Package 'epiGWAS'

May 21, 2019

Title Robust Methods for Epistasis Detection

Version 1.0.1

Description Functions to perform robust epistasis detection in genome-wide association studies, as described in Slim et al. (2018) <doi:10.1101/442749>. The implemented methods identify pairwise interactions between a particular target variant and the rest of the genotype, using a propensity score approach. The propensity score models the linkage disequilibrium between the target and the rest of the genotype. All methods are penalized regression approaches, which differently incorporate the propensity score to only recover the synergistic effects between the target and the genotype.

Depends R (>= 3.4.0)

Imports matrixStats, DescTools, glmnet, SNPknock

Suggests foreach, iterators, precrec, parallel, doParallel, bigmemory, biglasso, testthat, knitr, rmarkdown, kableExtra

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Lotfi Slim [aut, cre],
Clément Chatelain [ctb],
Chloé-Agathe Azencott [ctb],
Jean-Philippe Vert [ctb]

Maintainer Lotfi Slim <lotfi.slim@mines-paristech.fr>

Repository CRAN

Date/Publication 2019-05-21 21:20:04 UTC

R topics documented:

`BOOST` *Implements BOOST SNP-SNP interaction test*

Description

For a pair of SNPs (X_1, X_2) and a binary phenotype Y, the `BOOST` function computes the ratio of maximum log-likelihoods for two models: the full model and the main effects model. Mathematically speaking, the full model is a logistic regression model with both main effects and interaction terms $(X_1, X_2, X_1 \times X_2)$. The main effects model is a logistic regression model with only (X_1, X_2) as covariates. Since we are interested in the synergies with a single variant, we do not implement the initial sure screening stage in BOOST which filters out non-significant pairs.

Usage

```
BOOST(A, X, Y, ncores = 1)
```

Arguments

`A`	target variant. The SNP A is encoded as 0, 1, 2.
`X`	genotype matrix (excluding A). The only accepted SNP values are also 0, 1 and 2.
`Y`	observed phenotype. Binary or two-level factor.
`ncores`	number of threads (default 1)

Value

The interaction statistic between each column in X and A

See Also

The webpage http://bioinformatics.ust.hk/BOOST.html provides additional details about the BOOST software

Examples

```
X <- matrix((runif(500, min = 0, max = 1) < 0.5) +
    (runif(500, min = 0, max = 1) < 0.5), nrow = 50)
A <- (runif(50, min = 0, max = 1) < 0.5) + (runif(50, min = 0, max = 1) < 0.5)
Y <- runif(50, min = 0, max = 1) < 1/(1+exp(-.5 * A * X[, 3] + .25 * A * X[, 7]))
BOOST(A, X, Y)
```

---

cond_prob *Computes the propensity scores*

Description

In this function, and for each sample, we compute both propensity scores $P(A = 1|X)$ and $P(A = 0|X)$. The application of the forward algorithm on the passed hmm allows us to estimate the joint probability of (A, X), for all values of the target variant A = 0, 1, 2. The Bayes formula yields the corresponding conditional probabilities. Depending on the binarization rule, we combine them to obtain the propensity scores.

Usage

```
cond_prob(X, target_name, hmm, binary = FALSE, ncores = 1)
```

Arguments

X	genotype matrix. Make sure to assign colnames(X) beforehand.
target_name	target variant name
hmm	fitted parameters of the fastPHASE hidden Markov model. The HMM model is to be fitted with the fast_HMM function.
binary	if TRUE, the target SNP values 0 and (1,2) are respectively mapped to 0 and 1. That describes a dominant mechanism. Otherwise, if FALSE, we encode a recessive mechanism where the values 0 and 1 respectively map to (0,1) and 2.
ncores	number of threads (default 1)

Value

Two-column propensity score matrix. The first column lists the propensity score $P\,(A = 0|X)$, while the second gives $P\,(A = 1|X)$.

