Package 'ballgown'

April 15, 2024

Maintainer Jack Fu <jmfu@jhsph.edu>

Version 2.34.0

License Artistic-2.0

Title Flexible, isoform-level differential expression analysis

Description Tools for statistical analysis of assembled transcriptomes, including flexible differential expression analysis, visualization of transcript structures, and matching of assembled transcripts to annotation.

Depends R (>= 3.1.1), methods

Imports GenomicRanges (>= 1.17.25), IRanges (>= 1.99.22), S4Vectors (>= 0.9.39), RColorBrewer, splines, sva, limma, rtracklayer (>= 1.29.25), Biobase (>= 2.25.0), GenomeInfoDb

Suggests testthat, knitr, markdown

VignetteBuilder knitr

BugReports https://github.com/alyssafrazee/ballgown/issues

biocViews ImmunoOncology, RNASeq, StatisticalMethod, Preprocessing, DifferentialExpression

RoxygenNote 7.1.1

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

git_branch RELEASE_3_18

git_last_commit a7bfd55

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-04-15

Author Jack Fu [aut],
Alyssa C. Frazee [aut, cre],
Leonardo Collado-Torres [aut],
Andrew E. Jaffe [aut],
Jeffrey T. Leek [aut, ths]

R topics documented:

ballgown-package *The ballgown package for analysis of transcript assemblies*

Description

Super awesome transcript-level expression analysis

annotate_assembly *match assembled transcripts to annotated transcripts*

Description

match assembled transcripts to annotated transcripts

Usage

```
annotate_assembly(assembled, annotated)
```

Arguments

assembled	GRangesList object representing assembled transcripts
annotated	GRangesList object representing annotated transcripts

Details

If gown is a ballgown object, assembled can be structure(gown)$trans (or any subset). You can generate a GRangesList object containing annotated transcripts from a gtf file using the gffReadGR function and setting splitByTranscripts=TRUE.

Value

data frame, where each row contains assembledInd and annotatedInd (indexes of overlapping transcripts in assembled and annotated), and the percent overlap between the two transcripts.

Author(s)

Alyssa Frazee

Examples

```
data(bg)
gtfPath = system.file('extdata', 'annot.gtf.gz', package='ballgown')
annot = gffReadGR(gtfPath, splitByTranscript=TRUE)
info = annotate_assembly(assembled=structure(bg)$trans, annotated=annot)
```

ballgown-class *Ballgown*

Description

S4 class for storing and manipulating expression data from assembled transcriptomes

Slots

`expr` tables containing expression data for genomic features (introns, exons, transcripts)

`structure` genomic locations of features and their relationships to one another

`indexes` tables connecting components of the assembly and providing other experimental information (e.g., phenotype data and locations of read alignment files)

`dirs` directories holding data created by `tablemaker`

`mergedDate` date the ballgown object was created

`meas` which expression measurement(s) the object contains in its data slot. Vector of one or more of "rcount", "ucount", "mrcount", "cov", "cov_sd", "mcov", "mcov_sd", or "FPKM", if Tablemaker output is used, or one of "TPM" or "FPKM" if RSEM output is used. Can also be "all" for all measurements. See vignette for details.

`RSEM` TRUE if object was made from RSEM output, FALSE if object was made from Tablemaker/Cufflinks output.

Author(s)

Alyssa Frazee, Leonardo Collado-Torres, Jeff Leek

Examples

```
data(bg)
class(bg) #"ballgown"
dim(bg@expr$exon)
bg@structure$exon
head(bg@indexes$t2g)
head(bg@dirs)
bg@mergedDate
bg@meas
bg@RSEM
```

`ballgown-constructor` *constructor function for ballgown objects*

Description

constructor function for ballgown objects

Usage

```
ballgown(
  samples = NULL,
  dataDir = NULL,
  samplePattern = NULL,
  bamfiles = NULL,
  pData = NULL,
  verbose = TRUE,
  meas = "all"
)
```

Arguments

`samples`	vector of file paths to folders containing sample-specific ballgown data (generated by `tablemaker`). If `samples` is provided, `dataDir` and `samplePattern` are not used.
`dataDir`	file path to top-level directory containing sample-specific folders with ballgown data in them. Only used if `samples` is NULL.
`samplePattern`	regular expression identifying the subdirectories of\ `dataDir` containing data to be loaded into the ballgown object (and only those subdirectories). Only used if `samples` is NULL.
`bamfiles`	optional vector of file paths to read alignment files for each sample. If provided, make sure to sort properly (e.g., in the same order as `samples`). Default NULL.
`pData`	optional `data.frame` with rows corresponding to samples and columns corresponding to phenotypic variables.
`verbose`	if `TRUE`, print status messages and timing information as the object is constructed.
`meas`	character vector containing either "all" or one or more of: "rcount", "ucount", "mrcount", "cov", "cov_sd", "mcov", "mcov_sd", or "FPKM". The resulting ballgown object will only contain the specified expression measurements, for the appropriate features. See vignette for which expression measurements are available for which features. "all" creates the full object.

Details

Because experimental data is recorded so variably, it is the user's responsibility to format `pData` correctly. In particular, it's really important that the rows of `pData` (corresponding to samples) are ordered the same way as `samples` or the `dataDir`/`samplePattern` combo. You can run `list.files(path = dataDir, pattern = samplePattern)` to see the sample order if `samples` was not used.

If you are creating a ballgown object for a large experiment, this function may run slowly and use a large amount of RAM. We recommend running this constructor as a batch job and saving the resulting ballgown object as an rda file. The rda file usually has reasonable size on disk, and the object in it shouldn't take up too much RAM when loaded, so the time and memory use in creating the object is a one-time cost.

