Package ‘biobroom’
March 13, 2024

Title  Turn Bioconductor objects into tidy data frames
Version  1.34.0
Author  Andrew J. Bass, David G. Robinson, Steve Lianoglou, Emily Nelson, John D. Storey, with contributions from Laurent Gatto
Maintainer  John D. Storey <jstorey@princeton.edu> and Andrew J. Bass <ajbass@emory.edu>
Description  This package contains methods for converting standard objects constructed by bioinformatics packages, especially those in Bioconductor, and converting them to tidy data. It thus serves as a complement to the broom package, and follows the same the tidy, augment, glance division of tidying methods. Tidying data makes it easy to recombine, reshape and visualize bioinformatics analyses.
biocViews  MultipleComparison, DifferentialExpression, Regression, GeneExpression, Proteomics, DataImport
Depends  R (>= 3.0.0), broom
License  LGPL
LazyData  true
Imports  dplyr, tidyr, Biobase
Suggests  limma, DESeq2, airway, ggplot2, plyr, GenomicRanges, testthat, magrittr, edgeR, qvalue, knitr, data.table, MSnbase, markdown, SummarizedExperiment
VignetteBuilder  knitr
URL  https://github.com/StoreyLab/biobroom
BugReports  https://github.com/StoreyLab/biobroom/issues
RoxygenNote  5.0.1
git_url  https://git.bioconductor.org/packages/biobroom
git_branch  RELEASE_3_18
git_last_commit  cd67b42
git_last_commit_date  2023-10-24
Repository  Bioconductor 3.18
Date/Publication  2024-03-13
R topics documented:

- augment_sva ........................................... 2
- biobroom .................................................. 3
- DESeq2_tidiers ........................................... 3
- edgeR_tidiers ............................................ 5
- ExpressionSet_tidiers ................................... 6
- hammer .................................................... 7
- limma_tidiers ............................................. 8
- list_tidiers ............................................... 10
- MSnSet_tidiers ........................................... 11
- qvalue_tidiers ........................................... 12
- SummarizedExperiment_tidiers ..................... 14
- tidy.deSet ............................................... 15
- tidy.GRanges ............................................ 16

Index ..................................................... 18

augment_sva  

Tidying methods for a sva list

Description

These are methods for turning a sva list, from the sva package, into a tidy data frame. tidy returns a data.frame of the estimated surrogate variables, glance returns a data.frame of the posterior probabilities, and glance returns a data.frame with only the number of surrogate variables.

Usage

augment_sva(x, data, ...)

tidy_sva(x, addVar = NULL, ...)

glance_sva(x, ...)

Arguments

- x  sva list
- data Original data
- ... extra arguments (not used)
- addVar add additional coefficients to the estimated surrogate variables
Value

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

`augment` returns one row per gene. It always contains the columns

- `pprob.gam`: Posterior probability each gene is affected by heterogeneity
- `pprob.b`: Posterior probability each gene is affected by model

`tidy` returns the estimate surrogate variables.

`glance` returns the estimate surrogate variables.

biobroom

Convert Bioconductor Object into Tidy Data Frames

Description

This package contains methods for converting standard objects constructed by bioinformatics packages, especially those in Bioconductor, and converting them to tidy data. It thus serves as a complement to the broom package, and follows the same the tidy, augment, glance division of tidying methods. Tidying data makes it easy to recombine, reshape and visualize bioinformatics analyses.

DESeq2_tidders

Tidying methods for DESeq2 DESeqDataSet objects

Description

This reshapes a DESeq2 expressionset object into a tidy format. If the dataset contains hypothesis test results (p-values and estimates), this summarizes one row per gene per possible contrast.

Usage

```r
## S3 method for class 'DESeqDataSet'
tidy(x, colData = FALSE, intercept = FALSE, ...)

## S3 method for class 'DESeqResults'
tidy(x, ...)
```

Arguments

- `x`: DESeqDataSet object
- `colData`: whether `colData` should be included in the tidied output for those in the DESeqDataSet object. If dataset includes hypothesis test results, this is ignored
- `intercept`: whether to include hypothesis test results from the (Intercept) term. If dataset does not include hypothesis testing, this is ignored
- `...`: extra arguments (not used)
Details

colDat=TRUE adds covariates from colData to the data frame.