See Also

fast_HMM

Examples

```
p <- 3 # Number of states
K <- 2 # Dimensionality of the latent space

p_init <- rep(1 / K, K)
p_trans <- array(runif((p - 1) * K * K), c(p - 1, K, K))
# Normalizing the transition probabilities
for (j in seq_len(p - 1)) {
  p_trans[j, , ] <- p_trans[j, , ] / (matrix(rowSums(p_trans[j, , ]), ncol = 1) %*% rep(1, K))
}

p_emit <- array(stats::runif(p * 3 * K), c(p, 3, K))
# Normalizing the emission probabilities
for (j in seq_len(p)) {
  p_emit[j, , ] <- p_emit[j, , ] / (matrix(rep(1, 3), ncol = 1) %*% colSums(p_emit[j, , ]))
}

hmm <- list(pInit = p_init, Q = p_trans, pEmit = p_emit)

n <- 2
X <- matrix((runif(n * p, min = 0, max = 1) < 0.4) +
            (runif(n * p, min = 0, max = 1) < 0.4),
            nrow = 2, dimnames = list(NULL, paste0("SNP_", seq_len(p))))

cond_prob(X, "SNP_2", hmm, ncores = 1, binary = TRUE)
```

---

epiGWAS *Runs a selection of epistasis detection methods in a joint manner*

Description

This function is a wrapper for the different epistasis detection methods implemented in this package. If `methods` is `"all"`, we run `OWL` and the four modified outcome approaches. Otherwise, we run a selection of those methods. In this case, the `methods` argument is a character vector with its entries being the names of the functions to call.

Usage

```
epiGWAS(A, X, Y, propensity, methods = "all", parallel = TRUE,
  shift = 0.1, ...)
```

Arguments

A	target variant
X	rest of the genotype
Y	phenotype
propensity	propensity scores
methods	character vector for the epistasis detection methods to call
parallel	whether to perform support estimation in a parallelized fashion for the modified outcome family of methods
shift	regularization parameter for shifted_outcome
...	additional arguments to be passed to stabilityGLM or stabilityBIG

Value

list of numeric vectors. Each vector corresponds to the auc scores of a particular method in methods.

See Also

OWL, modified_outcome, shifted_outcome, normalized_outcome and robust_outcome

Examples

```
n <- 20
p <- 8
X <- matrix((runif(n * p) < 0.4) + (runif(n * p) < 0.4),
             ncol = p, nrow = n) # SNP matrix
A <- rbinom(n, 1, 0.3)
propensity <- runif(n, min = 0.4, max = 0.8)
Y <- rnorm(n)
aucs <- epiGWAS(A, X, Y, propensity, lambda_min_ratio = 0.01, parallel = FALSE,
                 shift = 0.2, n_subsample = 1, short = TRUE, eps = 1e-4,
                 methods = c("normalized_outcome", "robust_outcome"))

names(aucs)
```

---

fast_HMM *Fits a HMM to a genotype dataset by calling fastPHASE*

Description

In this function, we fit the fastPHASE hidden Markov model (HMM) using the EM algorithm. The fastPHASE executable is required to run fast_HMM. It can be downloaded from the following web page: http://scheet.org/software.html

Usage

```
fast_HMM(X, out_path = NULL, X_filename = NULL,
  fp_path = "bin/fastPHASE", n_state = 12, n_iter = 25)
```

Arguments

X	genotype matrix
out_path	prefix for the fitted parameters filenames. If NULL, the files are saved in a temporary directory.
X_filename	filename for the fastPHASE-formatted genotype file. If NULL, the file is created in a temporary directory.
fp_path	path to the fastPHASE executable
n_state	dimensionality of the latent space
n_iter	number of iterations for the EM algorithm

Details

Because of the quadratic complexity of the forward algorithm in terms of the dimensionality of the latent space n_state, we recommend setting this parameter to 12. Choosing a higher number does not result in a dramatic increase of performance. An optimal choice for the number of iterations for the EM algorithm is between 20 and 25.

Value

Fitted parameters of the fastPHASE HMM. They are grouped in a list with the following fields: pInit for the initial marginal distribution, the three-dimensional array Q for the transition probabilities and finally pEmit, another three-dimensional array for the emission probabilities

References

Scheet, P., & Stephens, M. (2006). A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. American Journal of Human Genetics, 78(4), 629–644.

Examples

```
p <- 50
n <- 100
genotypes <- matrix((runif(n * p, min = 0, max = 1) < 0.5) +
            (runif(n * p, min = 0, max = 1) < 0.5),
            nrow = n, dimnames = list(NULL, paste0("SNP_", seq_len(p))))

hmm <- fast_HMM(genotypes, fp_path = "/path/to/fastPHASE",
                n_state = 4, n_iter = 10)
```

forward *Applies the forward algorithm to a genotype dataset*

Description

Applies the forward_sample function to each row in X. If the ncores > 1, the function calling is performed in a parallel fashion to reduce the running time. The parallelization backend is doParallel. If the latter package is not installed, the function switches back to single-core mode.

