

Package ‘biotmle’

July 23, 2025

Title Targeted Learning with Moderated Statistics for Biomarker
Discovery

Version 1.32.1

Description Tools for differential expression biomarker discovery based on microarray and next-generation sequencing data that leverage efficient semiparametric estimators of the average treatment effect for variable importance analysis. Estimation and inference of the (marginal) average treatment effects of potential biomarkers are computed by targeted minimum loss-based estimation, with joint, stable inference constructed across all biomarkers using a generalization of moderated statistics for use with the estimated efficient influence function. The procedure accommodates the use of ensemble machine learning for the estimation of nuisance functions.

Depends R (>= 4.0)

License MIT + file LICENSE

URL <https://code.nimahejazi.org/biotmle>

BugReports <https://github.com/nhejazi/biotmle/issues>

Encoding UTF-8

LazyData false

Imports stats, methods, dplyr, tibble, ggplot2, ggsci, superheat,
assertthat, drtmle (>= 1.0.4), S4Vectors, BiocGenerics,
BiocParallel, SummarizedExperiment, limma

Suggests testthat, knitr, rmarkdown, BiocStyle, arm, earth, ranger,
SuperLearner, Matrix, DBI, biotmleData (>= 1.1.1)

VignetteBuilder knitr

RoxygenNote 7.1.2

biocViews Regression, GeneExpression, DifferentialExpression,
Sequencing, Microarray, RNASeq, ImmunoOncology

git_url <https://git.bioconductor.org/packages/biotmle>

git_branch RELEASE_3_21

git_last_commit 377fd0d

git_last_commit_date 2025-05-08
Repository Bioconductor 3.21
Date/Publication 2025-07-23
Author Nima Hejazi [aut, cre, cph] (ORCID:
 <<https://orcid.org/0000-0002-7127-2789>>),
Alan Hubbard [aut, ths] (ORCID:
 <<https://orcid.org/0000-0002-3769-0127>>),
Mark van der Laan [aut, ths] (ORCID:
 <<https://orcid.org/0000-0003-1432-5511>>),
Weixin Cai [ctb] (ORCID: <<https://orcid.org/0000-0003-2680-3066>>),
Philippe Boileau [ctb] (ORCID: <<https://orcid.org/0000-0002-4850-2507>>)
Maintainer Nima Hejazi <nh@nimahejazi.org>

Contents

biomarkertmle	2
bioTMLE-class	4
data.frame_OR_EList-class	5
eif	5
exp_biomarkertmle	6
heatmap_ic	6
modtest_ic	8
plot.bioTMLE	9
rnaseq_ic	10
toptable	11
volcano_ic	11

Index	13
--------------	-----------

biomarkertmle	<i>Biomarker Evaluation with Targeted Minimum Loss Estimation of the ATE</i>
---------------	--

Description

Computes the causal target parameter defined as the difference between the biomarker expression values under treatment and those same values under no treatment, using Targeted Minimum Loss Estimation.

Usage

```
biomarkertmle(  
  se,  
  varInt,  
  normalized = TRUE,  
  ngscounts = FALSE,
```

```

bppar_type = BiocParallel::MulticoreParam(),
bppar_debug = FALSE,
cv_folds = 1,
g_lib = c("SL.mean", "SL.glm", "SL.bayesglm"),
Q_lib = c("SL.mean", "SL.bayesglm", "SL.earth", "SL.ranger"),
...
)

```

Arguments

se	A SummarizedExperiment containing microarray expression or next-generation sequencing data in the assays slot and a matrix of phenotype-level data in the colData slot.
varInt	A numeric indicating the column of the design matrix corresponding to the treatment or outcome of interest (in the colData slot of the SummarizedExperiment argument "se").
normalized	A logical indicating whether the data included in the assay slot of the input SummarizedExperiment object has been normalized externally. The default is set to TRUE with the expectation that an appropriate normalization method has been applied. If set to FALSE, median normalization is performed for microarray data.
ngscounts	A logical indicating whether the data are counts generated from a next-generation sequencing experiment (e.g., RNA-seq). The default setting assumes continuous expression measures as generated by microarray platforms.
bppar_type	A parallelization option specified by BiocParallel. Consult the manual page for BiocParallelParam for possible types and their descriptions. The default for this argument is MulticoreParam , for multicore evaluation.
bppar_debug	A logical indicating whether or not to rely upon pkgBiocParallel. Setting this argument to TRUE, replaces the call to bplapply by a call to lapply , which significantly reduces the overhead of debugging. Note that invoking this option overrides all other parallelization arguments.
cv_folds	A numeric scalar indicating how many folds to use in performing targeted minimum loss estimation. Cross-validated estimates have been demonstrated to allow relaxation of certain theoretical conditions and accommodate the construction of more conservative variance estimates.
g_lib	A character vector specifying the library of machine learning algorithms for use in fitting the propensity score $P(A = a W)$.
Q_lib	A character vector specifying the library of machine learning algorithms for use in fitting the outcome regression $E[Y A, W]$.
...	Additional arguments to be passed to drtml in computing the targeted minimum loss estimator of the average treatment effect.