Value

If the dataset contains results (p-values and log2 fold changes), the result is a data frame with the columns:

<table>
<thead>
<tr>
<th>term</th>
<th>The contrast being tested, as given to results</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene</td>
<td>gene ID</td>
</tr>
<tr>
<td>baseMean</td>
<td>mean abundance level</td>
</tr>
<tr>
<td>estimate</td>
<td>estimated log2 fold change</td>
</tr>
<tr>
<td>stderr</td>
<td>standard error in log2 fold change estimate</td>
</tr>
<tr>
<td>statistic</td>
<td>test statistic</td>
</tr>
<tr>
<td>p.value</td>
<td>p-value</td>
</tr>
<tr>
<td>p.adjusted</td>
<td>adjusted p-value</td>
</tr>
</tbody>
</table>

If the dataset does not contain results (DESeq has not been run on it), tidy defaults to tidying the counts in the dataset:

<table>
<thead>
<tr>
<th>gene</th>
<th>gene ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample</td>
<td>sample ID</td>
</tr>
<tr>
<td>count</td>
<td>number of reads in this gene in this sample</td>
</tr>
</tbody>
</table>

If colData = TRUE, it also merges this with the columns present in colData(x).

Examples

# From DESeq2 documentation

```r
if (require("DESeq2")) {
  dds <- makeExampleDESeqDataSet(betaSD = 1)

  tidy(dds)
  # With design included
  tidy(dds, colData=TRUE)

  # add a noise confounding effect
  colData(dds)$noise <- rnorm(nrow(colData(dds)))
  design(dds) <- (~ condition + noise)

  # perform differential expression tests
  ddsres <- DESeq(dds, test = "Wald")
  # now results are per-gene, per-term
  tidied <- tidy(ddres)
  tidied

  if (require("ggplot2")) {
    ggplot(tidied, aes(p.value)) + geom_histogram(binwidth = .05) +
    }
edgeR_tidiers

Tidiers for edgeR’s differential expression objects

Description

Tidy, augment and glance methods for turning edgeR objects into tidy data frames, where each row represents one observation and each column represents one column.

Usage

## S3 method for class 'DGEExact'
tidy(x, ...)

## S3 method for class 'DGEList'
tidy(x, addSamples = FALSE, ...)

## S3 method for class 'DGEList'
augment(x, data = NULL, ...)

## S3 method for class 'DGEExact'
glance(x, alpha = 0.05, p.adjust.method = "fdr", ...)

Arguments

x

DGEExact, DGEList object

... extra arguments (not used)

addSamples

Merge information from samples. Default is FALSE.

data

merge data to augment. This is particularly useful when merging gene names or other per-gene information. Default is NULL.

alpha

Confidence level to test for significance

p.adjust.method

Method for adjusting p-values to determine significance; can be any in p.adjust.methods

Value

tidy defaults to tidying the counts in the dataset:

gene gene ID

sample sample ID

count number of reads in this gene in this sample
If `addSamples = TRUE`, it also merges this with the sample information present in `x$samples`.

`augment` returns per-gene information (DGEList only)

`glance` returns one row with the columns (DGEExact only)

- **significant**: number of significant genes using desired adjustment method and confidence level
- **comparison**: The pair of groups compared by edgeR, delimited by `/`

### Examples

```r
if (require("edgeR")) {
  library(Biobase)
  data(hammer)
  hammer.counts <- exprs(hammer)[, 1:4]
  hammer.treatment <- phenoData(hammer)$protocol[1:4]

  y <- DGEList(counts=hammer.counts,group=hammer.treatment)
  y <- calcNormFactors(y)
  y <- estimateCommonDisp(y)
  y <- estimateTagwiseDisp(y)
  et <- exactTest(y)

  head(tidy(et))
  head(glance(et))
}
```

---

**ExpressionSet_tidiers**  
*Tidying methods for Biobase’s ExpressionSet objects*

---

**Description**

Tidying methods for Biobase’s ExpressionSet objects

**Usage**