Usage

```
forward(X, p_init, p_trans, p_emit, ncores = 1)
```

Arguments

X	genotype matrix. Each row corresponds to a separate sample
p_init	marginal distributions for the first hidden state
p_trans	3D dimensional array for the transition probabilities
p_emit	3D dimensional array for the emission probabilities
ncores	number of threads (default 1)

Value

A vector of log probabilities

References

Rabiner, Lawrence R. 'A tutorial on hidden Markov models and selected applications in speech recognition.' Proceedings of the IEEE 77.2 (1989): 257-286.

Examples

```
p <- 3 # Number of states
K <- 2 # Dimensionality of the latent space

p_init <- rep(1 / K, K)
p_trans <- array(runif((p - 1) * K * K), c(p - 1, K, K))
# Normalizing the transition probabilities
for (j in seq_len(p - 1)) {
  p_trans[j, , ] <- p_trans[j, , ] / (matrix(rowSums(p_trans[j, , ]), ncol = 1) %*% rep(1, K))
}

p_emit <- array(stats::runif(p * 3 * K), c(p, 3, K))
# Normalizing the emission probabilities
for (j in seq_len(p)) {
  p_emit[j, , ] <- p_emit[j, , ] / (matrix(rep(1, 3), ncol = 1) %*% colSums(p_emit[j, , ]))
}
```

```
n <- 2
X <- matrix((runif(n * p, min = 0, max = 1) < 0.4) +
            (runif(n * p, min = 0, max = 1) < 0.4), nrow = 2)

# Computing the joint log-probabilities
log_prob <- forward(X, p_init, p_trans, p_emit)
```

`forward_sample` *Applies the forward algorithm to a single observation*

Description

The forward algorithm is applied in order to compute the joint probability for the observation `x`. For hidden Markov models, the forward algorithm is an attractive option because of its linear complexity in the number of hidden states. However, the complexity becomes quadratic in terms of the dimensionality of the latent space.

Usage

```
forward_sample(x, p_init, p_trans, p_emit)
```

Arguments

`x`	one-sample genotype
`p_init`	marginal distributions for the first hidden state
`p_trans`	3D dimensional array for the transition probabilities
`p_emit`	3D dimensional array for the emission probabilities

Details

Our implementation of the forward algorithm makes use of the LogSumExp transformation for increased numerical stability.

Value

Joint probability for the state x in a log form

References

Rabiner, Lawrence R. 'A tutorial on hidden Markov models and selected applications in speech recognition.' Proceedings of the IEEE 77.2 (1989): 257-286.

Examples

```
p <- 3 # Number of states
K <- 2 # Dimensionality of the latent space

p_init <- rep(1 / K, K)
p_trans <- array(runif((p - 1) * K * K), c(p - 1, K, K))
# Normalizing the transition probabilities
for (j in seq_len(p - 1)) {
  p_trans[j, , ] <- p_trans[j, , ] / (matrix(rowSums(p_trans[j, , ]), ncol = 1) %*% rep(1, K))
}

p_emit <- array(stats::runif(p * 3 * K), c(p, 3, K))
# Normalizing the emission probabilities
for (j in seq_len(p)) {
  p_emit[j, , ] <- p_emit[j, , ] / (matrix(rep(1, 3), ncol = 1) %*% colSums(p_emit[j, , ]))
}

X <- (runif(p, min = 0, max = 1) < 0.5) + (runif(p, min = 0, max = 1) < 0.5)

# Computing the joint log-probabilities
log_prob <- forward_sample(X, p_init, p_trans, p_emit)
```

`genotypes` *Simulated genotypes*

Description

We simulated 300 unphased European genotypes. For that matter, we used the HAPGEN2 software and the 1000 genome phase 3 reference data. The simulated regions are located on the 22 chromosome between the nucleotide positions 16061016 (rs9617528) and 19976834 (rs887201). Only the markers of the Affymetrix 500K are included.

Usage

```
data(genotypes)
```

Format

An integer matrix with 300 rows and 450 columns. The SNP rsIDs and positions, in addition to their reference and alternate alleles, are combined in `colnames(X)`.

References

Su, Z., Marchini, J., & Donnelly, P. (2011). HAPGEN2: Simulation of multiple disease SNPs. Bioinformatics, 27(16), 2304–2305.

Consortium, T. 1000 G. P., Auton, A., Abecasis, G. R., Altshuler (Co-Chair), D. M., Durbin (Co-Chair), R. M., Abecasis, G. R., ... Abecasis, G. R. (2015). A global reference for human genetic variation. Nature, 526, 68.

