, sigma2, beta, eigenvalues, x, y, nt, lambda, tol.kkt = 0.001)

grr_sigma2(eta, sigma2, beta, eigenvalues, x, y, nt)

grr_beta0(eta, sigma2, beta, eigenvalues, x, y, nt)
```

Arguments

`eta`	current estimate of the eta parameter
`sigma2`	current estimate of the sigma2 parameter
`beta`	current estimate of the beta parameter including the intercept. this should be of length p+1, where p is the number of variables.
`eigenvalues`	non-zero eigenvalues of the kinship matrix, or the square of the singular values of the matrix used to construct the kinship matrix
`x`	rotated x. Should be U^T X, where U is the matrix of eigenvectors and X contains the first column of ones for the intercept. x should be a mtrix of dimension n x (p+1). These are outputted by the constructor functions. See `ggmix_data_object` for details
`y`	rotated y. Should be U^T Y, where U is the matrix of eigenvectors and Y is the response.
`nt`	total number of observations
`lambda`	A user supplied lambda sequence (this is the tuning parameter). Typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Supplying a value of lambda overrides this. WARNING: use with care. Do not supply a single value for lambda (for predictions after CV use predict() instead). Supply instead a decreasing sequence of lambda values. glmnet relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.
`tol.kkt`	Tolerance for determining if an entry of the subgradient is zero

Value

returns the values of the gradient for each of the parameters

Note

`grr_sigma2` and `grr_beta0` are functions for the gradient of sigma2 and beta0, respectively

lambdalasso — *Estimation of Lambda Sequence for Linear Mixed Model with Lasso Penalty*

Description

`lambdalasso` estimates a decreasing sequence of tuning parameters

Usage

```
lambdalasso(ggmix_object, ...)

## Default S3 method:
lambdalasso(ggmix_object, ...)

## S3 method for class 'fullrank'
lambdalasso(
  ggmix_object,
  ...,
  penalty.factor,
  lambda_min_ratio,
  epsilon = 1e-14,
  tol.kkt = 1e-09,
  eta_init = 0.5,
  nlambda = 100,
  scale_x = FALSE,
  center_y = FALSE
)
```

Arguments

`ggmix_object` A ggmix_object object of class `lowrank` or `fullrank`

`...` Extra parameters. Currently ignored.

`penalty.factor` Separate penalty factors can be applied to each coefficient. This is a number that multiplies lambda to allow differential shrinkage. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is 1 for all variables

`lambda_min_ratio` Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default depends on the sample size nobs relative to the number of variables nvars. If nobs > nvars, the default is 0.0001, close to zero. If nobs < nvars, the default is 0.01. A very small value of lambda.min.ratio will lead to a saturated fit in the nobs < nvars case.

`epsilon` Convergence threshold for block relaxation of the entire parameter vector $\Theta = (\beta, \eta, \sigma^2)$. The algorithm converges when

$$crossprod(\Theta_{j+1} - \Theta_j) < \epsilon$$

. Defaults value is 1E-4

`tol.kkt` KKT tolerance. Currently ignored

`eta_init` initial value for the eta parameter, with $0 < \eta < 1$ used in determining lambda.max and starting value for fitting algorithm.

`nlambda` the number of lambda values - default is 100.

`scale_x` should the columns of x be scaled - default is FALSE

`center_y` should y be mean centered - default is FALSE.

Value

A decreasing sequence of tuning parameters

Note

This function isn't meant to be called directly by the user.

See Also

ggmix

---

lmmlasso *Estimation of Linear Mixed Model with Lasso Penalty*

Description

`lmmlasso` estimates the linear mixed model with lasso penalty

Usage

```
lmmlasso(ggmix_object, ...)