Value

an object of class `ballgown`

Author(s)

Leonardo Collado-Torres, Alyssa Frazee

See Also

`ballgownrsem`, for loading RSEM output into a ballgown object

Examples

```
bg = ballgown(dataDir=system.file('extdata', package='ballgown'),
    samplePattern='sample')
pData(bg) = data.frame(id=sampleNames(bg), group=rep(c(1,0), each=10))
```

`ballgownrsem` *load RSEM data into a ballgown object*

Description

Loads results of rsem-calculate-expression into a ballgown object for easy visualization, processing, and statistical testing

Usage

```
ballgownrsem(
  dir = "",
  samples,
  gtf,
  UCSC = TRUE,
  tfield = "transcript_id",
  attrsep = "; ",
  bamout = "transcript",
```

```
  pData = NULL,
  verbose = TRUE,
  meas = "all",
  zipped = FALSE
)
```

Arguments

dir	output directory containing RSEM output for all samples (i.e. for each run of rsem-calculate-expression)
samples	vector of sample names (i.e., of the sample_name arguments used in each RSEM run)
gtf	path to GTF file of genes/transcripts used in your RSEM reference. (where the reference location was denoted by the reference_name argument used in rsem-calculate-expression). RSEM references can be created with or without a GTF file, but currently the ballgown reader requires the GTF file.
UCSC	set to TRUE if gtf comes from UCSC: quotes will be stripped from transcript identifiers if so.
tfield	What keyword identifies transcripts in the "attributes" field of gtf? Default 'transcript_id'.
attrsep	How are attributes separated in the "attributes" field of gtf? Default '; ' (semicolon-space).
bamout	set to 'genome' if --output-genome-bam was used when running rsem-calculate-expression; set to 'none' if --no-bam-output was used when running rsem-calculate-expression; otherwise use the default ('transcript').
pData	data frame of phenotype data, with rows corresponding to samples. The first column of pData must be equal to samples, and rows must be in the same order as samples.
verbose	If TRUE (as by default), status messages are printed during data loading.
meas	character vector containing either "all" or one of "FPKM" or "TPM". The resulting ballgown object will only contain the specified expression measurement for the transcripts. "all" creates the full object.
zipped	set to TRUE if all RSEM results files have been gzipped (end) in ".gz").

Details

Currently exon- and intron-level measurements are not available for RSEM-generated ballgown objects, but development is ongoing.

Value

a ballgown object with the specified expression measurements and structure specified by GTF.

See Also

ballgown for reading Cufflinks/Tablemaker output

Examples

```
dataDir = system.file('extdata', package='ballgown')
gtf = file.path(dataDir, 'hg19_genes_small.gtf.gz')
rsemobj = ballgownrsem(dir=dataDir, samples=c('tiny', 'tiny2'), gtf=gtf,
    bamout='none', zipped=TRUE)
rsemobj
```

---

bg *Toy ballgown object*

Description

Small ballgown object created with simulated toy data, for demonstration purposes

Format

a ballgown object: 100 transcripts, 633 exons, 536 introns

Author(s)

Alyssa Frazee

Examples

```
data(bg)
bg
# ballgown instance with 100 transcripts and 20 samples
```

---

checkAssembledTx *plot annotated and assembled transcripts together*

Description

plot annotated and assembled transcripts together

Usage

```
checkAssembledTx(
  assembled,
  annotated,
  ind = 1,
  main = "Assembled and Annotated Transcripts",
  customCol = NULL
)
```

Arguments

assembled	a GRangesList object where the GRanges objects in the list represent sets of exons comprising assembled transcripts
annotated	a GRangesList object where the GRanges objects in the list represent sets of exons comprising annotated transcripts
ind	integer; index of annotated specifying which annotated transcript to plot. All transcripts (assembled and annotated) overlapping annotated[[ind]] will be plotted. Default 1.
main	optional character string giving the title for the resulting plot. Default: "Assembled and Annotated Transcripts"
customCol	optional vector of custom colors for the annotated transcripts. If not the same length as the number of annotated transcripts in the plot, recycling or truncation might occur.

Value

Plots annotated transcripts on the bottom panel (shaded in gray) and assembled transcripts on the top panel (shaded with diagonal lines).

Author(s)

Alyssa Frazee

Examples

```
gtfPath = system.file('extdata', 'annot.gtf.gz', package='ballgown')
annot = gffReadGR(gtfPath, splitByTranscript=TRUE)
data(bg)
checkAssembledTx(annotated=annot, assembled=structure(bg)$trans, ind=4)
```

---

clusterTranscripts *group a gene's assembled transcripts into clusters*

Description

group a gene's assembled transcripts into clusters

Usage

```
clusterTranscripts(gene, gown, k = NULL, method = c("hclust", "kmeans"))
```

Arguments

gene	name of gene whose transcripts will be clustered. When using Cufflinks output, usually of the form "XLOC_######"
gown	ballgown object containing experimental data
k	number of clusters to use
method	clustering method to use. Must be one of "hclust", for hierarchical clustering, or "kmeans", for k-means clustering.

Value

list with elements clusters and pctvar. clusters contains columns "cluster" and "t_id", and denotes which transcripts belong to which clusters. pctvar is only non-NULL when using k-means clustering and is the percentage of variation explained by these clusters, defined as the ratio of the between-cluster sum of squares to the total sum of squares.

Author(s)

Alyssa Frazee

See Also

hclust, kmeans, plotLatentTranscripts for visualizing the transcript clusters

Examples

```
data(bg)
clusterTranscripts('XLOC_000454', bg, k=2, method='kmeans')
# transcripts 1294 and 1301 cluster together, 91% variation explained.
```

---

collapseTranscripts *cluster a gene's transcripts and calculate cluster-level expression*

Description

cluster a gene's transcripts and calculate cluster-level expression

Usage

```
collapseTranscripts(
  gene,
  gown,
  meas = "FPKM",
  method = c("hclust", "kmeans"),
  k = NULL
)
```

Arguments

gene	which gene's transcripts should be clustered
gown	ballgown object
meas	which transcript-level expression measurement to use ('cov', average per-base coverage, or 'FPKM')
method	which clustering method to use: 'hclust' (hierarchical clustering) or 'kmeans' (k-means clustering).
k	how many clusters to use.