```r
## S3 method for class 'ExpressionSet'
tidy(x, addPheno = FALSE,
     assay = Biobase::assayDataElementNames(x)[1L], ...)
```

**Arguments**

- **x**: ExpressionSet object
- **addPheno**: whether columns should be included in the tidied output for those in the ExpressionSet’s phenoData
- **assay**: The name of the `assayDataElement` to use as the values to tidy. Defaults to `assayDataElementNames(x)[1L]`, which is usually equivalent to `exprs(x)`.
- **...**: extra arguments (not used)
Details

addPheno=TRUE adds columns that are redundant (since they add per-sample information to a per-sample-per-gene data frame), but that are useful for some kinds of graphs and analyses.

Value

tidy returns a data frame with one row per gene-sample combination, with columns

gene  gene name
sample sample name (from column names)
value  expressions on log2 scale

Examples

library(Biobase)
# import ExpressionSet object
data(hammer)

# Use tidy to extract genes, sample ids and measured value
tidy(hammer)
# add phenoType data
tidy(hammer, addPheno=TRUE)

Description

An ExpressionSet containing the results of the Hammer et al 2010 RNA-Seq study on the nervous system of rats (Hammer et al 2010).

This was downloaded from the ReCount database of analysis-ready RNA-Seq datasets (Frazee et al 2011).


Usage

hammer
**Format**
An object of class `ExpressionSet` with 29516 rows and 8 columns.

**Value**
ExpressionSet

---

**Description**
Tidy, augment, and glance methods for `MArrayLM` objects, which contain the results of gene-wise linear models to microarray datasets. This class is the output of the `lmFit` and `eBayes` functions.

**Usage**
```r
## S3 method for class 'MArrayLM'
tidy(x, intercept = FALSE, ...)
## S3 method for class 'MArrayLM'
augment(x, data, ...)
## S3 method for class 'MArrayLM'
glance(x, ...)
## S3 method for class 'MAList'
tidy(x, ...)
## S3 method for class 'EList'
tidy(x, addTargets = FALSE, ...)
```

**Arguments**
- **x**: `MArrayLM, MAList, Elist` object
- **intercept**: whether the (Intercept) term should be included (default FALSE)
- **...**: extra arguments, not used
- **data**: original expression matrix; if missing, `augment` returns only the computed per-gene statistics
- **addTargets**: Add sample level information. Default is FALSE.

**Details**
Tidying this fit computes one row per coefficient per gene, while augmenting returns one row per gene, with per-gene statistics included. (This is thus a rare case where the `augment` output has more rows than the `tidy` output. This is a side effect of the fact that the input to `limma` is not tidy but rather a one-row-per-gene matrix).
Value

The output of tidying functions is always a data frame without rownames.
tidy returns one row per gene per coefficient. It always contains the columns
gene The name of the gene (extracted from the rownames of the input matrix)
term The coefficient being estimated
estimate The estimate of each per-gene coefficient

Depending on whether the object comes from eBayes, it may also contain
statistic Empirical Bayes t-statistic
p.value p-value computed from t-statistic
lod log-of-odds score

augment returns one row per gene, containing the original gene expression matrix if provided. It then adds columns containing the per-gene statistics included in the MArrayLM object, each prepended with a .:
.gene gene ID, obtained from the rownames of the input
.sigma per-gene residual standard deviation
.df.residual per-gene residual degrees of freedom

The following columns may also be included, depending on which have been added by lmFit and eBayes:
.AMean average intensity across probes
.statistic moderated F-statistic
.p.value p-value generated from moderated F-statistic
.df.total total degrees of freedom per gene
.df.residual residual degrees of freedom per gene
.s2.prior prior estimate of residual variance
.s2.post posterior estimate of residual variance

glance returns one row, containing
rank rank of design matrix
df.prior empirical Bayesian prior degrees of freedom
s2.prior empirical Bayesian prior residual standard deviation

tidy returns a data frame with one row per gene-sample combination, with columns
gene gene name
sample sample name (from column names)
value expressions on log2 scale

tidy returns a data frame with one row per gene-sample combination, with columns
Examples

```r
if (require("limma")) {
  # create random data and design
  set.seed(2014)
  dat <- matrix(rnorm(1000), ncol=4)
  dat[, 1:2] <- dat[, 1:2] + .5  # add an effect
  rownames(dat) <- paste0("g", 1:nrow(dat))
  des <- data.frame(treatment = c("a", "a", "b", "b"),
                    confounding = rnorm(4))

  lfit <- lmFit(dat, model.matrix(~ treatment + confounding, des))
  eb <- eBayes(lfit)
  head(tidy(lfit))
  head(tidy(eb))

  if (require("ggplot2")) {
    # the tidied form puts it in an ideal form for plotting
    ggplot(tidy(lfit), aes(estimate)) + geom_histogram(binwidth=1) +
           facet_wrap(~ term)
    ggplot(tidy(eb), aes(p.value)) + geom_histogram(binwidth=.2) +
           facet_wrap(~ term)
  }
}
```

Description

This method handles the return values of functions that return lists rather than S3 objects, such as `sva`, and therefore cannot be handled by S3 dispatch.

Usage

```r
## S3 method for class 'list'
tidy(x, ...)

