Package ‘artemis’

December 30, 2015

Type Package

Title A package that complements Kallisto for quick, informative *seq analysis

Version 0.9.34

Date 2015-12-22

Author Tim Triche, Jr., Anthony Colombo, Harold Pimentel

Imports biomaRt, edgeR, erccdashboard, pathview, parallel, SummarizedExperiment, BiocGenerics, Rsamtools, RUVSeq, TxDbLite, tools, limma, rhdf5, matrixStats, GenomicRanges, GenomicFeatures, Matrix, KEGGREST, ggplot2

Suggests roxygen2, knitr, artemisData, jsonlite, qusage, Biobase, GenomeInfoDb, graphite, beeswarm,

VignetteBuilder knitr

Maintainer Tim Triche, Jr. <tim.triche@gmail.com>

Description Artemis was the hunting companion of Kallisto. This package wraps various aspects of workflow automation (assembling transcriptomes, hashing indices after they have been built, checking versions, and generally avoiding annoyance factors) useful in the day-to-day operation of a transcriptome aligner or pseudoaligner.
Tools for data extraction and (progressive) annotation of combined or individual transcriptomes, quality control and normalization (ERCC control plots and RUVSeq normalization), interpretation (gene-level, transcript-level, pathway- and network-level output), guidelines for validation (sample size estimation by simulation), and interactions with cloud computing services such as BaseSpace are loosely coupled within the default package workflow.
Please be aware that, while Artemis is licensed under the GPL, Kallisto itself is free FOR NON-COMMERCIAL USAGE ONLY. For-profit use of Kallisto requires a licensing agreement executed with the Regents of the University of California. Kallisto is not included within this package; Artemis assumes it is in the user's PATH.
Artemis' component parts are not restricted in this fashion (although we cannot guarantee that all [pseudo]aligners will

necessarily work as well as Kallisto in end-to-end pipelines), and
most downstream analyses can proceed from any SummarizedExperiment.
Artemis' annotation functions, in particular, can greatly ease data
integration, as when comparing in-house results to those of major
genome and transcriptome sequencing projects such as TCGA and ICGC.
Sample size estimates and pathway analyses are similarly decoupled
from the underlying source of transcript abundance estimates.

License GPL (>= 2)

RoxygenNote 5.0.1

R topics documented:

artemis-package *various utility functions for fast, informative RNAseq analysis*

Description

utility functions for Artemis (hunting companion of Kallisto)

Details

Forthcoming; see references for some background.

Author(s)

Tim Triche, Jr. <tim.triche@gmail.com>

References

Kallisto: http://arxiv.org/abs/1505.02710 limma/voom: http://genomebiology.com/2014/15/2/R29 ssizeRNA: http://bioinformatics.oxfordjournals.org/content/23/6/739 ReactomePA: http://www.bioconductor.org/packages/ReactomePA

Examples

```
## a single Kallisto run with bootstraps
h5 <- system.file("extdata", "abundance.h5", package="artemis", mustWork=T)
results <- fetchKallisto(h5)
ri <- system.file("extdata", "run_info.json", package="artemis", mustWork=T)
runinfo <- fetchRunInfo(ri)

## an experiment with multiple replicates per condition
```

annotateFeatures *annotate features (genes or transcripts) against (say) EnsemblDb this is becoming the default dispatcher for almost all annotation*

Description

annotate features (genes or transcripts) against (say) EnsemblDb this is becoming the default dispatcher for almost all annotation

Usage

```
annotateFeatures(kexp, level = c(NA, "gene", "transcript"),
  what = c("KallistoExperiment", "GRanges"), ...)
```

Arguments

`kexp`	a kexp
`level`	at what level has the data been summarized? (guess)
`what`	what data structure shall we return (KallistoExperiment)

Value

a GRanges or a KallistoExperiment, depending on ‘what‘

---

collapseBundles

Collapse bundles of transcripts, discard any that represent pointless tests, and optionally prune any whose joined bundle IDs tend to choke downstream packages for e.g. pathway- or network-based enrichment analysis. Note that this function may or may not be optimal for your RNAseq experiment. Please refer to ’Details’ for some thought exercises about the nature of ’genes’.

Description

Collapse bundles of transcripts, discard any that represent pointless tests, and optionally prune any whose joined bundle IDs tend to choke downstream packages for e.g. pathway- or network-based enrichment analysis. Note that this function may or may not be optimal for your RNAseq experiment. Please refer to ’Details’ for some thought exercises about the nature of ’genes’.