Value

S4 object of class `biotml`, inheriting from `SummarizedExperiment`, with additional slots `tmlOut` and `call`, among others, containing TML estimates of the ATE of exposure on biomarker expression.

Examples

```

library(dplyr)
library(biotmleData)
library(SuperLearner)
library(SummarizedExperiment)
data(illuminaData)

colData(illuminaData) <- colData(illuminaData) %>%
  data.frame() %>%
  mutate(age = as.numeric(age > median(age))) %>%
  DataFrame()
benz_idx <- which(names(colData(illuminaData)) %in% "benzene")

biomarkerTMLEout <- biomarkertmle(
  se = illuminaData[1:2, ],
  varInt = benz_idx,
  bppar_type = BiocParallel::SerialParam(),
  g_lib = c("SL.mean", "SL.glm"),
  Q_lib = c("SL.mean", "SL.glm")
)

```

bioTMLE-class

Constructor for class bioTMLE

Description

Constructor for class bioTMLE

Value

class biotmle object, sub-classed from SummarizedExperiment.

Examples

```

library(SummarizedExperiment)
library(biotmleData)
data(illuminaData)

example_biotmle_class <- function(se) {
  call <- match.call(expand.dots = TRUE)
  biotmle <- .biotmle(
    SummarizedExperiment(
      assays = assay(se),
      rowData = rowData(se),
      colData = colData(se)
    ),
    call = call,
    ateOut = as.numeric(rep(NA, 10)),
    tmleOut = as.data.frame(matrix(NA, 10, 10)),
  )
}

```

```

      topTable = as.data.frame(matrix(NA, 10, 10))
    )
    return(biotmle)
  }

example_class <- example_biotmle_class(se = illuminaData)

```

data.frame_OR_EList-class

S4 class union data.frame_OR_EList

Description

Virtual class union containing members of `data.frame` and `limma::EList`, used internally to handle situations when a returned object has a type that cannot be guessed from the function call.

Value

fusion of classes `data.frame` and `EList`, used within `.biotmle` by class `bioTMLE` to handle uncertainty in the object passed to slot "tmleOut".

eif

Accessor for Table of Raw Efficient Influence Function Values

Description

Accessor for Table of Raw Efficient Influence Function Values

Usage

```
eif(object)
```

Arguments

`object` S4 object of class `bioTMLE`.

Value

contents of `tmleOut` slot of object of class `biotmle`.

exp_biomarkertmle	<i>TMLE procedure using ATE for Biomarker Identification from Exposure</i>
-------------------	--

Description

This function performs influence curve-based estimation of the effect of an exposure on biological expression values associated with a given biomarker, controlling for a user-specified set of baseline covariates.

Usage

```
exp_biomarkertmle(Y, A, W, g_lib, Q_lib, cv_folds, ...)
```

Arguments

Y	A numeric vector of expression values for a given biomarker.
A	A numeric vector of discretized exposure vector (e.g., from a design matrix whose effect on expression values is of interest.
W	A Matrix of numeric values corresponding to baseline covariates to be marginalized over in the estimation process.
g_lib	A character vector identifying the library of learning algorithms to be used in fitting the propensity score $P[A = a W]$.
Q_lib	A character vector identifying the library of learning algorithms to be used in fitting the outcome regression $E[Y A, W]$.
cv_folds	A numeric scalar indicating how many folds to use in performing targeted minimum loss estimation. Cross-validated estimates are more robust, allowing relaxing of theoretical conditions and construction of conservative variance estimates.
...	Additional arguments passed to drtmle in computing the targeted minimum loss estimator of the average treatment effect.