## S3 method for class 'list'
glance(x, ...)
```
Arguments

x list object

Details

Those tiders themselves are implemented as functions of the form tidy_<function> that are not exported.

Value

tidy returns a data frame with one row per gene-sample combination, with columns

protein protein name
sample sample name (from column names)
value protein quantitation data
Examples

```r
if (require("MSnbase")) {
  library(MSnbase)
  # import MSnSet object
  data(msnset)

  # Use tidy to extract genes, sample ids and measured value
  tidy(msnset)
  # add phenoType data
  tidy(msnset, addPheno=TRUE)
}
```

**qvalue_tidders**

### Tidying methods for a qvalue object

**Description**

These are methods for turning a qvalue object, from the qvalue package for false discovery rate control, into a tidy data frame. `augment` returns a data.frame of the original p-values combined with the computed q-values and local false discovery rates, `tidy` constructs a table showing how the estimate of pi0 (the proportion of true nulls) depends on the choice of the tuning parameter lambda, and `glance` returns a data.frame with only the chosen pi0 value.

**Usage**

```r
## S3 method for class 'qvalue'
tidy(x, ...)
```

```r
## S3 method for class 'qvalue'
augment(x, data, ...)
```

```r
## S3 method for class 'qvalue'
glance(x, ...)
```

**Arguments**

- `x` qvalue object
- `...` extra arguments (not used)
- `data` Original data

**Value**

All tidying methods return a data.frame without rownames. The structure depends on the method chosen.

`tidy` returns one row for each choice of the tuning parameter lambda that was considered (argument lambda to `qvalue`), containing
lambda the tuning parameter
pi0 corresponding estimate of pi0
smoothed whether the estimate has been spline-smoothed

If pi0.method="smooth", the pi0 estimates and smoothed values both appear in the table. If pi0.method="bootstrap", smoothed is FALSE for all entries.

augment returns a data.frame with

p.value the original p-values given to qvalue
q.value the computed q-values
lfdr the local false discovery rate

glance returns a one-row data.frame containing

pi0 the estimated pi0 (proportion of nulls)
lambda lambda used to compute pi0. Note that if pi0 is 1, this may be NA since it can be ambiguous which lambda was used

Examples

library(ggplot2)
if (require("qvalue") ) {
  set.seed(2014)
  
  # generate p-values from many one sample t-tests: half of them null
  oracle <- rep(c(0, .5), each=1000)
  pvals <- sapply(oracle, function(mu) t.test(rnorm(15, mu))$p.value)
  qplot(pvals)
  q <- qvalue(pvals)
  tidy(q)
  head(augment(q))
  glance(q)

  # use augmented data to compare p-values to q-values
  ggplot(augment(q), aes(p.value, q.value)) + geom_point()

  # use tidy see how pi0 estimate changes with lambda, comparing
  # to smoothed version
  g <- ggplot(tidy(q), aes(lambda, pi0, color=smoothed)) + geom_line()
  g

  # show the chosen value
  g + geom_hline(yintercept=q$pi0, lty=2)
}
SummarizedExperiment_tidiers

Tidying methods for Biobase’s SummarizedExperiment objects

Description

Tidying methods for Biobase’s SummarizedExperiment objects

Usage

```r
## S3 method for class 'RangedSummarizedExperiment'
tidy(x, addPheno = FALSE,
    assay = SummarizedExperiment::assayNames(x)[1L], ...)
```

Arguments

- `x` SummarizedExperiment object
- `addPheno` whether columns should be included in the tidied output for those in the SummarizedExperiment colData
- `assay` Which assay to return as the value column. Defaults to assays(x)[[1L]]
- `...` extra arguments (not used)

Details

`addPheno=TRUE` adds columns that are redundant (since they add per-sample information to a per-sample-per-gene data frame), but that are useful for some kinds of graphs and analyses.

Value

`tidy` returns a data frame with one row per gene-sample combination, with columns

- `gene` gene name
- `sample` sample name (from column names)
- `value` expressions

If `addPheno` is `TRUE` then information from colData is added.

Examples