## Default S3 method:
lmmlasso(ggmix_object, ...)

## S3 method for class 'fullrank'
lmmlasso(
  ggmix_object,
  ...,
  penalty.factor,
  lambda,
  lambda_min_ratio,
  nlambda,
  n_design,
  p_design,
  eta_init,
  maxit,
  fdev,
  standardize,
  alpha,
  thresh_glmnet,
  epsilon,
  dfmax,
  verbose
)
```

Arguments

`ggmix_object` A ggmix_object object of class `lowrank` or `fullrank`

`...` Extra parameters. Currently ignored.

`penalty.factor` Separate penalty factors can be applied to each coefficient. This is a number that multiplies lambda to allow differential shrinkage. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is 1 for all variables

`lambda` A user supplied lambda sequence (this is the tuning parameter). Typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Supplying a value of lambda overrides this. WARNING: use with care. Do not supply a single value for lambda (for predictions after CV use predict() instead). Supply instead a decreasing sequence of lambda values. glmnet relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.

`lambda_min_ratio` Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default depends on the sample size nobs relative to the number of variables nvars. If nobs > nvars, the default is 0.0001, close to zero. If nobs < nvars, the default is 0.01. A very small value of lambda.min.ratio will lead to a saturated fit in the nobs < nvars case.

`nlambda` the number of lambda values - default is 100.

`n_design` total number of observations

`p_design` number of variables in the design matrix, excluding the intercept column

`eta_init` initial value for the eta parameter, with $0 < \eta < 1$ used in determining lambda.max and starting value for fitting algorithm.

`maxit` Maximum number of passes over the data for all lambda values; default is 10^2.

`fdev` Fractional deviance change threshold. If change in deviance between adjacent lambdas is less than fdev, the solution path stops early. factory default = 1.0e-5 `standardize` Logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is standardize=FALSE. If variables are in the same units already, you might not wish to standardize.

`alpha` The elasticnet mixing parameter, with $0 \leq \alpha \leq 1$. alpha=1 is the lasso penalty, and alpha=0 the ridge penalty.

`thresh_glmnet` Convergence threshold for coordinate descent for updating beta parameters. Each inner coordinate-descent loop continues until the maximum change in the objective after any coefficient update is less than thresh times the null deviance. Defaults value is 1E-7

`epsilon` Convergence threshold for block relaxation of the entire parameter vector $\Theta = (\beta, \eta, \sigma^2)$. The algorithm converges when

$$crossprod(\Theta_{j+1} - \Theta_j) < \epsilon$$

. Defaults value is 1E-4

`dfmax` limit the maximum number of variables in the model. Useful for very large `p` (the total number of predictors in the design matrix), if a partial path is desired. Default is the number of columns in the design matrix + 2 (for the variance components)

`verbose` display progress. Can be either 0,1 or 2. 0 will not display any progress, 2 will display very detailed progress and 1 is somewhere in between. Default: 0.

Value

A object of class `ggmix`

See Also

`ggmix`

`logliklasso` *Estimation of Log-likelihood for Linear Mixed Model with Lasso Penalty*

Description

`sigma2lasso` estimates the value of the sigma2 for the linear mixed model with lasso penalty

Usage

```
logliklasso(ggmix_object, ...)

## Default S3 method:
logliklasso(ggmix_object, ...)

## S3 method for class 'fullrank'
logliklasso(ggmix_object, ..., eta, sigma2, beta, nt, x, y)
```

Arguments

`ggmix_object` A ggmix_object object of class `lowrank` or `fullrank`

`...` Extra parameters. Currently ignored.

`eta` current estimate of the eta parameter

`sigma2` current estimate of the sigma2 parameter

`beta` current estimate of the beta parameter including the intercept. this should be of length p+1, where p is the number of variables.

`nt` total number of observations

`x` input matrix, of dimension n x p; where n is the number of observations and p are the number of predictors.

`y` response variable. must be a quantitative variable

Value

A decreasing sequence of tuning parameters

Note

This function isn't meant to be called directly by the user.

See Also

`ggmix`

`ggmix_data_object`

`plot.ggmix_fit` *Plot Method for* `ggmix_fit` *object*

Description

Produces a coefficient profile plot of the coefficient paths for a fitted `ggmix_fit` object.

Usage

```
## S3 method for class 'ggmix_fit'
plot(x, ..., xvar = c("norm", "lambda", "dev"), label = FALSE, sign.lambda = 1)

plotCoef(
  beta,
  norm,
  lambda,
  df,
  dev,
  label = FALSE,
  xvar = c("norm", "lambda", "dev"),
  xlab = iname,
  ylab = "Coefficients",
  ...
)
```

Arguments

`x` a `ggmix_fit` object

`...` other graphical parameters passed to `plot`

`xvar` What is on the X-axis. "norm" plots against the L1-norm of the coefficients, "lambda" against the log-lambda sequence, and "dev" against the percent deviance explained.

`label` If TRUE, label the curves with variable sequence numbers.

`sign.lambda` Either plot against log(lambda) (default) or its negative if sign.lambda=-1

`beta` fixed effects estimates

`norm` l1 norm of fixed effect estimates. if missing, (default) this function will calculate it

`lambda` sequence of tuning parameters

`df` number of non-zero fixed + random effects

`dev` percent deviance

`xlab` x-axis label

`ylab` y-axis label

Details

A coefficient profile plot is produced

Value

A plot is produced and nothing is returned

`plot.ggmix_gic` *Plot the Generalised Information Criteria curve produced by* `gic`

Description

Plots the Generalised Information Criteria curve, as a function of the lambda values used

Usage