Usage

```
collapseBundles(kexp, bundleID = "gene_id", read.cutoff = 1,
  discardjoined = TRUE)
```

Arguments

`kexp`	A KallistoExperiment (or something very much like it)
`bundleID`	The column (in mcols(features(kexp))) of the bundle IDs
`read.cutoff`	Discard transcripts and bundles with < this many counts
`discardjoined`	Discard bundles with IDs "joined" by a ";"? (TRUE)

Details

This function sums the estimated counts for each transcript within a bundle of transcripts (where "bundle" is a user-defined identifier, often but not always a ’gene’, sometimes a biotype or a class of repeat elements). The default approach is to discard all rows where the maximum count is less than the specified read.cutoff. Since the default cutoff is 1, this means discarding transcripts (and bundles) that were not be detected in any sample. (Filtering tends to increase statistical power at a given false-positive rate per Bourgon et al, 2010, http://www.pnas.org/content/107/21/9546.long)

Value

a matrix of summarized counts per sample bundle

See Also

collapseTranscripts

`collapseTpm` *Collapse bundles of transcripts, discard any with (default) < 1TPM/bundle, and optionally prune any whose joined bundle IDs tend to choke downstream packages for e.g. pathway- or network-based enrichment analysis. Note that this function may or may not be optimal for your RNAseq experiment. Please refer to 'Details' for some thought exercises about the nature of 'genes'.*

Description

Collapse bundles of transcripts, discard any with (default) < 1TPM/bundle, and optionally prune any whose joined bundle IDs tend to choke downstream packages for e.g. pathway- or network-based enrichment analysis. Note that this function may or may not be optimal for your RNAseq experiment. Please refer to 'Details' for some thought exercises about the nature of 'genes'.

Usage

```
collapseTpm(kexp, bundleID = "gene_id", minTPM = 0.01,
  discardjoined = TRUE, tx_biotype = NULL, gene_biotype = NULL,
  biotype_class = NULL, ...)
```

Arguments

`kexp`	A KallistoExperiment (or something very much like it)
`bundleID`	The column (in mcols(features(kexp))) of the bundle IDs
`minTPM`	Discard transcripts/bundles with < this many TPMs (0.01)
`discardjoined`	Discard bundles with IDs "joined" by a ";"? (TRUE)
`tx_biotype`	Restrict to a specific mcols(kexp)$tx_biotype? (NULL)
`gene_biotype`	Restrict to a specific mcols(kexp)$gene_biotype? (NULL)
`biotype_class`	Restrict to a specific mcols(kexp)$biotype_class? (No)

Details

This function sums transcripts per million (TPM) of each transcript within bundle of transcripts ("bundle" being a user-defined identifier, often but not always a 'gene', sometimes a biotype or a class of repeat elements).

The default approach is to discard all rows where the maximum TPM is less than the specified cutoff. Since the default cutoff is 1TPM, this means discarding bundles where the total transcripts per million estimate is < 1. (Filtering tends to increase statistical power at a given false-positive rate per Bourgon et al, 2010, http://www.pnas.org/content/107/21/9546.long)

Value

a matrix of TPMs by bundle for each sample

`collapseTranscripts` *Collapse transcripts, discarding any with low reads (pointless to test them) Filtering usually increases statistical power for given false-positive rate; see Bourgon et al, 2010, http://www.pnas.org/content/107/21/9546.long.*

Description

Collapse transcripts, discarding any with low reads (pointless to test them) Filtering usually increases statistical power for given false-positive rate; see Bourgon et al, 2010, http://www.pnas.org/content/107/21/9546.long.

Usage

```
collapseTranscripts(kexp, read.cutoff = 1, ...)
```

Arguments

`kexp`	A KallistoExperiment (or something very much like it)
`read.cutoff`	Discard transcripts and/or bundles w/ < this many reads

Value

a matrix of filtered transcript counts

See Also

collapseBundles

`erccAnalysis` *QC plots of ERCC spike-in controls (FIXME: automate RUVSeq normalization?)*

Description

QC plots of ERCC spike-in controls (FIXME: automate RUVSeq normalization?)

Usage

```
erccAnalysis(kexp, ...)
```

Arguments

`kexp`	something that behaves like a KallistoExperiment

`ERCC_annotated` *ERCC spike-in control annotations*

Description

ERCC spike-in data annotated directly from Life Tech: `URL <- "https://tools.lifetechnologies.com/content/sfs` `ERCC_raw <- read.table(URL, sep="\t", header=T)` `ERCC_raw <- ERCC_raw[ , 3:5]` `names(ERCC_raw) <- c("subgroup","concentration.mix1","concentration.mix2")` `save(ERCC_raw, file="da`

Usage

```
data(ERCC_annotated)
```

Examples

```
 data(ERCC_annotated)
 split(rownames(ERCC_annotated), ERCC_annotated$subgroup)
split(ERCC_annotated$concentration.mix1 / ERCC_annotated$concentration.mix2, ERCC_annotated$subgroup)
```

`extractIndexName` *extract the transcriptome index used for a Kallisto hdf5 file*

Description

extract the transcriptome index used for a Kallisto hdf5 file

Usage

```
extractIndexName(callinfo)
```

Arguments

`callinfo`	the Kallisto call string

Value

the index name

fetchAppSession *fetch app session variables from BaseSpace JSON*

Description

fetch app session variables from BaseSpace JSON

Usage

```
fetchAppSession(jsonFile)
```

Arguments

`jsonFile`	character, the name and/or path to the JSON file

Value

list the appSession created from that JSON file

fetchKallisto *fetch one sample's worth of Kallisto estimates, perhaps with bootstraps*