Examples

```
data(genotypes)

hmm <- fast_HMM(genotypes, fp_path = '/path/to/fastPHASE')
```

`gen_model` *Samples effect sizes for the disease model*

Description

The generated disease model is the list of effect size coefficients. The list comprises the following fields: 'syner', 'marg' and 'inter'. 'syner' is itself a list of numeric vectors with two entries named 'A0' and 'A1'. 'A0' refers to the vector of effect sizes when the target variant $A = 0$. Similarly, 'A1' refers to the vector of effect sizes in the case $A = 1$. The two other entries 'marg' and 'inter' are, respectively, the marginal and epistatic effect sizes. The effect sizes are independent and normally-distributed. The `mean` parameter is either a list of vectors or a vector of length 4. If `mean` is a vector, then the effect sizes for each type of effects have the same mean. Otherwise, the corresponding vector in the list specifies their individual means. The same logic applies to `sd`, the standard deviation parameter. For coherence, the parameters `mean` and `sd` are encoded in the same order as the output.

Usage

```
gen_model(nX, nY, nZ12, mean = rep(0, 4), sd = rep(1, 4))
```

Arguments

`nX`	number of SNPs interacting with the target variant
`nY`	number of SNPs with marginal effects
`nZ12`	number of SNP pairs with epistatic effects
`mean`	vector or list of means
`sd`	vector or list of standard deviations

Value

a list of vectors corresponding to the effect size coefficients.

Examples

```
effect_sizes <- gen_model(nX = 2, nY = 2, nZ12 = 1,
                          mean = rep(1, 4), sd = rep(1, 4))
```

maf *SNP minor allele frequencies*

Description

maf contains the minor allele frequencies for the 450 SNPs in the genotypes dataset

Usage

```
data(maf)
```

Format

a numeric vector

Examples

```
data(maf)
all((maf <= 0.5) & (maf >= 0))
```

merge_cluster *Merges a number of clusters around the target*

Description

The purpose of the function merge_cluster is to define an enlarged window of SNPs which are in linkage disequilibrium with the target. It replaces the indices of neighbor clusters with center, the target cluster index. The neighborhood is defined according to the parameter k (see Arguments for more details). Subsequently, we filter them out for the estimate of the propensity scores.

Usage

```
merge_cluster(clusters, center, k = 3)
```

Arguments

clusters	vector of cluster memberships. Typically, the output of cutree
center	the target variant cluster
k	vector or integer. if k is given as a vector, it corresponds to the cluster indices to be updated. Otherwise, if k is an integer, the cluster indices to be updated lie between center-k and center+k.

Value

The updated cluster membership vector. The cluster indexing is also updated so that the maximum cluster index is equal to the total number of clusters after merging.

Examples

```
hc <- hclust(dist(USArrests))
clusters <- cutree(hc, k = 10)
merge_cluster(clusters, center=5, k=2)
```

`modified_outcome` *Implements the modified outcome approach*

Description

In the modified outcome approach, we estimate the risk difference $E[Y|A=1,X]-E[Y|A=0,X]$. The risk difference measures the synergy between `A` and the set of covariates in `X`. For genome-wide association studies, it can be interpreted as a pure epistatic term. However, for a single sample, we only observe one of the two possibilities A=1 or A=0, making the direct estimate of the risk difference impossible. Through propensity scores, modified outcome was proposed as a solution to this problem. The risk difference is recovered by constructing a modified outcome that combines A, Y and the propensity score $\pi(A|X)$: $Y \times \left[\frac{A}{\pi(A=1|X)} - \frac{1-A}{1-\pi(A=1|X)}\right]$. The use of `stabilityGLM` or `stabilityBIG` for the modified outcome regression allows us to recover the interacting components within `X`.

Usage

```
modified_outcome(A, X, Y, propensity, parallel = FALSE, ...)
```

Arguments

`A`	target variant
`X`	rest of the genotype
`Y`	phenotype
`propensity`	propensity scores vector/matrix. If given as a matrix, the first column is $\pi(A = 0\|X)$ while the second is $\pi(A = 1\|X)$
`parallel`	whether to perform support estimation in a parallelized fashion with the `stabilityBIG` function
`...`	additional arguments to be passed to `stabilityGLM` or `stabilityBIG`

Value

a vector containing the area under the stability selection path for each variable in `X`

References

Rosenbaum, Paul R., and Donald B. Rubin. 'The central role of the propensity score in observational studies for causal effects.' Biometrika 70.1 (1983): 41-55.

