Package ‘adaptest’

July 26, 2019

Title Data-Adaptive Statistics for High-Dimensional Multiple Testing

Version 1.4.0

Author Weixin Cai [aut, cre, cph], Nima Hejazi [aut], Alan Hubbard [ctb, ths]

Maintainer Weixin Cai <wcai@berkeley.edu>

Description Data-adaptive test statistics represent a general methodology for performing multiple hypothesis testing on effects sizes while maintaining honest statistical inference when operating in high-dimensional settings (<doi here>). The utilities provided here extend the use of this general methodology to many common data analytic challenges that arise in modern computational and genomic biology.

Depends R (>= 3.5.0)

License GPL-2


Encoding UTF-8

LazyData true

Imports origami (>= 1.0.0), tmle, calibrate, methods, graphics, stats, utils, SummarizedExperiment

Suggests testthat, Matrix, SuperLearner, earth, gam, nnls, airway, rmarkdown, knitr, BiocStyle

VignetteBuilder knitr

RoxygenNote 6.0.1.9000

biocViews Genetics, GeneExpression, DifferentialExpression, Sequencing, Microarray, Regression, DimensionReduction, MultipleComparison

git_url https://git.bioconductor.org/packages/adaptest

git_branch RELEASE_3_9

git_last_commit 5f46ff4

git_last_commit_date 2019-05-02

Date/Publication 2019-07-26
Description

Computes marginal average treatment effects of a binary point treatment on multi-dimensional outcomes, adjusting for baseline covariates, using Targeted Minimum Loss-Based Estimation. A data-mining algorithm is used to perform biomarker selection before multiple testing to increase power.

Usage

```r
adaptest(Y, A, W = NULL, n_top, n_fold, parameter_wrapper = rank_DE, learning_library = c("SL.glm", "SL.step", "SL.glm.interaction", "SL.gam", "SL.earth"), absolute = FALSE, negative = FALSE, p_cutoff = 0.05, q_cutoff = 0.05)
```

Arguments

- **Y** (numeric vector) - A `data.frame` or matrix of binary or continuous biomarker measures (outcome variables). Alternatively, this will be an object of class `adapTMLE` if the wrapper `bioadaptest` is invoked (n.b., the wrapper is the preferred interface for standard data analytic use-cases arising in computational and genomic biology).
- **A** (numeric vector) - binary treatment indicator: 1 = treatment, 0 = control
- **W** (numeric vector, numeric matrix, or numeric `data.frame`) - matrix of baseline covariates where each column correspond to one baseline covariate and each row corresponds to one observation.
- **n_top** (integer vector) - value for the number of candidate covariates to generate using the data-adaptive estimation algorithm
- **n_fold** (integer vector) - number of cross-validation folds.
parameter_wrapper

(function) - user-defined function that takes input (Y, A, W, absolute, negative) and outputs a (integer vector) containing ranks of biomarkers (outcome variables). For details, please refer to the documentation for `rank_DE`

learning_library

(character vector) - library of learning algorithms to be used in fitting the "Q" and "g" step of the standard TMLE procedure.

absolute

(logical) - whether or not to test for absolute effect size. If FALSE, test for directional effect. This overrides argument negative.

negative

(logical) - whether or not to test for negative effect size. If FALSE = test for positive effect size. This is effective only when absolute = FALSE.

p_cutoff

(numeric) - p-value cutoff (default as 0.05) at and below which to be considered significant. Used in inference stage.

q_cutoff

(numeric) - q-value cutoff (default as 0.05) at and below which to be considered significant. Used in multiple testing stage.

Value

S4 object of class `data_adapt`, sub-classed from the container class `SummarizedExperiment`, with the following additional slots containing data-mining selected biomarkers and their TMLE-based differential expression and inference, as well as the original call to this function (for user reference), respectively.

top_index (integer vector) - indices for the data-mining selected biomarkers

top_colname (character vector) - names for the data-mining selected biomarkers

top_colname_significant_q (character vector) - names for the data-mining selected biomarkers, which are significant after multiple testing stage

DE (numeric vector) - differential expression effect sizes for the biomarkers in `top_colname`

p_value (numeric vector) - p-values for the biomarkers in `top_colname`

q_value (numeric vector) - q-values for the biomarkers in `top_colname`

significant_q (integer vector) - indices of `top_colname` which is significant after multiple testing stage.

mean_rank_top (numeric vector) - average ranking across folds of cross-validation folds for the biomarkers in `top_colname`

folds (origami::folds class) - cross validation object

Examples

```r
set.seed(1234)
data(simpleArray)
simulated_array <- simulated_array
simulated_treatment <- simulated_treatment

adaptest(Y = simulated_array,
         A = simulated_treatment,
         W = NULL,
         n_top = 5,
         n_fold = 3,
         learning_library = 'SL.glm',
         parameter_wrapper = adaptest::rank_DE,
         absolute = FALSE,
         negative = FALSE)
```
adapTMLE-class

Constructor for class adaptmle

**Description**

Constructor for class adaptmle

**Value**

class adaptmle object, sub-classed from SummarizedExperiment.