Value

list with two elements:

- tab, a cluster-by-sample table of expression measurements (meas, either cov or FPKM), where the expression measurement for each cluster is the mean (for 'cov') or aggregate (for 'FPKM', as in gexpr) expression measurement for all the transcripts in that cluster. This table can be used as the gowntable argument to stattest, if differential expression results for transcript *clusters* are desired.
- cl output from clusterTranscripts that was run to produce tab, for reference. Cluster IDs in the cluster component correspond to row names of tab

Author(s)

Alyssa Frazee

See Also

hclust, kmeans, clusterTranscripts, plotLatentTranscripts

Examples

```
data(bg)
collapseTranscripts(bg, gene='XLOC_000454', meas='FPKM', method='kmeans')
```

---

contains *determine if one set of GRanges fully contains any of another set of GRanges*

Description

determine if one set of GRanges fully contains any of another set of GRanges

Usage

```
contains(transcripts, cds)
```

Arguments

transcripts	GRangesList object (assume for now that it represents transcripts)
cds	GRangesList object (assume for now that it represents sets of coding sequences)

Details

If gown is a ballgown object, transcripts can be structure(gown)$trans (or any subset).

Value

vector with length equal to length(transcripts), where each entry is TRUE if the corresponding transcript contains a coding sequence (i.e., is a superset of at least one entry of cds).

Author(s)

Alyssa Frazee

Examples

```
## pretend this annotation is coding sequence:
gtfPath = system.file('extdata', 'annot.gtf.gz', package='ballgown')
annot = gffReadGR(gtfPath, splitByTranscript=TRUE)
data(bg)
results = contains(structure(bg)$trans, annot)
# results is a boolean vector
sum(results) #61
```

---

dirs *extract paths to tablemaker output*

Description

extract paths to tablemaker output

Usage

```
dirs(x)

## S4 method for signature 'ballgown'
dirs(x)
```

Arguments

x	a ballgown object

Examples

```
data(bg)
dirs(bg)
```

eexpr *extract exon-level expression measurements from ballgown objects*

Description

extract exon-level expression measurements from ballgown objects

Usage

```
eexpr(x, meas = "rcount")

## S4 method for signature 'ballgown'
eexpr(x, meas = "rcount")
```

Arguments

x	a ballgown object
meas	type of measurement to extract. Can be "rcount", "ucount", "mrcount", "cov", "mcov", or "all". Default "rcount".

Value

exon-by-sample matrix containing exon-level expression values (measured by meas). If meas is "all", or x@RSEM is TRUE, a data frame is returned, containing all measurements and location information.

Examples

```
data(bg)
exon_rcount_matrix = eexpr(bg)
exon_ucount_matrix = eexpr(bg, 'ucount')
exon_data_frame = eexpr(bg, 'all')
```

expr *extract expression components from ballgown objects*

Description

extract expression components from ballgown objects

Usage

```
expr(x)

## S4 method for signature 'ballgown'
expr(x)
```

Arguments

x	a ballgown object

Value

list containing elements `intron`, `exon`, and `trans`, which are feature-by-sample data frames of expression data.

See Also

`texpr`, `gexpr`, `eexpr`, `iexpr`

Examples

```
data(bg)
names(expr(bg))
class(expr(bg))
dim(expr(bg)$exon)
```

---

expr<- *Replacement method for expr slot in ballgown objects*

Description

Replacement method for expr slot in ballgown objects

Usage

```
expr(x) <- value

## S4 replacement method for signature 'ballgown'
expr(x) <- value
```

Arguments

x	a ballgown object
value	the updated value for `expr(x)` or a subcomponent

Examples

```
data(bg)
n = ncol(bg@expr$trans)
#multiply all transcript expression measurements by 10:
bg@expr$trans[,11:n] = 10*bg@expr$trans[11:n]
```

`exprfilter` *subset ballgown objects using an expression filter*

Description

Create a new ballgown object containing only transcripts passing a mean expression filter

Usage

```
exprfilter(gown, cutoff, meas = "FPKM")
```

Arguments

`gown`	a ballgown object
`cutoff`	transcripts must have mean expression across samples above this value to be included in the return
`meas`	how should transcript expression be measured? Default FPKM, but can also be `'cov'`.

Value

A new ballgown object derived from `gown`, but only containing transcripts (and associated exons/introns) with mean `meas` greater than `cutoff` across all samples.

See Also

`subset`

Examples

```
data(bg)
# make a ballgown object containing only transcripts with mean FPKM > 100:
over100 = exprfilter(bg, cutoff=100)
```

`geneIDs` *get gene IDs from a ballgown object*

Description

get gene IDs from a ballgown object

Usage

```
geneIDs(x)

## S4 method for signature 'ballgown'
geneIDs(x)
```

Arguments

x	a ballgown object

Details

This vector differs from that produced by geneNames in that geneIDs produces names of loci created during the assembly process, not necessarily annotated genes.

Value

named vector of gene IDs included in the ballgown object. If object was created using Tablemaker, these gene IDs will be of the form "XLOC_*". Vector is named and ordered by corresponding numeric transcript ID.

