Package 'aziztest'

November 21, 2020

Type Package

Title Novel Statistical Test for Aberration Enrichment

Version 0.2.1

Author Aziz M. Mezlini [aut,cre,cph]

Maintainer Aziz M. Mezlini <mmezlini@mgh.harvard.edu>

Description Testing for heterogeneous effects in a case-control setting.
The aim here to discover an association that is beyond a mean difference
between all cases and all controls. Instead, the signal of interest here
is present in only a proportion of the cases. This test should be more
powerful than a t-test or Wilcoxon test in this heterogeneous setting.
Please cite the corresponding paper: Mezlini et al. (2020)
<doi:10.1101/2020.03.23.002972>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://www.biorxiv.org/content/10.1101/2020.03.23.002972v2

Imports stats

NeedsCompilation no

Repository CRAN

Date/Publication 2020-11-21 06:00:02 UTC

R topics documented:

`aziz.test` *Statistical test for heterogeneous effects*

Description

Main function running the statistical test looking for heterogeneous effects/ aberration enrichment. Takes a vector of case/control labels (`y`) and a vector of numeric measurements (`x`) to be tested for association with case/control status. For example, in a clinical trial setting `y` can indicate individuals on a drug vs placebo and `x` can be a change in disease severity measurement from baseline. This test will return a p-value indicating drug efficacy and is more powerful than other test in a heterogeneous effects setting. Another usage example is in -Omics data where `y` would indicate disease vs healthy control and `x` could be a gene's expression vector across samples.

Usage

```
aziz.test(
  y,
  x,
  w = NULL,
  rep = 1e+05,
  doall = FALSE,
  eps = 1e-09,
  unidirectional = 0,
  flatten = 0.5,
  ignoremax = 0,
  normmethod = 1,
  novariance = F,
  conservative = T
)
```

Arguments

`y`	A binary vector of sample labels (cases=1, controls=0).
`x`	A numerical vector. Variable tested for association. Preferably continuous
`w`	Default = NULL. Optional numerical vector of weights. 1 means all weights are equal to 1 and only the ordering is considered. If NULL (default), a standardisation of x is used to calculate the weights giving larger weights to aberrations of larger magnitude.
`rep`	Default=100000. Number of permutations to be used to calculate p-values.
`doall`	Default=FALSE. Logical. If TRUE all `rep` permutations are performed. If FALSE only enough permutations are performed to get accurate p-values. Variable that are clearly not associated need only a 100 permutations.
`eps`	Default = 0.000000001. Small numeric value. Standard deviation of the gaussian node added to x before ordering samples. In the case of equalities, this ensures the ordering is not biased. Adjust lower if x has low variability.

`unidirectional` Default = 0. Can be 0, 1 or -1. 0 is for testing both directions of effect. 1 is for testing cases<controls and -1 is for testing cases>controls.

`flatten` Default = 0.5. Numeric value recommended between 0 and 1. If weights are not given, we take the max of flatten and the absolute value of the Z-score of `x` as the weights (Default behavior).

`ignoremax` Default=0. Optional value indicating if we should ignore the first few values when selecting the maximal enrichment score. Alternatively, it can be viewed as the minimal size considered for the aberrant interval.

`normmethod` Default=1. If w=NULL the weights are generated by subtracting the mean and dividing by the standard deviation. If `normmethod=2`, the median and MAD are used instead, for a better treatment of outliers.

`novariance` Default=FALSE. aziz.test is able to detect a difference in variance between cases and controls as an association (when variance of cases is larger than the variance of controls). `novariance=True` changes the behaviour and penalizes scenarios with outliers going both ways in the cases. This will remove the associations that would usually be picked by a Levene test. Consider using this when using unidirectional testing if variance changes between groups are irrelevant in your considered problem. Results in loss of power.