Value

TMLE-based estimate of the relationship between biomarker expression and changes in an exposure variable, computed iteratively and saved in the `tmleOut` slot in a `biotmle` object.

heatmap_ic	<i>Heatmap for class biotmle</i>
------------	----------------------------------

Description

Heatmap of contributions of a select subset of biomarkers to the variable importance measure changes as assessed by influence curve-based estimation, across all subjects. The heatmap produced performs supervised clustering, as per Pollard & van der Laan (2008) <doi:10.2202/1544-6115.1404>.

Usage

```
heatmap_ic(x, ..., design, FDRcutoff = 0.25, type = c("top", "all"), top = 25)
```

Arguments

<code>x</code>	Object of class <code>biotmle</code> as produced by an appropriate call to <code>biomarkertmle</code> .
<code>...</code>	additional arguments passed to <code>superheat::superheat</code> as necessary
<code>design</code>	A vector giving the contrast to be displayed in the heatmap.
<code>FDRcutoff</code>	Cutoff to be used in controlling the False Discovery Rate.
<code>type</code>	A character describing whether to plot only a top number (as defined by FDR-corrected p-value) of biomarkers or all biomarkers.
<code>top</code>	Number of identified biomarkers to plot in the heatmap.

Value

heatmap (from **superheat**) using hierarchical clustering to plot the changes in the variable importance measure for all subjects across a specified top number of biomarkers.

Examples

```
## Not run:
library(dplyr)
library(biotmleData)
library(SummarizedExperiment)
data(illuminaData)

colData(illuminaData) <- colData(illuminaData) %>%
  data.frame() %>%
  mutate(age = as.numeric(age > median(age))) %>%
  DataFrame()
benz_idx <- which(names(colData(illuminaData)) %in% "benzene")

biomarkerTMLEout <- biomarkertmle(
  se = illuminaData,
  varInt = benz_idx,
  bpar_type = BiocParallel::SerialParam(),
  g_lib = c("SL.mean", "SL.glm"),
  Q_lib = c("SL.mean", "SL.glm")
)

limmaTMLEout <- modtest_ic(biotmle = biomarkerTMLEout)

heatmap_ic(x = limmaTMLEout, design = design, FDRcutoff = 0.05, top = 10)

## End(Not run)
```

modtest_ic

*Moderated Statistical Tests for Influence Functions***Description**

Performs variance shrinkage via application of an empirical Bayes procedure (of LIMMA) on the observed data after a transformation moving the data to influence function space, based on the average treatment effect parameter.

Usage

```
modtest_ic(biotmle, adjust = "BH", pval_type = c("normal", "logistic"), ...)
```

Arguments

biotmle	biotmle object as generated by biomarkertmle
adjust	the multiple testing correction to be applied to p-values that are generated from the moderated tests. The recommended (default) method is that of Benjamini and Hochberg. See topTable for a list of appropriate methods.
pval_type	The reference distribution to be used for computing the p-value. Those based on the normal approximation tend to provide misleading inference when working with moderately sized (finite) samples. Use of the logistic distribution has been found to empirically improve performance in settings where multiple hypothesis testing is a concern.
...	Other arguments passed to topTable .

Value

biotmle object containing the results of applying both [lmFit](#) and [topTable](#).

Examples

```
library(dplyr)
library(biotmleData)
library(SuperLearner)
library(SummarizedExperiment)
data(illuminaData)

colData(illuminaData) <- colData(illuminaData) %>%
  data.frame() %>%
  dplyr::mutate(age = as.numeric(age > median(age))) %>%
  Dataframe()
benz_idx <- which(names(colData(illuminaData)) %in% "benzene")

biomarkerTMLEout <- biomarkertmle(
  se = illuminaData[1:2, ],
  varInt = benz_idx,
  bppar_type = BiocParallel::SerialParam(),
```



```

    g_lib = c("SL.mean", "SL.glm"),
    Q_lib = c("SL.mean", "SL.glm")
  )

  limmaTMLEout <- modtest_ic(biotmle = biomarkerTMLEout)

```

plot.bioTMLE

Plot p-values from moderated statistical tests for class biotmle

Description

Histogram of raw or FDR-adjusted p-values from the moderated t-test.

Usage

```

## S3 method for class 'bioTMLE'
plot(x, ..., type = "pvals_adj")

```

Arguments

x	object of class biotmle as produced by an appropriate call to biomarkertmle
...	additional arguments passed plot as necessary
type	character describing whether to provide a plot of unadjusted or adjusted p-values (adjustment performed via Benjamini-Hochberg)

Value

object of class ggplot containing a histogram of the raw or Benjamini-Hochberg corrected p-values (depending on user input).