See Also

geneNames

Examples

```
data(bg)
geneIDs(bg)
```

---

geneNames *get gene names from a ballgown object*

Description

get gene names from a ballgown object

Usage

```
geneNames(x)

## S4 method for signature 'ballgown'
geneNames(x)
```

Arguments

x	a ballgown object

Details

This vector differs from that produced by geneIDs in that geneNames produces *annotated* gene names that correspond to assembled transcripts. The return will be empty/blank/NA if the transcriptome assembly is de novo (i.e., was not compared to an annotation before the ballgown object was created). See `getGenes` for matching transcripts to gene names. Some entries of this vector will be empty/blank/NA if the corresponding transcript did not overlap any annotated genes.

Value

named vector of gene names included in the ballgown object, named and ordered by corresponding numeric transcript ID.

See Also

`geneIDs`

Examples

```
data(bg)
# this is a de novo assembly, so it does not contain gene info as it stands
# but we can add it:
annot = system.file('extdata', 'annot.gtf.gz', package='ballgown')
gnames = getGenes(annot, structure(bg)$trans, UCSC=FALSE)
gnames_first = lapply(gnames, function(x) x[1]) #just take 1 overlapping gene
expr(bg)$trans$gene_name = gnames_first

# now we can extract these gene names:
geneNames(bg)
```

---

getAttributeField *extract a specific field of the "attributes" column of a data frame created from a GTF/GFF file*

Description

extract a specific field of the "attributes" column of a data frame created from a GTF/GFF file

Usage

```
getAttributeField(x, field, attrsep = "; ")
```

Arguments

`x`	vector representing the "attributes" column of GTF/GFF file
`field`	name of the field you want to extract from the "attributes" column
`attrsep`	separator for the fields in the attributes column. Defaults to '; ', the separator for GTF files outputted by Cufflinks.

Value

vector of nucleotide positions included in the transcript

Author(s)

Wolfgang Huber, in the davidTiling R package (LGPL license)

See Also

gffRead for creating a data frame from a GTF/GFF file, and http://useast.ensembl.org/info/website/upload/gff.html for specifics of the GFF/GTF file format.

Examples

```
gtfPath = system.file('extdata', 'annot.gtf.gz', package='ballgown')
gffdata = gffRead(gtfPath)
gffdata$transcriptID = getAttributeField(gffdata$attributes,
  field = "transcript_id")
```

---

getGenes *label assembled transcripts with gene names*

Description

label assembled transcripts with gene names

Usage

```
getGenes(gtf, assembled, UCSC = TRUE, attribute = "gene_id")
```

Arguments

gtf	path to a GTF file containing locations of annotated transcripts
assembled	GRangesList object, with each set of ranges representing exons of an assembled transcript.
UCSC	set to TRUE if you're using a UCSC gtf file. (Requires some extra text processing).
attribute	set to attribute name in gtf that gives desired gene identifiers. Default "gene_id"; another commone one is "gene_name" (for the gene symbol).

Details

chromosome labels in gtf and assembled should match. (i.e., you should provide the path to a gtf corrsponding to the same annotation you used when constructing assembled)

Value

an IRanges CharacterList of the same length as assembled, providing the name(s) of the gene(s) that overlaps each transcript in assembled.

Author(s)

Alyssa Frazee, Andrew Jaffe

Examples

```
data(bg)
gtfPath = system.file('extdata', 'annot.gtf.gz', package='ballgown')
geneoverlaps = getGenes(gtfPath, structure(bg)$trans, UCSC=FALSE)
```

---

gexpr *extract gene-level expression measurements from ballgown objects*

Description

For objects created with Cufflinks/Tablemaker, gene-level measurements are calculated by appropriately combining FPKMs from the transcripts comprising the gene. For objects created with RSEM, gene-level measurements are extracted directly from the RSEM output.

Usage

```
gexpr(x)

## S4 method for signature 'ballgown'
gexpr(x)
```

Arguments

x	a ballgown object

Value

gene-by-sample matrix containing per-sample gene measurements.

Examples

```
data(bg)
gene_matrix = gexpr(bg)
```

gffRead *read in GTF/GFF file as a data frame*

Description

read in GTF/GFF file as a data frame

Usage

```
gffRead(gffFile, nrows = -1, verbose = FALSE)
```

Arguments

gffFile	name of GTF/GFF on disk
nrows	number of rows to read in (default -1, which means read all rows)
verbose	if TRUE, print status info at beginning and end of file read. Default FALSE.

Value

data frame representing the GTF/GFF file

Author(s)

Kasper Hansen

See Also

getAttributeField to extract data from "attributes" column; http://useast.ensembl.org/info/website/upload/gff.html for more information on the GTF/GFF file format.

Examples

```
gtfPath = system.file('extdata', 'annot.gtf.gz', package='ballgown')
annot = gffRead(gtfPath)
```

gffReadGR *read in gtf file as GRanges object*

Description

(very) light wrapper for rtracklayer::import

Usage

```
gffReadGR(
  gtf,
  splitByTranscript = FALSE,
  identifier = "transcript_id",
  sep = "; "
)
```

Arguments

gtf	name of GTF/GFF file on disk
splitByTranscript	if TRUE, return a GRangesList of transcripts; otherwise return a GRanges object containing all genomic features in gtf. Default FALSE.
identifier	name of transcript identifier column of attributes field in gtf. Default "transcript_id". Only used if splitByTranscript is TRUE.
sep	field separator in the attributes field of gtf. Default "; " (semicolon + space). Only used if splitByTranscript is TRUE.

Value

if splitByTranscript is FALSE, an object of class GRanges representing the genomic features in gtf. If splitByTranscript is TRUE, an object of class GRangesList, where each element is a GRanges object corresponding to an annotated transcript (designated in names).

Author(s)

Alyssa Frazee

See Also

gffRead for reading in a GTF file as a data frame rather than a GRanges/GRangesList object.