`conservative` Default=TRUE. p-values = b+1/ (1+ #permutations) is the returned value. As described in Phibson 2010: "Permutation p-values should never be zero"

Value

A result object with the following fields: (for clarity use `print_summary`)

es Max enrichment score.

pval Permutation p-value, if permutations were performed.

oddcas Proportion of cases in the aberrant interval driving the max enrichment score. This is described as the proportion r in the main paper.

direction direction of the effect. 1: cases<controls, 2: cases>controls.

oddratio Odds ratio of being in the aberrant interval for cases/controls. Equal to `oddcas` divided by the same calculation on controls.

Other info fields (Can be useful):

esm Max enrichment score in both directions.

esind Index of the Max enrichment score in both directions. can also be interpreted the number of samples in the aberrant interval.

ncas Number of cases in the aberrant interval.

escurve A vector of the computed standardized enrichment scores at all positions.

perm A vector of all max enrichment scores obtained in permutations.

Examples

```
y = c(rep(1,200),rep(0,200))
x = rnorm(400)

res = aziz.test(y,x,rep=100) #run 100 permutations to calculate p-value
print_summary(res)

#Inducing an aberration enrichment signal by perturbing some of the cases
x[1:20]=x[1:20]-3;
res2 = aziz.test(y,x,rep=100)
print_summary(res2)
```

`aziztest` *aziztest: A package for finding associations in heterogeneous setting (aberration enrichment)*

Description

This package contains the statistical test presented in Mezlini et al. (2020) "Finding associations in a heterogeneous setting: Statistical test for aberration enrichment" <https://www.biorxiv.org/content/10.1101/2020.03.23.0029
It is used to detect associations that are beyond the broad pattern of comparing averages between all cases and all controls. Instead it looks for a heterogeneous association where only some of the cases present the signal of interest while the majority are indistinguishable from controls. For example, in a clinical trial setting our test can be used to assess treatment efficacy in a context of heterogeneous treatment effect, where the drug works well on only some of the patients. Another usage example is in -Omics data where a relevant gene's dysregulation is present in only some of the disease cases.

Details

The main function is the `aziz.test()` function used to test for heterogeneous associations/ aberration enrichment.

aziztest extra functions

If you are testing multiple variables at once (such as all genes in a gene expression dataset), you can store the results in a list and the reformat it into an easy to use data.frame using the function `reformat_results()`.

In the context of a large number of variables, calibration can be used to speed up p-value calculation with functions `calibrate_test()` and `get_calibrated_pvalues()`.

`calibrate_test` *Calibration of p-values in a slow multi-hypothesis setting*

Description

Compute the null distribution of test statistics on one gaussian variable. Useful if testing a large number of variables at once since it allows running permutations only once beforehand rather than for every variable. Used in conjunction with "`get_calibrated_pvalues`"

Usage

```
calibrate_test(
  y,
  w = NULL,
  rep = 1e+07,
  doall = TRUE,
  unidirectional = 0,
  flatten = 0.5,
  ignoremax = 0,
  normmethod = 1,
  novariance = F
)
```

Arguments

`y`	A binary vector of sample labels (cases=1, controls=0).
`w`	Default = NULL. Optional numerical vector of weights. 1 means all weights are equal to 1 and only the ordering is considered. If NULL (default), a standardisation of x is used to calculate the weights giving larger weights to aberrations of larger magnitude.
`rep`	Default=100000. Number of permutations to be used to calculate p-values.
`doall`	Default=TRUE. All permutations are performed
`unidirectional`	Default = 0. Can be 0, 1 or -1. 0 is for testing both directions of effect. 1 is for testing cases<controls and -1 is for testing cases>controls.
`flatten`	Default = 0.5. Numeric value recommended between 0 and 1. If weights are not given, we take the max of flatten and the absolute value of the Z-score of `x` as the weights (Default behavior).
`ignoremax`	Default=0. Optional value indicating if we should ignore the first few values when selecting the maximal enrichment score. Alternatively, it can be viewed as the minimal size considered for the aberrant interval.
`normmethod`	Default=1. If w=NULL the weights are generated by subtracting the mean and dividing by the standard deviation. If `normmethod=2`, the median and MAD are used instead, for a better treatment of outliers.

novariance	Default=FALSE. aziz.test is able to detect a difference in variance between cases and controls as an association (when variance of cases is larger than the variance of controls). novariance=True changes the behaviour and penalizes scenarios with outliers going both ways in the cases. This will remove the associations that would usually be picked by a Levene test. Consider using this when using unidirectional testing if variance changes between groups are irrelevant in your considered problem. Results in loss of power.