**Examples**

```r
library(SummarizedExperiment)
library(airway)
data(airway)

example_adaptmle_class <- function(se, n_top = 20, n_fold = 10) {
  call <- match.call(expand.dots = TRUE)
  adaptmle <- .adaptmle(
    SummarizedExperiment::SummarizedExperiment(
      assays = SummarizedExperiment::assay(se),
      colData = SummarizedExperiment::colData(se)
    ),
    call = call,
    folds = list(), # folds (from origami)
    plot_ingredients = list(), # top_colname
    diff_exp = as.numeric(rep(NaN, n_top)), # DE
    p_value = as.numeric(rep(NaN, n_top)), # p_value
    q_value = as.numeric(rep(NaN, n_top)), # q_value
    q_sig = as.numeric(rep(NaN, n_top)), # significant_q
    q_sig_names = list(), # top_colname_significant_q
    rank_mean = as.numeric(rep(NaN, n_top * n_fold)), # mean_rank_top
    prob_top = as.numeric(rep(NaN, n_top * n_fold)), # prob_in_top
    top_index = as.numeric(rep(NaN, n_top * n_fold)) # top_index
  )
  return(adaptmle)
}

example_class <- example_adaptmle_class(se = airway)
```

**bioadaptest**

Data Adaptive Multiple Testing for Computational Biology

**Description**

A thin wrapper that implements the main data-adaptive multiple hypothesis testing strategy for data structures commonly found in computational biology experiments, using the popular SummarizedExperiment container class.
bioadaptest

Usage

bioadaptest(data_in, var_int, cntrl_set = NULL, n_top = 25, n_fold = 10,
    parameter_wrapper = rank_DE, learning_library = c("SL.mean", "SL.glm"),
    absolute = FALSE, negative = FALSE, p_cutoff = 0.05, q_cutoff = 0.05)

Arguments

data_in An object of class SummarizedExperiment, a common container class for computational biology and bioinformatics. This object is used to construct the output object of class adaptmle.

var_int A numeric vector of binary treatment assignment whose effect on the biological units is to be assessed. The data-adaptive target parameter approach finds any biological sites strongly impacted by this quantity across the observed experimental units (subjects).

cntrl_set A matrix of discrete variables representing baseline covariates that are controlled for in the estimation of the data-adaptive target parameter via targeted maximum likelihood estimation. If NULL, an identity vector is generated internally.

n_top (integer vector) - value for the number of candidate covariates to generate using the data-adaptive estimation algorithm.

n_fold (integer vector) - number of cross-validation folds.

parameter_wrapper (function) - user-defined function that takes input (Y, A, W, absolute, negative) and outputs a (integer vector) containing ranks of biomarkers (outcome variables). For detail, please refer to the documentation for rank_DE.

learning_library (character vector) - library of learning algorithms to be used in fitting the "Q" and "g" step of the standard TMLE procedure.

absolute (logical) - whether or not to test for absolute effect size. If FALSE, test for directional effect. This overrides argument negative.

negative (logical) - whether or not to test for negative effect size. If FALSE = test for positive effect size. This is effective only when absolute = FALSE.

p_cutoff The minimum p-value required to evaluate a given biological unit (e.g., gene) as statistically significant.

q_cutoff The minimum p-value required to evaluate a given biological unit (e.g., gene) as statistically significant after applying a correction for multiple hypothesis testing.

Value

An object of class adaptmle, sub-classed from the popular container class SummarizedExperiment, containing information about the experiment being analyzed as well as results from applying the TMLE for the data-adaptive target parameter as produced by adaptest.

Examples

library(SummarizedExperiment)
library(airway)
set.seed(5678)
data(airway)
genes_sub <- order(sample(seq_len(100)))
air_reduced <- airway[genes_sub, ]
simple_air <- cbind(air_reduced, air_reduced)
dex_var = as.numeric(as.matrix(colData(simple_air))[, 3] - 1)
airway_out <- bioadaptest(data_in = simple_air,
                          var_int = dex_var,
cntrl_set = NULL,
n_top = 5,
n_fold = 2,
parameter_wrapper = rank_DE)

---

**cv_param_est**

Compute data-adaptive parameter estimate for a single cross-validation fold

**Description**

Compute data-adaptive parameter estimate for a single cross-validation fold

**Usage**

```r
cv_param_est(fold, data, parameter_wrapper, absolute, negative, n_top,
learning_library, Y_name, A_name, W_name)
```