Examples

```
gtfPath = system.file('extdata', 'annot.gtf.gz', package='ballgown')

# read in exons as GRanges:
annotgr = gffReadGR(gtfPath)

# read in groups of exons as transcripts, in GRangesList:
transcripts_grl = gffReadGR(gtfPath, splitByTranscript=TRUE)
```

iexpr *extract transcript-level expression measurements from ballgown objects*

Description

extract transcript-level expression measurements from ballgown objects

Usage

```
iexpr(x, meas = "rcount")

## S4 method for signature 'ballgown'
iexpr(x, meas = "rcount")
```

Arguments

x	a ballgown object
meas	type of measurement to extract. Can be "rcount", "ucount", "mrcount", or "all". Default "rcount".

Value

intron-by-sample matrix containing the number of reads (measured as specified by meas) supporting each intron, in each sample. If meas is "all", a data frame is returned, containing all measurements and location information.

Examples

```
data(bg)
intron_rcount_matrix = iexpr(bg)
intron_data_frame = iexpr(bg, 'all')
```

indexes *extract the indexes from ballgown objects*

Description

extract the indexes from ballgown objects

Usage

```
indexes(x)

## S4 method for signature 'ballgown'
indexes(x)
```

Arguments

x	a ballgown object

Value

list containing elements `e2t`, `i2t`, `t2g`, `bamfiles`, and `pData`, where `e2t` and `i2t` are data frames linking exons and introns (respectively) to transcripts, `t2g` is a data frame linking transcripts to genes, and `bamfiles` and `pData` are described in `?ballgown`.

Examples

```
data(bg)
names(indexes(bg))
class(indexes(bg))
head(indexes(bg)$t2g)
```

---

indexes<- *Replace method for indexes slot in ballgown objects*

Description

Replace method for indexes slot in ballgown objects

Usage

```
indexes(x) <- value

## S4 replacement method for signature 'ballgown'
indexes(x) <- value
```

Arguments

x	a ballgown object
value	the updated value for `indexes(x)` or a subcomponent

Examples

```
data(bg)
indexes(bg)$bamfiles = paste0('/path/to/bamfolder/',
  sampleNames(bg), '_accepted_hits.bam')
```

last *get the last element*

Description

get the last element

Usage

```
last(x)
```

Arguments

x	anything you can call `tail` on (vector, data frame, etc.)

Details

this function is made of several thousand lines of complex code, so be sure to read it carefully.

Value

the last element of `x`

Author(s)

Alyssa Frazee

Examples

```
last(c('h', 'e', 'l', 'l', 'o'))
```

mergedDate *extract package version & creation date from ballgown object*

Description

extract package version & creation date from ballgown object

Usage

```
mergedDate(x)

## S4 method for signature 'ballgown'
mergedDate(x)
```

Arguments

x	a ballgown object

Examples

```
data(bg)
mergedDate(bg)
```

---

pctOverlap *calculate percent overlap between two GRanges objects*

Description

calculate percent overlap between two GRanges objects

Usage

```
pctOverlap(tx1, tx2)
```

Arguments

tx1	GRanges object
tx2	GRanges object

Details

In the ballgown context, tx1 and tx2 are two transcripts, each represented by GRanges objects whose ranges represent the exons comprising the transcripts. The percent overlap is the number of nucleotides falling within both transcripts divided by the number of nucleotides falling within either transcript. Useful as a measure of transcript closeness (as it is essentially Jaccard distance).

Value

percent overlap between tx1 and tx2, as defined by the ratio of the intersection of tx1 and tx2 to the union of tx1 and tx2.

Author(s)

Alyssa Frazee

Examples

```
data(bg)
gtfPath = system.file('extdata', 'annot.gtf.gz', package='ballgown')
annot_grl = gffReadGR(gtfPath, splitByTranscript=TRUE)
pctOverlap(structure(bg)$trans[[2]], annot_grl[[369]]) #79.9%
```

pData *extract phenotype data from a ballgown object*

Description

extract phenotype data from a ballgown object

Usage

```
pData(object)

## S4 method for signature 'ballgown'
pData(object)
```

Arguments

object	a ballgown object

Value

sample-by-phenotype data frame

Examples

```
data(bg)
pData(bg)
```

pData<- *Replacement method for pData slot in ballgown objects*

Description

Replacement method for pData slot in ballgown objects

Usage

```
pData(object) <- value

## S4 replacement method for signature 'ballgown,ANY'
pData(object) <- value
```

Arguments

object	a ballgown object
value	the updated value for pData(x).

Examples

```
# add "timepoint" covariate to ballgown object:
data(bg) # already contains pData
pData(bg) = data.frame(pData(bg), timepoint=rep(1:10, 2))
head(pData(bg))
```

`plotLatentTranscripts` *cluster assembled transcripts and plot the results*

Description

This is an experimental, first-pass function that clusters assembled transcripts based on their overlap percentage, then plots and colors the transcript clusters.

Usage

```
plotLatentTranscripts(
  gene,
  gown,
  method = c("hclust", "kmeans"),
  k = NULL,
  choosek = c("var90", "thumb"),
  returncluster = TRUE,
  labelTranscripts = TRUE,
  ...
)
```

Arguments

`gene`	string, name of gene whose transcripts should be clustered (e.g., "XLOC_000001")
`gown`	object of class `ballgown` being used for analysis
`method`	clustering method to use. Currently can choose from hierarchical clustering (`hclust`) or K-means (`kmeans`). More methods are in development.
`k`	number of transcripts clusters to use. By default, `k` is `NULL` and thus is chosen using a rule of thumb, but providing `k` overrides those rules of thumb.
`choosek`	if `k` is not provided, how should the number of clusters be chosen? Must be one of "var90" (choose a `k` that explains 90 percent of the observed variation) or "thumb" (`k` is set to be approximately `sqrt(n)`, where `n` is the total number of transcripts for `gene`)
`returncluster`	if TRUE (as it is by default), return the results of the call to `clusterTrancsripts` so the data is available for later use. Nothing is returned if FALSE.
`labelTranscripts`	if TRUE (as it is by default), print transcript IDs on the y-axis
`...`	other arguments to pass to plotTranscripts

Value

if returncluster is TRUE, the transcript clusters are returned as described in clusterTranscripts. A plot of the transcript clusters is also produced, in the style of plotTranscripts.