Examples

```
n <- 30
p <- 10
X <- matrix((runif(n * p) < 0.5) + (runif(n * p) < 0.5), ncol = p, nrow = n)
A <- (runif(n, min = 0, max = 1) < 0.3)
propensity <- runif(n, min = 0.4, max = 0.8)
Y <- runif(n) < 1/ (1 + exp(- 2 * X[, 5, drop = FALSE]))
auc_scores <- modified_outcome(A, X, Y, propensity,
                               ncores = 1, parallel = TRUE, n_subsample = 1)
```

`normalized_outcome` *Implements the normalized modified outcome approach*

Description

Normalized modified outcome is an improvement to `modified_outcome`. Its large-sample variance is lower than the original modified outcome approach. The only difference between the two methods lies in the normalization of the propensity scores. The inverses of the propensity scores $1/\pi(A = 1|X)$ and $1/\pi(A = 0|X)$ are respectively normalized by their sum $\sum_i 1/\pi(A_i = 1|X_i)$ and $\sum_i 1/\pi(A_i = 0|X_i)$.

Usage

```
normalized_outcome(A, X, Y, propensity, parallel = FALSE, ...)
```

Arguments

`A`	target variant
`X`	rest of the genotype
`Y`	phenotype
`propensity`	propensity scores
`parallel`	whether to perform support estimation in a parallelized fashion
`...`	additional arguments to be passed to `stabilityGLM` or `stabilityBIG`

Value

a vector containing the area under the stability selection path for each variable in `X`

Examples

```
n <- 30
p <- 10
X <- matrix((runif(n * p) < 0.5) + (runif(n * p) < 0.5),
            ncol = p, nrow = n) # SNP matrix
A <- (runif(n) < 0.3)
propensity <- runif(n, min = 0.4, max = 0.8)
```

```
Y <- runif(n) < 0.4
normalized_scores <- normalized_outcome(A, X, Y, propensity,
                                lambda_min_ratio = 0.02 , n_subsample = 1)
```

OWL *Implements the outcome weighted learning approach*

Description

To recover the synergistic interactions between the target A and the rest of the genotype X, OWL formulates a weighted binary classification problem. The outcome is the mapping of A to {0,1}. The covariates are X. The propensity scores and the phenotypes are combined in the sample weights $Y/\pi(A|X)$. For binary phenotypes, OWL is a case-only approach. The approach also accommodates nonnegative continuous phenotypes.

Usage

```
OWL(A, X, Y, propensity, ...)
```

Arguments

A	target variant. If not binary, the variable A must be encoded as either (0, 1) or (0, 1, 2).
X	rest of the genotype
Y	phenotype (binary or continuous)
propensity	propensity scores (a vector or a two-column matrix)
...	additional arguments to stabilityGLM

Details

For continuous phenotypes, if the outcome Y is not nonnegative, it is translated to make it nonnegative.

Value

a vector containing the area under the stability selection path for each variable in X

References

Zhao, Y., Zeng, D., Rush, A. J., & Kosorok, M. R. (2012). Estimating Individualized Treatment Rules Using Outcome Weighted Learning. Journal of the American Statistical Association, 107(499), 1106–1118.

Examples

```
n <- 30
p <- 10
X <- matrix((runif(n * p) < 0.5) + (runif(n * p) < 0.5), ncol = p, nrow = n)
A <- (runif(n, min = 0, max = 1) < 0.3)
propensity <- runif(n, min = 0.4, max = 0.8)
Y <- runif(n, min = 0, max = 1) < 1/ (1 + exp(-X[, c(1, 7)] %*% rnorm(2)))
OWL(A, X, Y, propensity, short = FALSE, n_lambda = 50, n_subsample = 1)
```

---

propensity *propensity scores*

Description

To obtain the scores for the samples in genotypes, we first remove all SNPs in the genomic interval [18038910, 18288361] to alleviate linkage disequilibrium around the target SNP rs2535708 (position 18184169). Afterwards, we apply fast_HMM followed by cond_prob. The results are stored in propensity.