Value

A vector that can be used in get_calibrated_pvalues()

See Also

get_calibrated_pvalues, aziz.test

Examples

```
y = c(rep(1,200),rep(0,200))
x = rnorm(400)
calibration = calibrate_test(y,rep=100)
es = aziz.test(y,x,rep=0)$es #No need for permutations, p-values computed from calibration
get_calibrated_pvalues(calibration,es)
```

---

get_calibrated_pvalues

Use the calibration of p-values in a slow multi-hypothesis setting

---

Description

Compute the p-values from a single set of permutations obtained from calibrate_test. Useful if testing a large number of variables at once since it allows running permutations only once beforehand rather than for every variable. Used in conjunction with "calibrate_test"

Usage

```
get_calibrated_pvalues(calibration, es1, conservative = T)
```

Arguments

calibration	Output of function calibrate_test()
es1	Max Enrichment score given by function aziz.test() $es. A vector containing the max enrichment scores from many variables is acceptable
conservative	Default=TRUE. p-values = b+1/ (1+ #permutations) is the returned value. As described in Phibson 2010: "Permutation p-values should never be zero"

Value

calibrated p-value(s) corresponding to the max enrichment score(s) given

See Also

calibrate_test, aziz.test

Examples

```
y = c(rep(1,200),rep(0,200))
x = rnorm(400)
calibration = calibrate_test(y,rep=1000)
es = aziz.test(y,x,rep=0)$es #No need for permutations, p-values computed from calibration
get_calibrated_pvalues(calibration,es)
x[1:20]=x[1:20]-2;
es2 = aziz.test(y,x,rep=0)$es
get_calibrated_pvalues(calibration,es2)
```

---

print_details *Print a formatted version of the result details*

Description

Print a formatted version of the result details

Usage

```
print_details(x)
```

Arguments

x	output of the aziz.test() function

See Also

print_summary

print_summary *Print a formatted version of the results*

Description

Print a formatted version of the results for clarity

Usage

```
print_summary(x)
```

Arguments

x	output of the `aziz.test()` function

See Also

`print_details` for more info

reformat_results *Reformats multiple results into one table*

Description

If running multiple variables (and storing in a list), this transform the list of results to one coherent data.frame

Usage

```
reformat_results(res_esa)
```

Arguments

res_esa	listed outputs of multiple calls to `aziz.test()` on multiple variables

Value

A data.frame containing all results in an accessible presentation

`which_aberrant` *Returns which samples are in the aberrant interval defined by* `test.aziz()`

Description

Returns index of samples that are in the aberrant interval defined by `test.aziz()` Can take the same samples or a new set of previously unseen samples

Usage

```
which_aberrant(xi, x, res)
```

Arguments

`xi`	Numerical vector. Can be the same as the tested variable x or it can be a new set of unseen samples.
`x`	Numerical vector of the variable tested by `test.aziz()`
`res`	Result of running `test.aziz()`

Value

indexes of samples in `xi` that are within the aberrant interval

Examples

```
y = c(rep(1,200),rep(0,200))
x = rnorm(400)
#Inducing an aberration enrichment signal by perturbing some of the cases
x[1:20]=x[1:20]-3;
res2 = aziz.test(y,x,rep=20000)
print_summary(res2)
which_aberrant(x,x,res2)
which_aberrant(c(-5,1.5,-2.5,-0.5,2),x,res2)#testing if new values are within the aberrant interval
```

Index