Author(s)

Alyssa Frazee

See Also

clusterTranscripts, plotTranscripts

Examples

```
data(bg)
plotLatentTranscripts('XLOC_000454', bg, method='kmeans', k=2)
```

---

plotMeans *visualize transcript abundance by group*

Description

visualize transcript abundance by group

Usage

```
plotMeans(
  gene,
  gown,
  overall = FALSE,
  groupvar,
  groupname = "all",
  meas = c("cov", "FPKM", "rcount", "ucount", "mrcount", "mcov"),
  colorby = c("transcript", "exon"),
  legend = TRUE,
  labelTranscripts = FALSE
)
```

Arguments

gene	name of gene whose transcripts will be plotted. When using Cufflinks/Tablemaker output, usually of the form "XLOC_######"
gown	ballgown object containing experimental and phenotype data
overall	if TRUE, color features by the overall (experiment-wide) mean rather than a group-specific mean

groupvar	string representing the name of the variable denoting which sample belongs to which group. Can be "none" (if you want the study-wide mean), or must correspond to the name of a column of pData(gown). Usually a categorical variable.
groupname	string representing which group's expression means you want to plot. Can be "none" (if you want the study-wide mean), "all" (if you want a multipanel plot of each group's mean expression), or any of the levels of groupvar.
meas	type of expression measurement to plot. One of "cov", "FPKM", "rcount", "ucount", "mrcount", or "mcov". Not all types are valid for all features. (See description of tablemaker output for more information).
colorby	one of "transcript" or "exon", indicating which feature's abundances should dictate plot coloring.
legend	if TRUE (as it is by default), a color legend is drawn on top of the plot indicating the scale for feature abundances.
labelTranscripts	if TRUE, transcript ids are labeled on the left side of the plot. Default FALSE.

Value

produces a plot of the transcript structure for the specified gene in the current graphics device, colored by study-wide or group-specific mean expression level.

Author(s)

Alyssa Frazee

See Also

plotTranscripts

Examples

```
data(bg)
plotMeans('XLOC_000454', bg, groupvar='group', meas='FPKM',
  colorby='transcript')
```

---

plotTranscripts *visualize structure of assembled transcripts*

Description

visualize structure of assembled transcripts

Usage

```
plotTranscripts(
  gene,
  gown,
  samples = NULL,
  colorby = "transcript",
  meas = "FPKM",
  legend = TRUE,
  labelTranscripts = FALSE,
  main = NULL,
  blackBorders = TRUE,
  log = FALSE,
  logbase = 2,
  customCol = NULL,
  customOrder = NULL
)
```

Arguments

gene	name of gene whose transcripts will be plotted. When using Cufflinks output, usually of the form "XLOC_######"
gown	ballgown object containing experimental and phenotype data
samples	vector of sample(s) to plot. Can be 'none' if only one plot (showing transcript structure in gray) is desired. Use sampleNames(gown) to see sample names for gown. Defaults to sampleNames(gown)[1].
colorby	one of "transcript", "exon", or "none", indicating which feature's abundances should dictate plot coloring. If "none", all transcripts are drawn in gray.
meas	which expression measurement to color features by, if any. Must match an available measurement for whatever feature you're plotting.
legend	if TRUE (as it is by default), a color legend is drawn on top of the plot indicating scales for feature abundances.
labelTranscripts	if TRUE, transcript ids are labeled on the left side of the plot. Default FALSE.
main	optional string giving the desired plot title.
blackBorders	if TRUE, exon borders are drawn in black. Otherwise, they are drawn in the same color as their transcript or exon. Switching blackBorders to FALSE can be useful for visualizing abundances for skinny exons and/or smaller plots, which can be the case when length(samples) is large.
log	if TRUE, color transcripts on the log scale. Default FALSE. To account for expression values of 0, we add 1 to all expression values before taking the log.
logbase	log base to use if log = TRUE. Default 2.
customCol	an optional vector of custom colors to color transcripts by. There must be the same number of colors as transcripts in the gene being plotted.
customOrder	an optional vector of transcript ids (matching ids in texpr(gown, 'all')$t_id), indicating which order transcripts will appear in the plot. All transcripts in gene must appear in the vector exactly once.

Value

produces a plot of the transcript structure for the specified gene in the current graphics device.

Author(s)

Alyssa Frazee

See Also

plotMeans, plotLatentTranscripts

Examples

```
data(bg)

# plot one gene for one sample:
plotTranscripts(gene='XLOC_000454', gown=bg, samples='sample12', meas='FPKM',
    colorby='transcript',
    main='transcripts from gene XLOC_000454: sample 12, FPKM')

# plot one gene for many samples:
plotTranscripts('XLOC_000454', bg,
    samples=c('sample01', 'sample06', 'sample12', 'sample19'),
    meas='FPKM', colorby='transcript')
```

---

sampleNames *get names of samples in a ballgown objects*

Description

get names of samples in a ballgown objects

Usage

```
sampleNames(object)

## S4 method for signature 'ballgown'
sampleNames(object)
```

Arguments

object	a ballgown object

Value

vector of sample IDs for x. If pData exists, samples in its rows correspond to samples in sampleNames(x) (in order).