```
## S3 method for class 'ggmix_gic'
plot(
  x,
  ...,
  sign.lambda = 1,
  type = c("gic", "QQranef", "QQresid", "predicted", "Tukey-Anscombe"),
  s = "lambda.min",
  newy,
  newx
)

plotGIC(x, sign.lambda, lambda.min, ...)
```

Arguments

`x` fitted linear mixed model object of class `ggmix_gic` from the `gic` function

`...` Other graphical parameters to plot

`sign.lambda` Either plot against log(lambda) (default) or its negative if sign.lambda=-1

`type` `gic` returns a plot of the GIC vs. log(lambda). `QQranef` return a qqplot of the random effects. `QQresid` returns a qqplot of the residuals which is $y - X\beta - b_i$ where b_i is the subject specific random effect. `predicted` returns a plot of the predicted response ($X\beta$ + b_i) vs. the observed response, where b_i is the subject specific random effect. `Tukey-Anscombe` returns a plot of the residuals vs. fitted values ($X\beta$)

`s` Value of the penalty parameter `lambda` at which predictions are required. Default is the value `s="lambda.min"`. If `s` is numeric, it is taken as the value of `lambda` to be used. Must be a single value of the penalty parameter `lambda` at which coefficients will be extracted via the `coef` method for objects of class `ggmix_gic`. If more than one is supplied, only the first one will be used.

`newy` the response variable that was provided to `ggmix`. this is only required for `type="QQresis"`, `type="Tukey-Anscombe"` and `type="predicted"`

`newx` matrix of values for `x` at which predictions are to be made. Do not include the intercept. this is only required for `type="QQresis"`, `type="Tukey-Anscombe"` and `type="predicted"`

`lambda.min` the value of lambda which minimizes the gic

Details

A plot is produced, and nothing is returned.

Value

plot depends on the type selected

See Also

`gic`

Examples

```
data("admixed")
fit <- ggmix(x = admixed$xtrain,
             y = admixed$ytrain,
             kinship = admixed$kin_train)
hdbic <- gic(fit)

# plot solution path
plot(fit)

# plot HDBIC curve as a function of lambda
plot(hdbic)
```

---

predict.ggmix_fit *Make predictions from a* ggmix_fit *object*

Description

Similar to other predict methods, this functions predicts fitted values, coefficients and more from a fitted ggmix_fit object.

Usage

```
## S3 method for class 'ggmix_fit'
predict(
  object,
  newx,
  s = NULL,
  type = c("link", "response", "coefficients", "all", "nonzero", "individual"),
  covariance,
  ...
)

## S3 method for class 'ggmix_fit'
coef(object, s = NULL, type, ...)
```

Arguments

object Fitted ggmix_fit model object from the ggmix function

newx matrix of values for x at which predictions are to be made. Do not include the intercept. Must be a matrix. This argument is not used for type = c("coefficients","nonzero","all"). This matrix must have the same number of columns originally supplied to the ggmix fitting function.

s Value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence used to create the model.

`type` Type of prediction required. Type `"link"` gives the fitted values $X\beta$. Type `"response"` is equivalent to type `"link"`. Type `"coefficients"` computes the coefficients at the requested values for `s` and returns the regression coefficients only, including the intercept. Type `"all"` returns both the regression coefficients and variance components at the requested value of `s`. Type `"nonzero"` returns a 1 column matrix of the the nonzero fixed effects, as well as variance components for each value of `s`. If more than one `s` is provided, then `"nonzero"` will return a list of 1 column matrices. Default: "link"

`covariance` covariance between test and training individuals. if there are q testing individuals and N-q training individuals, then this covariance matrix is q x (N-q)

`...` additional arguments to pass to predict function

Details

`s` is the new vector at which predictions are requested. If `s` is not in the lambda sequence used for fitting the model, the predict function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right lambda indices. `coef(...)` is equivalent to `predict(ggmix_fit, type="coefficients",...)`. To get individual level predictions at each value of lambda, you must provide the lambda sequence to the s argument. You can pass either a ggmix_fit or ggmix_gic object. See examples for more details.

Value

The object returned depends on type.

Examples

```
data("admixed")
fitlmm <- ggmix(x = admixed$xtrain, y = admixed$ytrain,
                kinship = admixed$kin_train,
                estimation = "full")
bicGGMIX <- gic(fitlmm,
                an = log(length(admixed$ytrain)))
plot(bicGGMIX)
coef(bicGGMIX, s = "lambda.min")
yhat_test <- predict(bicGGMIX, s="lambda.min",
                     newx = admixed$xtest, type = "individual",
                     covariance = admixed$kin_test_train)
cor(yhat_test, admixed$ytest)
yhat_test_population <- predict(bicGGMIX, s="lambda.min",
                                newx = admixed$xtest,
                                type = "response")
```

`predict.ggmix_gic` *Make predictions from a* `ggmix_gic` *object*

Description

This function makes predictions from a ggmix_gic object, using the stored "ggmix_fit" object, and the optimal value chosen for lambda using the gic.

Usage

```
## S3 method for class 'ggmix_gic'
predict(object, newx, s = c("lambda.min"), ...)