```r
if (require("SummarizedExperiment", "airway")) {
  data(airway)

  se <- airway
  tidy(se)
}
```
Description

Tidying methods for edge’s deSet object

Usage

## S3 method for class 'deSet'
tidy(x, addPheno = FALSE, ...)

## S3 method for class 'deSet'
augment(x, data, ...)

## S3 method for class 'deSet'
glance(x, ...)

Arguments

x  deSet object
addPheno  whether columns should be included in the tidied output for those in the ExpressionSet’s phenoData
...  extra arguments (not used)
data  Original data can be added. Default is NULL.

Details

addPheno=TRUE adds columns that are redundant (since they add per-sample information to a per-sample-per-gene data frame), but that are useful for some kinds of graphs and analyses.

Value

tidy returns a data frame with one row per gene-sample combination, with columns
gene  gene name
sample  sample name (from column names)
value  expressions on log2 scale

augment returns a data frame with
p.value  the original p-values given to qvalue
q.value  the computed q-values
lfdr  the local false discovery rate

glance returns a data frame with the model fits
Tidy methods for GRanges and GRangesList objects.

Description

Tidying methods for GRanges and GRangesList objects.

Usage

```r
## S3 method for class 'GRanges'
tidy(x, ...)

## S3 method for class 'GRangesList'
tidy(x, ...)

## S3 method for class 'GRanges'
glance(x, ...)

## S3 method for class 'GRangesList'
glance(x, ...)
```

Arguments

- `x` GRanges or GRangesList object
- `...` Not used.

Value

All tidying methods return a data.frame without rownames. tidy returns one row for each range, which contains

- start of the range
- end of the range
- width (or length) of the range
- names of the range
- strand
- seqname Name of the sequence from which the range comes (usually the chromosome)
- metadata Any included metadata, (ie, score, GC content)

For GRangesList, there will also be a column representing which group the ranges comes from. glance returns a data.frame with the number of ranges, the number of sequences, and the number of groups (if applicable).
Examples

```r
if (require("GenomicRanges", "airway")) {
  data(airway)

  # GRangesList object
  air_gr <- rowRanges(airway)
  tidy(air_gr)
  glance(air_gr)

  # GRanges object
  air_gr <- rowRanges(airway)$unlistData
  tidy(air_gr)
  glance(air_gr)
}
```
Index

* datasets
  hammer, 7
  assayDataElement, 6
  augment.deSet (tidy.deSet), 15
  augment.DGEList (edgeR_tidiers), 5
  augment.MArrayLM (limma_tidiers), 8
  augment.qvalue (qvalue_tidiers), 12
  augment_sva, 2
  biobroom, 3
  biobroom-package (biobroom), 3
  DESeq2_tidiers, 3
  edgeR_tidiers, 5
  ExpressionSet_tidiers, 6
  glance.deSet (tidy.deSet), 15
  glance.DGEEexact (edgeR_tidiers), 5
  glance.GRanges (tidy.GRanges), 16
  glance.GRangesList (tidy.GRangesList), 16
  glance.list (list_tidiers), 10
  glance.MArrayLM (limma_tidiers), 8
  glance.qvalue (qvalue_tidiers), 12
  glance_sva (augment_sva), 2
  hammer, 7
  limma_tidiers, 8
  list_tidiers, 10
  MSnSet_tidiers, 11
  qvalue_tidiers, 12
  SummarizedExperiment_tidiers, 14
  tidy.DESeqDataSet (DESeq2_tidiers), 3
  tidy.DESeqResults (DESeq2_tidiers), 3
  tidy.deSet, 15
  tidy.DGEEexact (edgeR_tidiers), 5
  tidy.DGEList (edgeR_tidiers), 5
  tidy.EList (limma_tidiers), 8
  tidy.ExpressionSet
    (ExpressionSet_tidiers), 6
  tidy.GRanges, 16
  tidy.GRangesList (tidy.GRanges), 16
  tidy.list (list_tidiers), 10
  tidy.MAList (limma_tidiers), 8
  tidy.MArrayLM (limma_tidiers), 8
  tidy.MSnSet (MSnSet_tidiers), 11
  tidy.qvalue (qvalue_tidiers), 12
  tidy.RangedSummarizedExperiment
    (SummarizedExperiment_tidiers), 14
  tidy_sva (augment_sva), 2