Usage

```
data(propensity)
```

Format

a numeric vector

Examples

```
data(propensity)
```

---

robust_outcome *Implements the robust modified outcome approach*

Description

A key feature of robust_outcome is its resilience to the misspecification of propensity scores, which is a major limitation of classical modified outcome approaches. Except for shifted_outcome, all of the modified outcome approaches belong to a parameterized class of unbiased estimators for the risk difference term $E[Y|A=1,X]-E[Y|A=0,X]$. Within that class, robust modified outcome is the approach with the least large-sample variance. For more details about this approach, see Lunceford and Davidian (2004)

Usage

```
robust_outcome(A, X, Y, propensity, parallel = FALSE, ...)
```

Arguments

A	target variant
X	rest of the genotype
Y	phenotype
propensity	propensity scores
parallel	whether to perform support estimation in a parallelized fashion
...	additional arguments to be passed to stabilityGLM or stabilityBIG

Value

a vector containing the area under the stability selection path for each variable in X

References

Lunceford, J. K., & Davidian, M. (2004). Stratification and weighting via the propensity score in estimation of causal treatment effects: A comparative study. Statistics in Medicine, 23(19), 2937–2960.

Examples

```
n <- 30
p <- 10
X <- matrix((runif(n * p) < 0.4) + (runif(n * p) < 0.4),
            ncol = p, nrow = n) # SNP matrix
A <- rbinom(n, 1, 0.3)
propensity <- runif(n, min = 0.4, max = 0.8)
Y <- runif(n) < 0.4
robust_scores <- robust_outcome(A, X, Y, propensity,
                                lambda_min_ratio = 0.01 , n_subsample = 1)
```

---

sample_SNP *Samples causal SNPs with different effect types*

Description

The sampled SNPs are combined in a list of character vectors with the following fields: target, marginal, inter1 and inter2. Through the parameters overlap_marg and overlap_inter, the synergistic SNPs with the target can have additional marginal and epistatic effects. The SNPs are consecutively sampled in the following order: target, marginal, inter1 and inter2. For each SNP, we iteratively sample until the picked SNP candidate meets the constraints defined by thresh_MAF and window_size (see arguments for more details) or until the maximum number of resamplings is reached. To avoid duplication of effects, we sample at most one SNP per cluster.

Usage

```
sample_SNP(nX, nY, nZ12, clusters, MAF, thresh_MAF = 0.2,
  window_size = 3, overlap_marg = 0, overlap_inter = 0,
  max_iter = 10000)
```

Arguments

nX	number of SNPs interacting with the target variant
nY	number of SNPs with marginal effects
nZ12	number of SNP pairs with epistatic effects
clusters	vector of cluster memberships. Typically, the output of cutree. For ease of identification, the SNP IDs in names(clusters) are mandatory.
MAF	vector of minor allele frequencies. The order of the SNPs in MAF is identical to that in clusters.
thresh_MAF	lower-bound on the minor allele frequencies of causal SNPs. Rare variants are inherently difficult to recover. Assessing the retrieval performance on common variants better reflects the true performance of the epistasis detection algorithm.
window_size	in number of clusters. Beside the target variant, the causal SNPs are sampled outside of a window centered around the target. On each side of the target variant, the width of the window is window_size.
overlap_marg	number of SNPs with both synergistic effects with the target and marginal effects
overlap_inter	number of SNPs with both synergistic effects with the target and additional epistatic effects
max_iter	maximum number of sampling rejections for each SNP. If exceeded, the function generates an error

Value

list of character vectors containing to the causal SNP IDs. The output list entries are: target, marginal, inter1 and inter2. An epistatic pair is obtained from the combination of two SNPs with identical positions in inter1 and inter2.

Warning

Make sure to supply the SNP IDs in names(clusters). The SNPs in the output list are referenced by their names.

Examples

```
clusters <- rep(seq_len(10), each = 3)
names(clusters) <- paste0("SNP_", seq_along(clusters))
MAF <- runif(length(clusters), min = 0.1, max = 0.5)

sample_SNP(nX = 2, nY = 2, nZ12 = 1, clusters, MAF, thresh_MAF = 0.2,
           window_size = 2, overlap_marg = 1, overlap_inter = 0)
```

`shifted_outcome` *Implements the shifted modified outcome approach*

Description

Shifted modified outcome is an improvement to modified outcome. It is a heuristic which consists in the addition of of a small translation term to the inverse of the propensity score. The goal is to avoid numerical instability due to low propensity scores values. More precisely, the inverses of the propensity scores become $1/(\pi(A|X) + \xi)$. We recommend keeping the default value of the parameter ξ at 0.1.