## S3 method for class 'ggmix_gic'
coef(object, s = c("lambda.min"), type, ...)
```

Arguments

object	fitted ggmix_gic object

newx	matrix of values for x at which predictions are to be made. Do not include the intercept. Must be a matrix. This argument is not used for type = c("coefficients","nonzero","all"). This matrix must have the same number of columns originally supplied to the ggmix fitting function.

s	Value(s) of the penalty parameter lambda at which predictions are required. Default is the value s="lambda.min" can be used. If s is numeric, it is taken as the value(s) of lambda to be used.

...	other arguments passed to predict.ggmix_fit

type	Type of prediction required. Type "link" gives the fitted values $X\beta$. Type "response" is equivalent to type "link". Type "coefficients" computes the coefficients at the requested values for s and returns the regression coefficients only, including the intercept. Type "all" returns both the regression coefficients and variance components at the requested value of s. Type "nonzero" returns a 1 column matrix of the the nonzero fixed effects, as well as variance components for each value of s. If more than one s is provided, then "nonzero" will return a list of 1 column matrices. Default: "link"

Details

This function makes it easier to use the results of gic chosen model to make a prediction.

Value

The object returned depends the ... argument which is passed on to the predict method for ggmix_fit objects.

See Also

predict.ggmix_fit

`print.ggmix_fit` *Print Method for Objects of Class* `ggmix_fit`

Description

print method for objects of class `ggmix_fit`

Usage

```
## S3 method for class 'ggmix_fit'
print(x, ..., digits = max(3, getOption("digits") - 3))

## S3 method for class 'ggmix_gic'
print(x, ..., digits = max(3, getOption("digits") - 3))
```

Arguments

`x`	an object of class objects of class `ggmix_fit`
`...`	other arguments passed to `print`
`digits`	significant digits in printout. Default: `max(3,getOption("digits") - 3)`

Value

The call that produced the object `x` is printed, followed by a three-column matrix with columns `Df`, `%Dev`, and and `Lambda`. The `Df` columns correspond to the number of nonzero coefficients including variance components. `%dev` is the percent deviance explained (relative to the null deviance). `Lambda` is the sequence of converged tuning parameters.

See Also

`ggmix`

`ranef` *Extract Random Effects*

Description

Generic function for extracting the random effects. This is the same generic (and same name) defined in the nlme and lme4 package.

Usage

```
ranef(object, ...)

random.effects(object, ...)

## Default S3 method:
random.effects(object, ...)

## Default S3 method:
ranef(object, ...)

## S3 method for class 'ggmix_gic'
ranef(object, s = "lambda.min", ...)
```

Arguments

object	any fitted model object from which random effects estimates can be extracted. Currently supports "ggmix_gic" objects outputted by the gic function
...	other parameters. currently ignored
s	Value(s) of the penalty parameter lambda at which predictions are required. Default is the value s="lambda.min" can be used. If s is numeric, it is taken as the value(s) of lambda to be used.

Details

For objects of class "ggmix_gic", this function returns the subject-specific random effect value for the model which minimizes the GIC using the maximum a posteriori principle

Value

a numeric vector of length equal to the number of observations of subject-specific random effects

See Also

gic

Examples

```
data("admixed")
fit <- ggmix(x = admixed$xtrain, y = admixed$ytrain,
            kinship = admixed$kin_train)
gicfit <- gic(fit)
# random effect at selected value of lambda
plot(ggmix::ranef(gicfit))
# random effects at specific values of lambda
head(ggmix::ranef(gicfit, s = c(0.1,0.2)))
```

sigma2lasso *Estimation of Sigma2 for Linear Mixed Model with Lasso Penalty*

Description

`sigma2lasso` estimates the value of the sigma2 for the linear mixed model with lasso penalty

Usage

```
sigma2lasso(ggmix_object, ...)

## Default S3 method:
sigma2lasso(ggmix_object, ...)

## S3 method for class 'fullrank'
sigma2lasso(ggmix_object, ..., n, beta, eta)
```

Arguments

`ggmix_object`	A ggmix_object object of class `lowrank` or `fullrank`
`...`	Extra parameters. Currently ignored.
`n`	number of observations
`beta`	current estimate of the beta parameter including the intercept. this should be of length p+1, where p is the number of variables.
`eta`	current estimate of the eta parameter

Value

A decreasing sequence of tuning parameters

Note

There is a closed form solution for sigma^2, given beta and eta. This function isn't meant to be called directly by the user.

See Also

`ggmix`

Index