Examples

```
data(bg)
sampleNames(bg)
```

`seqnames` *get sequence (chromosome) names from ballgown object*

Description

get sequence (chromosome) names from ballgown object

Usage

```
seqnames(x)

## S4 method for signature 'ballgown'
seqnames(x)
```

Arguments

x	a ballgown object

Value

vector of sequence (i.e., chromosome) names included in the ballgown object

Examples

```
data(bg)
seqnames(bg)
```

`stattest` *statistical tests for differential expression in ballgown*

Description

Test each transcript, gene, exon, or intron in a ballgown object for differential expression, using comparisons of linear models.

Usage

```
stattest(
  gown = NULL,
  gowntable = NULL,
  pData = NULL,
  mod = NULL,
  mod0 = NULL,
  feature = c("gene", "exon", "intron", "transcript"),
  meas = c("cov", "FPKM", "rcount", "ucount", "mrcount", "mcov"),
  timecourse = FALSE,
  covariate = NULL,
  adjustvars = NULL,
  gexpr = NULL,
  df = 4,
  getFC = FALSE,
  libadjust = NULL,
  log = TRUE
)
```

Arguments

gown	name of an object of class ballgown
gowntable	matrix or matrix-like object with rownames representing feature IDs and columns representing samples, with expression estimates in the cells. Provide the feature name with feature. You must provide exactly one of gown or gowntable. NB: gowntable is log-transformed within stattest if log is TRUE, so provide unlogged expression values in gowntable.
pData	Required if gowntable is provided: data frame giving phenotype data for the samples in the columns of gowntable. (Rows of pData correspond to columns of gowntable). If gown is used instead, it must have a non-null, valid pData slot (and the pData argument to stattest should be left NULL).
mod	object of class model.matrix representing the design matrix for the linear regression model including covariates of interest
mod0	object of class model.matrix representing the design matrix for the linear regression model without the covariates of interest.
feature	the type of genomic feature to be tested for differential expression. If gown is used, must be one of "gene", "transcript", "exon", or "intron". If gowntable is used, this is just used for labeling and can be whatever the rows of gowntable represent.
meas	the expression measurement to use for statistical tests. Must be one of "cov", "FPKM", "rcount", "ucount", "mrcount", or "mcov". Not all expression measurements are available for all features. Leave as default if gowntable is provided.
timecourse	if TRUE, tests whether or not the expression profiles of genomic features vary over time (or another continuous covariate) in the study. Default FALSE. Natural splines are used to fit time profiles, so you must have more timepoints than degrees of freedom used to fit the splines. The default df is 4.

covariate	string representing the name of the covariate of interest for the differential expression tests. Must correspond to the name of a column of pData(gown). If timecourse=TRUE, this should be the study's time variable.
adjustvars	optional vector of strings representing the names of potential confounders. Must correspond to names of columns of pData(gown).
gexpr	optional data frame that is the result of calling gexpr(gown)). (You can speed this function up by pre-creating gexpr(gown).)
df	degrees of freedom used for modeling expression over time with natural cubic splines. Default 4. Only used if timecourse=TRUE.
getFC	if TRUE, also return estimated fold changes (adjusted for library size and confounders) between populations. Only available for 2-group comparisons at the moment. Default FALSE.
libadjust	library-size adjustment to use in linear models. By default, the adjustment is defined as the sum of the sample's log expression measurements below the 75th percentile of those measurements. To use a different library-size adjustment, provide a numeric vector of each sample's adjustment value. Entries of this vector correspond to samples in in rows of pData. If no library size adjustment is desired, set to FALSE.
log	if TRUE, outcome variable in linear models is log(expression+1), otherwise it's expression. Default TRUE.

Details

At minimum, you need to provide a ballgown object or count table, the type of feature you want to test (gene, transcript, exon, or intron), the expression measurement you want to use (FPKM, cov, rcount, etc.), and the covariate of interest, which must be the name of one of the columns of the 'pData' component of your ballgown object (or provided pData). This covariate is automatically converted to a factor during model fitting in non-timecourse experiments.

By default, models are fit using log2(meas + 1) as the outcome for each feature. To disable the log transformation, provide 'log = FALSE' as an argument to 'stattest'. You can use the gowntable option if you'd like to to use a different transformation.

Library size adjustment is performed by default by using the sum of the log nonzero expression measurements for each sample, up to the 75th percentile of those measurements. This adjustment can be disabled by setting libadjust=FALSE. You can use mod and mod0 to specify alternative library size adjustments.

mod and mod0 are optional arguments. If mod is specified, you must also specify mod0. If neither is specified, mod0 defaults to the design matrix for a model including only a library-size adjustment, and mod defaults to the design matrix for a model including a library-size adjustment and covariate. Note that if you supply mod and mod0, covariate, timecourse, adjustvars, and df are ignored, so make sure your covariate of interest and all appropriate confounder adjustments, including library size, are specified in mod and mod0. By default, the library-size adjustment is the sum of all counts below the 75th percentile of nonzero counts, on the log scale (log2 + 1).

Full model details are described in the supplement of http://biorxiv.org/content/early/2014/03/30/003665.

Value

data frame containing the columns `feature`, `id` representing feature id, `pval` representing the p-value for testing whether this feature was differentially expressed according to `covariate`, and `qval`, the estimated false discovery rate using this feature's signal strength as a significance cutoff. An additional column, `fc`, is included if `getFC` is `TRUE`.