**Arguments**

- `fold`: fold output from `origami`
- `data`: entire training data
- `parameter_wrapper`: user-defined function
- `absolute`: boolean: `TRUE` = test for absolute effect size. This `FALSE` = test for directional effect. This overrides argument `negative`.
- `negative`: boolean: `TRUE` = test for negative effect size, `FALSE` = test for positive effect size
- `n_top`: integer value for the number of candidate covariates to generate using the data-adaptive estimation algorithm
- `learning_library`: character of `SuperLearner` library
- `Y_name`: (character) colnames of all biomarkers
- `A_name`: (character) colnames of treatment
- `W_name`: (character) colnames of all baseline covariates

**Value**

- `data_adaptive_index` (integer vector) rank for each gene
- `index_grid` (integer matrix) gene index from rank 1 to rank K
- `psi_est` estimand of DE for rank 1 to rank K genes
- `EIC_est` estimand of EIC for rank 1 to rank K genes
data_adapt S3-Style Constructor for Data Adaptive Parameter Class

Description
S3-Style Constructor for Data Adaptive Parameter Class

Usage
data_adapt(Y, A, W = NULL, n_top, n_fold, absolute, negative, parameter_wrapper, learning_library)

Arguments
Y (numeric vector) - continuous or binary biomarkers outcome variables
A (numeric vector) - binary treatment indicator: 1 = treatment, 0 = control
W (numeric vector, numeric matrix, or numeric data.frame) - matrix of baseline covariates where each column correspond to one baseline covariate. Each row correspond to one observation
n_top (integer vector) - value for the number of candidate covariates to generate using the data-adaptive estimation algorithm.
n_fold (integer vector) - number of cross-validation folds.
absolute (logical) - whether or not to test for absolute effect size. If FALSE, test for directional effect. This overrides argument negative.
negative (logical) - whether or not to test for negative effect size. If FALSE = test for positive effect size. This is effective only when absolute = FALSE.
parameter_wrapper (function) - user-defined function that takes input (Y, A, W, absolute, negative) and outputs a (integer vector) containing ranks of biomarkers (outcome variables). For detail, please refer to the documentation for rank_DE.
learning_library (character vector) - library of learning algorithms to be used in fitting the "Q" and "g" step of the standard TMLE procedure.

Value
S3 object of class "data_adapt" for data-adaptive multiple testing.

g_get_composition Decomposition tables of the data-adaptive parameter after data-mining

Description
Customized informative tables for examining data-adaptive statistics.
**get_significant_biomarker**

*Extract statistically significant biomarkers*

**Description**

Extract statistically significant biomarkers

**Usage**

```
get_significant_biomarker(object, cutoff = 0.5)
```

**Arguments**

- **object** (data_adapt) - object of class data_adapt as returned by adaptest
- **cutoff** (numeric) - cut-off value for composition percentage

**Value**

(integer vector) of significant gene index

---

**get_composition**

get_composition(object, type = "small")

**Arguments**

- **object** (data_adapt) - object of class data_adapt as returned by adaptest
- **type** (character) - 'small' or 'big'. 'small' mode returns composition of data-adaptive parameters after multiple testing stage. 'big' mode returns composition of data-adaptive parameters before multiple testing stage.

**Value**

(numeric matrix) containing what fraction of the data-adaptive parameter comes from which biomarker in the original dataset.

**Examples**

```r
set.seed(1234)
data(simpleArray)
simulated_array <- simulated_array
simulated_treatment <- simulated_treatment

adaptest_out <- adaptest(Y = simulated_array,
                         A = simulated_treatment,
                         W = NULL,
                         n_top = 5,
                         n_fold = 3,
                         learning_library = 'SL.glm',
                         parameter_wrapper = adaptest::rank.DE,
                         absolute = FALSE,
                         negative = FALSE)

get_composition(adaptest_out, type = 'small')
```
Examples

```r
set.seed(1234)
data(simpleArray)
simulated_array <- simulated_array
simulated_treatment <- simulated_treatment

adaptest_out <- adaptest(Y = simulated_array,
                         A = simulated_treatment,
                         W = NULL,
                         n_top = 5,
                         n_fold = 3,
                         learning_library = 'SL.glm',
                         parameter_wrapper = adaptest::rank_DE,
                         absolute = FALSE,
                         negative = FALSE)

get_significant_biomarker(adaptest_out)
```

plot.data_adapt  
_Plot method for data_adapt objects_

Description

Customized plotting method for easily examining data-adaptive statistics

Usage

```r
## S3 method for class 'data_adapt'
plot(x, ..., plot_type = c("biomarker", "adapt_param"))
```

Arguments

- `x` (data_adapt) - object of class data_adapt as returned by adaptest
- `...` additional arguments passed to `plot` as necessary
- `plot_type` character vector specifying which of the two types of plots to generate: "biomarker" for a plot sorted average CV-rank, or "adapt_param" for a plot sorted by q-values with labels corresponding to indices

Value

plot of model statistics
print.data_adapt  

Print method for data_adapt objects

Description
Customized informative print method for examining data-adaptive statistics

Usage
## S3 method for class 'data_adapt'
print(x, ...)