Examples

```

## Not run:
library(dplyr)
library(biotmleData)
library(SuperLearner)
library(SummarizedExperiment)
data(illuminaData)

colData(illuminaData) <- colData(illuminaData) %>%
  data.frame() %>%
  mutate(age = as.numeric(age > median(age))) %>%
  DataFrame()
benz_idx <- which(names(colData(illuminaData)) %in% "benzene")

biomarkerTMLEout <- biomarkertmle(
  se = illuminaData,
  varInt = benz_idx,
  bppar_type = BiocParallel::SerialParam(),

```

```

    g_lib = c("SL.mean", "SL.glm"),
    Q_lib = c("SL.mean", "SL.glm")
  )

  limmaTMLEout <- modtest_ic(biotmle = biomarkerTMLEout)

  plot(x = limmaTMLEout, type = "pvals_adj")

  ## End(Not run)

```

rnaseq_ic	<i>Utility for using voom transformation with TMLE for biomarker discovery</i>
-----------	--

Description

This function prepares next-generation sequencing data (counts) for use with the biomarker TMLE procedure by invoking the voom transform of limma.

Usage

```
rnaseq_ic(biotmle, weights = TRUE, ...)
```

Arguments

biotmle	A bioTMLE object containing next-generation sequencing count data in its "assays" slot.
weights	A logical indicating whether to return quality weights of samples in the output object.
...	Other arguments to be passed to voom or voomWithQualityWeights as appropriate.

Value

EList object containing voom-transformed expression measures of count data (actually, the mean-variance trend) in the "E" slot, to be passed into the biomarker TMLE procedure.

toptable	<i>Accessor for Results of Moderated Influence Function Hypothesis Testing</i>
----------	--

Description

Accessor for Results of Moderated Influence Function Hypothesis Testing

Usage

```
toptable(object)
```

Arguments

object S4 object of class bioTMLE.

Value

contents of topTable slot of object of class biotmle.

volcano_ic	<i>Volcano plot for class biotmle</i>
------------	---------------------------------------

Description

Volcano plot of the log-changes in the target causal paramter against the log raw p-values from the moderated t-test.

Usage

```
volcano_ic(biotmle, ate_bound = 1, pval_bound = 0.2)
```

Arguments

biotmle object of class biotmle as produced by an appropriate call to biomarkertmle

ate_bound A numeric indicating the highest magnitude of the average treatment effect to be colored on the x-axis of the volcano plot; this limits the observations to be considered differentially expressed to those in a user-specified interval.

pval_bound A numeric indicating the largest corrected p-value to be colored on the y-axis of the volcano plot; this limits observations considered as differentially expressed to those in a user-specified interval.

Value

object of class ggplot containing a standard volcano plot of the log-fold change in the causal target parameter against the raw log p-value computed from the moderated tests in modtest_ic.

Examples

```
## Not run:
library(dplyr)
library(biotmleData)
library(SuperLearner)
library(SummarizedExperiment)
data(illuminaData)

colData(illuminaData) <- colData(illuminaData) %>%
  data.frame() %>%
  mutate(age = as.numeric(age > median(age))) %>%
  DataFrame()
benz_idx <- which(names(colData(illuminaData)) %in% "benzene")

biomarkerTMLEout <- biomarkertmle(
  se = illuminaData,
  varInt = benz_idx,
  bppar_type = BiocParallel::SerialParam(),
  g_lib = c("SL.mean", "SL.glm"),
  Q_lib = c("SL.mean", "SL.glm")
)

limmaTMLEout <- modtest_ic(biotmle = biomarkerTMLEout)

volcano_ic(biotmle = limmaTMLEout)

## End(Not run)
```

Index

`.biotmle` (`bioTMLE`-class), [4](#)

`BiocParallelParam`, [3](#)
`biomarkertmle`, [2](#)
`bioTMLE`-class, [4](#)
`bplapply`, [3](#)

`data.frame_OR_EList`-class, [5](#)
`drtmle`, [3](#), [6](#)

`EIF`, [5](#)
`exp_biomarkertmle`, [6](#)

`heatmap_IC`, [6](#)

`lmFit`, [8](#)

`modtest_IC`, [8](#)
`MulticoreParam`, [3](#)

`plot.bioTMLE`, [9](#)

`rnaseq_IC`, [10](#)

`topTable`, [8](#)
`toptable`, [11](#)

`volcano_IC`, [11](#)
`voom`, [10](#)
`voomWithQualityWeights`, [10](#)