Package ‘biobroom’

April 22, 2016

Title Turn Bioconductor objects into tidy data frames

Version 1.2.0

Author David G. Robinson, Andrew J. Bass, Emily Nelson, John D. Storey, with contributions from Laurent Gatto

Maintainer John D. Storey <jstorey@princeton.edu> and Andrew J. Bass <ajbass@princeton.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, edgeR, qvalue, knitr, data.table, MSnbase, SummarizedExperiment

VignetteBuilder knitr

URL `https://github.com/StoreyLab/biobroom`

BugReports `https://github.com/StoreyLab/biobroom/issues`

NeedsCompilation no

R topics documented:

`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_tidiers` *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

```
## 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

`term`	The contrast being tested, as given to `results`
`gene`	gene ID
`baseMean`	mean abundance level
`estimate`	estimated log2 fold change
`stderror`	standard error in log2 fold change estimate
`statistic`	test statistic
`p.value`	p-value
`p.adjusted`	adjusted p-value

If the dataset does not contain results (`DESeq` has not been run on it), `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 `colData = TRUE`, it also merges this with the columns present in `colData(x)`.

Examples

```
# From DESeq2 documentation

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(ddsres)
    tidied

    if (require("ggplot2")) {
        ggplot(tidied, aes(p.value)) + geom_histogram(binwidth = .05) +
            facet_wrap(~ term, scale = "free_y")
    }
}
```

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

```
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

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

Arguments

`x`	ExpressionSet object
`addPheno`	whether columns should be included in the tidied output for those in the ExpressionSet's phenoData
`...`	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)
```

`hammer` *ExpressionSet results from Hammer et al 2010*

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).

Hammer, P., Banck, M. S., Amberg, R., Wang, C., Petznick, G., Luo, S., Khrebtukova, I., Schroth, G. P., Beyerlein, P., and Beutler, A. S. (2010). mRNA-seq with agnostic splice site discovery for nervous system transcriptomics tested in chronic pain. Genome research, 20(6), 847-860. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2877581/

Frazee, A. C., Langmead, B., and Leek, J. T. (2011). ReCount: a multi-experiment resource of analysis-ready RNA-seq gene count datasets. BMC Bioinformatics, 12, 449. http://bowtie-bio.sourceforge.net/recount/

Usage

```
hammer
```

Format

```
Formal class 'ExpressionSet' [package "Biobase"] with 7 slots
  ..@ experimentData   :Formal class 'MIAME' [package "Biobase"] with 13 slots
  .. .. ..@ name             : chr ""
  .. .. ..@ lab              : chr ""
  .. .. ..@ contact          : chr ""
  .. .. ..@ title            : chr ""
  .. .. ..@ abstract         : chr ""
  .. .. ..@ url              : chr ""
  .. .. ..@ pubMedIds        : chr ""
  .. .. ..@ samples          : list()
  .. .. ..@ hybridizations   : list()
  .. .. ..@ normControls     : list()
  .. .. ..@ preprocessing    : list()
  .. .. ..@ other            : list()
 .. .. ..@ .__classVersion__:Formal class 'Versions' [package "Biobase"] with 1 slot
  .. .. .. .. ..@ .Data:List of 2
  .. .. .. .. .. ..$ : int [1:3] 1 0 0
  .. .. .. .. .. ..$ : int [1:3] 1 1 0
  ..@ assayData        :<environment: 0x389fd18>
 ..@ phenoData       :Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots
  .. .. ..@ varMetadata      :'data.frame': 5 obs. of  1 variable:
  .. .. .. ..$ labelDescription: chr [1:5] NA NA NA NA ...
  .. .. ..@ data             :'data.frame': 8 obs. of  5 variables:
 .. .. .. ..$ sample.id    : Factor w/ 8 levels "SRX020088-90",..: 5 6 7 8 2 1 3 4
  .. .. .. ..$ num.tech.reps: num [1:8] 1 2 1 2 1 2 1 2
 .. .. .. ..$ protocol     : Factor w/ 2 levels "control","L5 SNL": 1 1 2 2 1 1 2 2
  .. .. .. ..$ strain       : Factor w/ 1 level "Sprague Dawley": 1 1 1 1 1 1 1 1
 .. .. .. ..$ Time        : Factor w/ 3 levels "2months","2 months",..: 2 2 2 1 3 3 3 3
  .. .. ..@ dimLabels        : chr [1:2] "sampleNames" "sampleColumns"
 .. .. ..@ .__classVersion__:Formal class 'Versions' [package "Biobase"] with 1 slot
  .. .. .. .. ..@ .Data:List of 1
  .. .. .. .. .. ..$ : int [1:3] 1 1 0
 ..@ featureData     :Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots
  .. .. ..@ varMetadata      :'data.frame': 1 obs. of  1 variable:
  .. .. .. ..$ labelDescription: chr NA
  .. .. ..@ data             :'data.frame': 29516 obs. of  1 variable:
 .. .. .. ..$ gene: Factor w/ 29516 levels "ENSRNOG00000000001",..: 1 2 3 4 5 6 7 8 9 10 ...
  .. .. ..@ dimLabels        : chr [1:2] "featureNames" "featureColumns"
 .. .. ..@ .__classVersion__:Formal class 'Versions' [package "Biobase"] with 1 slot
  .. .. .. .. ..@ .Data:List of 1
  .. .. .. .. .. ..$ : int [1:3] 1 1 0
  ..@ annotation       : chr(0)
 ..@ protocolData    :Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots
  .. .. ..@ varMetadata      :'data.frame': 0 obs. of  1 variable:
  .. .. .. ..$ labelDescription: chr(0)
  .. .. ..@ data             :'data.frame': 8 obs. of  0 variables
  .. .. ..@ dimLabels        : chr [1:2] "sampleNames" "sampleColumns"
 .. .. ..@ .__classVersion__:Formal class 'Versions' [package "Biobase"] with 1 slot
  .. .. .. .. ..@ .Data:List of 1
  .. .. .. .. .. ..$ : int [1:3] 1 1 0
  ..@ .__classVersion__:Formal class 'Versions' [package "Biobase"] with 1 slot
  .. .. ..@ .Data:List of 4
  .. .. .. ..$ : int [1:3] 2 13 0
  .. .. .. ..$ : int [1:3] 2 12 1
  .. .. .. ..$ : int [1:3] 1 3 0
  .. .. .. ..$ : int [1:3] 1 0 0
```

Value

ExpressionSet

---

limma_tidiers *Tidiers for the output of limma (linear models for microarray analysis)*

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.

Tidying method for a MA list

Tidy an EList expression object

Usage

```
## 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.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

`gene`	gene name
`sample`	sample name (from column names)
`value`	expressions on log2 scale
`weight`	present if `weights` is set
`other columns`	if present and if `addTargets` is set

Examples

```
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)
    }
}
```

`MSnSet_tidiers` *Tidying methods for Biobase's ExpressionSet objects*

Description

Tidying methods for Biobase's ExpressionSet objects

Usage

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

Arguments

`x`	MSnSet object
`addPheno`	whether columns should be included in the tidied output for those in the MSnSet's phenoData
`...`	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

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

Examples

```
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_tidiers` *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

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

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

## 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)
}
```

Index