Author(s)

Jeff Leek, Alyssa Frazee

References

http://biorxiv.org/content/early/2014/03/30/003665

Examples

```
data(bg)

# two-group comparison:
stat_results = stattest(bg, feature='transcript', meas='FPKM',
  covariate='group')

# timecourse test:
pData(bg) = data.frame(pData(bg), time=rep(1:10, 2)) #dummy time covariate
timecourse_results = stattest(bg, feature='transcript', meas='FPKM',
  covariate='time', timecourse=TRUE)

# timecourse test, adjusting for group:
group_adj_timecourse_results = stattest(bg, feature='transcript',
  meas='FPKM', covariate='time', timecourse=TRUE, adjustvars='group')

# custom model matrices:
### create example data:
set.seed(43)
sex = sample(c('M','F'), size=nrow(pData(bg)), replace=TRUE)
age = sample(21:52, size=nrow(pData(bg)), replace=TRUE)

### create design matrices:
mod = model.matrix(~ sex + age + pData(bg)$group + pData(bg)$time)
mod0 = model.matrix(~ pData(bg)$group + pData(bg)$time)

### build model:
adjusted_results = stattest(bg, feature='transcript', meas='FPKM',
  mod0=mod0, mod=mod)
```

structure *extract structure components from ballgown objects*

Description

extract structure components from ballgown objects

Usage

```
structure(x)

## S4 method for signature 'ballgown'
structure(x)
```

Arguments

x	a ballgown object

Value

list containing elements `intron`, `exon`, and `trans`. `exon` and `intron` are `GRanges` objects, where each range is an exon or intron, and `trans` is a `GRangesList` object, where each `GRanges` element is a set of exons representing a transcript.

Examples

```
data(bg)
names(structure(bg))
class(structure(bg))
structure(bg)$exon
```

subset *subset ballgown objects to specific samples or genomic locations*

Description

subset ballgown objects to specific samples or genomic locations

Usage

```
subset(x, ...)

## S4 method for signature 'ballgown'
subset(x, cond, genomesubset = TRUE)
```

Arguments

x	a ballgown object
...	further arguments to generic subset
cond	Condition on which to subset. See details.
genomesubset	if TRUE, subset x to a specific part of the genome. Otherwise, subset x to only include specific samples. TRUE by default.

Details

To use subset, you must provide the cond argument as a string representing a logical expression specifying your desired subset. The subset expression can either involve column names of texpr(x, "all") (if genomesubset is TRUE) or of pData(x) (if genomesubset is FALSE). For example, if you wanted a ballgown object for only chromosome 22, you might call subset(x, "chr == 'chr22'"). (Be sure to handle quotes within character strings appropriately).

Value

a subsetted ballgown object, containing only the regions or samples satisfying cond.

Author(s)

Alyssa Frazee

Examples

```
data(bg)
bg_twogenes = subset(bg, "gene_id=='XLOC_000454' | gene_id=='XLOC_000024'")
bg_twogenes
# ballgown instance with 4 assembled transcripts and 20 samples

bg_group0 = subset(bg, "group == 0", genomesubset=FALSE)
bg_group0
# ballgown instance with 100 assembled transcripts and 10 samples
```

---

texpr *extract transcript-level expression measurements from ballgown objects*

Description

extract transcript-level expression measurements from ballgown objects

Usage

```
texpr(x, meas = "FPKM")

## S4 method for signature 'ballgown'
texpr(x, meas = "FPKM")
```

Arguments

x	a ballgown object
meas	type of measurement to extract. Can be "cov", "FPKM", or "all". Default "FPKM".

Value

transcript-by-sample matrix containing expression values (measured by meas). If meas is "all", a data frame is returned, containing all measurements and location information.

Examples

```
data(bg)
transcript_fpkm_matrix = texpr(bg)
transcript_data_frame = texpr(bg, 'all')
```

---

tGene *Connect a transcript to its gene*

Description

find the gene to which a transcript belongs

Usage

```
tGene(bg, transcript, tid = TRUE, gid = TRUE, warnme = TRUE)
```

Arguments

bg	ballgown object
transcript	transcript identifier
tid	set to TRUE if transcript is a numeric transcript identifier (i.e., t_id in expression tables), or FALSE if transcript is a named identifie (e.g., TCONS_000001 or similar.
gid	if FALSE, return the gene *name* associated with transcript in bg instead of the gene *id*, which is returned by default. Take care to remember that not all ballgown objects include gene *name* information. (They do all include gene IDs).
warnme	if TRUE, and if gid is FALSE, print a warning if no gene name is available for the transcript. This could either mean the transcript didn't overlap an annotated gene, or that no gene names were included when bg was created.

Examples

```
data(bg)
tGene(bg, 10)
tGene(bg, 'TCONS_00000010', tid=FALSE)
tGene(bg, 10, gid=FALSE) #empty: no gene names included in bg.
```

`transcriptIDs` *get numeric transcript IDs from a ballgown object*

Description

get numeric transcript IDs from a ballgown object

Usage

```
transcriptIDs(x)

## S4 method for signature 'ballgown'
transcriptIDs(x)
```

Arguments

x	a ballgown object

Value

vector of numeric transcript IDs included in the ballgown object

Examples

```
data(bg)
transcriptIDs(bg)
```

`transcriptNames` *get transcript names from a ballgown object*

Description

get transcript names from a ballgown object

Usage

```
transcriptNames(x)

## S4 method for signature 'ballgown'
transcriptNames(x)
```

Arguments

x	a ballgown object

Value

vector of transcript names included in the ballgown object. If object was created using Cufflinks/Tablemaker, these transcript names will be of the form "TCONS_*". Return vector is named and ordered by corresponding numeric transcript ID.

Examples

```
data(bg)
transcriptNames(bg)
```

---

writeFiles *write files to disk from ballgown object*

Description

create tablemaker-like files on disk from a ballgown object

Usage

```
writeFiles(gown, dataDir)
```

Arguments

gown	ballgown object
dataDir	top-level directory for sample-specific folders

Examples

```
data(bg)
writeFiles(bg, dataDir=getwd())
```

Index