Arguments

x (data_adapt) - object of class data_adapt as returned by adaptest

... additional arguments passed to print as necessary

Value
strings into stdout; containing information of the fitted model

rank_DE  

Compute ranking of biomarkers by sorting effect sizes

Description
Computes ranking of biomarkers based effect sizes, which are computed by Targeted Minimum Loss-Based Estimation. This function is designed to be called inside adaptest; it should not be run by itself outside of that context.

Usage
rank_DE(Y, A, W, absolute = FALSE, negative = FALSE,
learning_library = c("SL.glm", "SL.step", "SL.glm.interaction", "SL.gam"))

Arguments

Y (numeric vector) - continuous or binary biomarkers outcome variables
A (numeric vector) - binary treatment indicator: 1 = treatment, 0 = control
W (numeric vector, numeric matrix, or numeric data.frame) - matrix of baseline covariates where each column correspond to one baseline covariate. Each row correspond to one observation

absolute (logical) - whether or not to test for absolute effect size. If FALSE, test for directional effect. This overrides argument negative.
negative (logical) - whether or not to test for negative effect size. If FALSE = test for positive effect size. This is effective only when absolute = FALSE.

learning_library (character vector) - library of learning algorithms to be used in fitting the "Q" and "g" step of the standard TMLE procedure.
Value

an integer vector containing ranks of biomarkers.

Examples

```r
set.seed(1234)
data(simpleArray)
simulated_array <- simulated_array
simulated_treatment <- simulated_treatment
rank_DE(Y = simulated_array,
       A = simulated_treatment,
       W = rep(1, length(simulated_treatment)),
       absolute = FALSE,
       negative = FALSE)
```

---

**rank_ttest**  
*Compute ranking of biomarkers by sorting t-test p-values*

Description

Compute ranking of biomarkers by sorting t-test p-values

Usage

```r
rank_ttest(Y, A, W)
```

Arguments

- **Y**  
  (numeric vector) - continuous or binary biomarkers outcome variables
- **A**  
  (numeric vector) - binary treatment indicator: 1 = treatment, 0 = control
- **W**  
  (numeric vector, numeric matrix, or numeric data.frame) - matrix of baseline covariates where each column correspond to one baseline covariate and each row correspond to one observation.

Value

an integer vector containing ranks of biomarkers.

Examples

```r
set.seed(1234)
data(simpleArray)
r.rank_ttest(Y = simulated_array,
             A = simulated_treatment,
             W = rep(1, length(A)))
```
**simulated_array**  
*Simulated differential expression data with one exposure*

**Description**
A dataset containing 1e4 biomarkers and one exposure

**Usage**
simulated_array

**Format**
A numeric matrix containing 1e4 biomarkers of 1e2 subjects.  
This is example data to be used in testing the adaptest procedure. Consult the vignettes for how to use this data.

**Value**
A matrix simulated_array

---

**simulated_treatment**  
*Simulated differential expression data with one exposure*

**Description**
A dataset containing 1e4 biomarkers and one exposure

**Usage**
simulated_treatment

**Format**
A numeric vector containing binary exposures  
This is example data to be used in testing the adaptest procedure. Consult the vignettes for how to use this data.

**Value**
A numeric vector simulated_treatment.
Summary tables for data_adapt objects

Description

Summary tables for data_adapt objects

Usage

```r
# S3 method for class 'data_adapt'
summary(object, type = "adapt_param", ...)
```

Arguments

- `object` (data_adapt) object as returned by adaptest
- `type` (character) - ‘adapt_param’ or ‘biomarker’. ‘adapt_param’ mode summarizes the data-adaptive target parameter. ‘biomarker’ mode summarizes characteristics of the biomarkers from the original data
- `...` not implemented

Value

(data.frame) of the summary statistics

- `type = 'adapt_param'` with columns: 'data-adaptive parameters', 'Differential expression', 'p-values', 'q-values'
- `type = 'biomarker'` with columns: 'biomarkers', 'mean rank', 'appear in top'
Index

*Topic datasets
  simulated_array, 12
  simulated_treatment, 12
  .adaptmle(adapTMLE-class), 4

adaptest, 2
adapTMLE-class, 4

bioadaptest, 4

cv_param_est, 6

data_adapt, 7

get_composition, 7
get_significant_biomarker, 8

plot.data_adapt, 9
print.data_adapt, 10

rank_DE, 10
rank_ttest, 11

simulated_array, 12
simulated_treatment, 12
summary.data_adapt, 13