Usage

```
shifted_outcome(A, X, Y, propensity, parallel = FALSE, shift = 0.1,
  ...)
```

Arguments

`A`	target variant
`X`	rest of the genotype
`Y`	phenotype
`propensity`	propensity scores
`parallel`	whether to perform support estimation in a parallelized fashion
`shift`	regularization term to be added to the propensity scores to avoid numerical stability
`...`	additional arguments to be passed to `stabilityGLM` or `stabilityBIG`

Value

a vector containing the area under the stability selection path for each variable in `X`

Examples

```
n <- 30
p <- 10
X <- matrix((runif(n * p) < 0.5) + (runif(n * p) < 0.5),
            ncol = p, nrow = n) # SNP matrix
A <- (runif(n) < 0.3)
propensity <- runif(n, min = 0.4, max = 0.8)
Y <- runif(n) < 1/ (1 + exp(-X[, 2, drop = FALSE]))
shifted_scores <- shifted_outcome(A, X, Y, propensity,
                                  shift = 0.1, n_subsample = 1)
```

`sim_phenotype` *Simulates a binary phenotype*

Description

The phenotypes are simulated according to a logistic regression model. Depending on the chosen configuration in `sample_SNP`, the model includes different effect types: synergistic effects with the target, marginal effects and additional epistatic effects. We offer the option to generate a balanced phenotype vector between cases and controls, through the `intercept` parameter.

Usage

```
sim_phenotype(X, causal, model, intercept = TRUE)
```

Arguments

`X`	genotype matrix
`causal`	causal SNPs.
`model`	disease model
`intercept`	binary flag. If `intercept=TRUE`, a non-null intercept is added so that the output is (approximately) balanced between cases and controls.

Value

A vector of simulated phenotypes which are encoded as a two-level factor (TRUE/FALSE).

See Also

`sample_SNP` and `gen_model`

Examples

```
nX <- 5
nY <- 3
nZ12 <- 2
clusters <- rep(seq_len(25), each = 3)
names(clusters) <- paste0("SNP_", seq_along(clusters))
MAF <- runif(length(clusters), min = 0.2, max = 0.5)

n_samples <- 3
X <- matrix((runif(n_samples * length(clusters)) < 0.4) +
            (runif(n_samples * length(clusters)) < 0.4),
            ncol = length(clusters), nrow = n_samples)

colnames(X) <- names(clusters)

causal <- sample_SNP(
 nX, nY, nZ12, clusters, MAF, thresh_MAF = 0.2, window_size = 2,
```

```
 overlap_inter = 0)
model <- gen_model(nX, nY, nZ12, mean = rnorm(4), sd = rep(1, 4))
Y <- sim_phenotype(X, causal, model, intercept = TRUE)
```

`stabilityBIG` *Computes the area under the stability path for all covariates*

Description

This function implements the same model selection technique extensively described in `stabilityGLM`. The sole difference is the use of a different elastic net solver. In this function, we make use of `biglasso`. Thanks to its parallel backend, `biglasso` scales well to high-dimensional GWAS datasets. However, in our case, because of the use of additional backend files, a slight decrease in runtime is to be expected, compared with `stabilityGLM`.

Usage

```
stabilityBIG(X, Y, family = "gaussian", n_subsample = 20,
  n_lambda = 100, lambda_min_ratio = 0.01, eps = 1e-05,
  short = TRUE, ncores = 2)
```

Arguments

`X`	design matrix formatted as a `big.matrix` object
`Y`	response vector
`family`	response type. Either 'gaussian' or 'binomial'
`n_subsample`	number of subsamples for stability selection
`n_lambda`	total number of lambda values
`lambda_min_ratio`	the minimum value of the regularization parameter lambda as a fraction of the maximum lambda, the first value for which the elastic net support is not empty.
`eps`	elastic net mixing parameter (see `stabilityGLM` for more details)
`short`	whether to compute the aucs only on the first half of the stability path. We observed better performance with thresholded paths
`ncores`	number of cores for the `biglasso` solver

Value

a vector grouping the aucs of all covariates within `X`

References

Slim, L., Chatelain, C., Azencott, C.-A., & Vert, J.-P. (2018). Novel Methods for Epistasis Detection in Genome-Wide Association Studies. BioRxiv.

Meinshausen, N., & Bühlmann, P. (2010). Stability selection. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 72(4), 417–473.

Haury, A. C., Mordelet, F., Vera-Licona, P., & Vert, J. P. (2012). TIGRESS: Trustful Inference of Gene REgulation using Stability Selection. BMC Systems Biology, 6.

See Also

biglasso-package

Other support estimation functions: stabilityGLM

Examples

```
n <- 100
p <- 25
X <- bigmemory::as.big.matrix(matrix(runif(n * p), ncol = p))
Y <- runif(n, min = 0, max = 1) < 0.5
aucBIG <- stabilityBIG(X, Y, family = "binomial", short = TRUE,
                       ncores = 1, n_lambda = 200, n_subsample = 1)
```

`stabilityGLM` *Computes the area under the stability path for all covariates*

Description

To perform model selection, this function scores all covariates in `X` according to the area under their stability selection paths. Our model selection procedure starts by dynamically defining a grid for the elastic net penalization parameter λ. To define the grid, we solve the full-dataset elastic net. This yields `n_lambda` log-scaled values between λ_{max} and λ_{min}. λ_{max} is the maximum value for which the elastic net support is not empty. On the other hand, λ_{min} can be derived through `lambda_min_ratio`, which is the ratio of λ_{min} to λ_{max}. The next step is identical to the original stability selection procedure. For each value of λ, we solve `n_subsample` times the same elastic net, though for a different subsample. The subsample is a random selection of half of the samples of the original dataset. The empirical frequency of each covariate entering the support is then the number of times the covariate is selected in the support as a fraction of `n_subsample`. We obtain the stability path by associating each value of λ with the corresponding empirical frequency. The final scores are the areas under the stability path curves. This is a key difference with the original stability selection procedure where the final score is the maximum empirical frequency. On simulations, our scoring technique outperformed maximum empirical frequencies.

Usage

```
stabilityGLM(X, Y, weights = rep(1, nrow(X)), family = "gaussian",
  n_subsample = 20, n_lambda = 100, short = TRUE,
  lambda_min_ratio = 0.01, eps = 1e-05)
```

Arguments

X	input design matrix
Y	response vector
weights	nonnegative sample weights
family	response type. Either 'gaussian' or 'binomial'
n_subsample	number of subsamples for stability selection
n_lambda	total number of lambda values
short	whether to compute the aucs only on the first half of the stability path. We observed better performance for thresholded paths
lambda_min_ratio	ratio of λ_{min} to λ_{max} (see description for a thorough explanation)
eps	elastic net mixing parameter.

Details

For a fixed λ, the L2 penalization is $\lambda \times eps$, while the L1 penalization is $\lambda \times (1 - eps)$. The goal of the L2 penalization is to ensure the uniqueness of the solution. For that reason, we recommend setting eps « 1.

Value

a vector containing the areas under the stability path curves

References

Slim, L., Chatelain, C., Azencott, C.-A., & Vert, J.-P. (2018). Novel Methods for Epistasis Detection in Genome-Wide Association Studies. BioRxiv.

Meinshausen, N., & Bühlmann, P. (2010). Stability selection. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 72(4), 417–473.

Haury, A. C., Mordelet, F., Vera-Licona, P., & Vert, J. P. (2012). TIGRESS: Trustful Inference of Gene REgulation using Stability Selection. BMC Systems Biology, 6.

See Also

glmnet-package

Other support estimation functions: stabilityBIG

Examples

```
# ---- Continuous data ----
n <- 50
p <- 20
X <- matrix(rnorm(n * p), ncol = p)
Y <- crossprod(t(X), rnorm(p))
aucs_cont <- stabilityGLM(X, Y, family = "gaussian", n_subsample = 1,
                          short = FALSE)
```

```
# ---- Binary data ----
X <- matrix(rnorm(n * p), ncol = p)
Y <- runif(n, min = 0, max = 1) < 1/ (1 + exp(-X[, c(1, 7, 15)] %*% rnorm(3)))
weights <- runif(n, min = 0.4, max = 0.8)
aucs_binary <- stabilityGLM(X, Y, weights = weights,
                            n_lambda = 50, lambda_min_ratio = 0.05, n_subsample = 1)
```

`subsample` *Creates multiple subsamples without replacement*

Description

The subsampling is iteratively performed in order to generate multiple subsamples of a predetermined size.

Usage

```
subsample(n, size = n%/%2, n_subsample)
```

Arguments

`n`	original sample size
`size`	subsample size
`n_subsample`	total number of subsamples

Value

a matrix of indices with `size` rows and `n_subsample` columns.

Examples

```
n <- 50 # Total number of samples
n_subsample <- 10 # Number of subsamples

sub_matrix <- subsample(n = n, n_subsample = n_subsample)
```

Index