Description

fetch one sample's worth of Kallisto estimates, perhaps with bootstraps

Usage

```
fetchKallisto(sampleDir = ".", h5file = "abundance.h5",
  collapse = "_mergedWith_", ...)
```

Arguments

`sampleDir`	character string: the path to h5/json files
`h5file`	character string: the file to read
`collapse`	string: collapsing string for indices ("_mergedWith_")

`fitBundles` *encapsulate limma/voom analysis for consistency with ebrowser()*

Description

encapsulate limma/voom analysis for consistency with ebrowser()

Usage

```
fitBundles(kexp, design, bundleID = "gene_id", read.cutoff = 1, ...)
```

Arguments

`kexp`	A KallistoExperiment
`design`	A model matrix
`bundleID`	The ID to bundle on (default is gene_id)
`read.cutoff`	Exclude bundles where the maximum count is < this

Value

A list with elements (design, voomed, fit)

`fitTranscripts` *encapsulate limma/voom analysis and TMM normalization at transcript level FIXME: just farm it out to sleuth, e.g. as seen in ?artemisData::withSleuth*

Description

encapsulate limma/voom analysis and TMM normalization at transcript level FIXME: just farm it out to sleuth, e.g. as seen in ?artemisData::withSleuth

Usage

```
fitTranscripts(kexp, design, read.cutoff = 1, ...)
```

Arguments

`kexp`	A KallistoExperiment
`design`	A model matrix
`read.cutoff`	Exclude transcripts where the maximum count is < this

Value

A list with elements (design, voomed, fit)

`formatResults` *downstream results from fitBundles tabulated into a single object showing genes and quantified limma fitted values.*

Description

example:

Usage

```
formatResults(res)
```

Arguments

`res`	the output from geneWiseAnalysis

Details

```
 library("artemisData")  data("NS", package="artemisData")  formatResults(geneWiseAnalysis(NS, exptD
```

Value

a single merged object with gene names and limma quantified values for differential expression

`geneWiseAnalysis` *Downstream analysis of bundle-aggregated transcript abundance estimates.*

Description

Downstream analysis of bundle-aggregated transcript abundance estimates.

Usage

```
geneWiseAnalysis(kexp, design = NULL, how = c("cpm", "tpm"),
  p.cutoff = 0.05, fold.cutoff = 1, read.cutoff = 1,
  species = c("Homo.sapiens", "Mus.musculus"), fitOnly = FALSE, ...)
```

Arguments

`kexp`	a KallistoExperiment or SummarizedExperiment-like object
`design`	a design matrix with 2nd coefficient as one to test
`p.cutoff`	where to set the p-value cutoff for plots, etc. (0.05)
`fold.cutoff`	where to set the log2-FC cutoff for plots, etc. (1==2x)
`read.cutoff`	minimum read coverage (estimated) for a gene bundle

`species`	which species? (Homo.sapiens; FIX: get from TxDbLite)
`fitOnly`	exit after fitting the EBayes linear model? (FALSE)
`topheat`	how many bundles to include in cluster heatmaps? (100)

Details

If no design matrix is found, the function will look in exptData(kexp)$design. If that too is empty it fails.

Value

a list w/items design, voomed, fit, top, enriched, Figures, scaledExprs, clusts, species, features, ... (perhaps)

`getKallistoVersion` *get the running version of Kallisto in the default path (if there is one)*

Description

get the running version of Kallisto in the default path (if there is one)

Usage

```
getKallistoVersion()
```

Value

a string (the version of Kallisto that was found)

`icgcImport` *import RNAseq data from ICGC (at least the way it comes from their DCC) right now, this means only accepting gene-level summaries; may change later.*

Description

import RNAseq data from ICGC (at least the way it comes from their DCC) right now, this means only accepting gene-level summaries; may change later.

Usage

```
icgcImport(counts = NULL, filename = NULL, filepath = ".", transcriptome,
  level = c("gene", "transcript"), cols = c("submitted_sample_id",
  "gene_id", "raw_read_count"), ...)
```

Arguments

`counts`	A matrix of counts (else provide filename)
`filename`	A filename to pull in (if counts == NULL)
`filepath`	Where the above-specified file resides (".")
`transcriptome`	what transcriptome these were derived from
`level`	At what level shall features be annotated? (gene)
`cols`	Which columns matter in the data? (see function code)
`...`	Other stuff (such as covariates=covs and the like)

Value

KallistoExperiment with derived effective lengths

See Also

CountsAndFeaturesToKallistoExperiment

`indexKallisto` *index transcriptome/transcriptomes*

Description

index transcriptome/transcriptomes

Usage

```
indexKallisto(fastaFiles, fastaPath, fastaTxDbLite = TRUE,
  collapse = "_mergedWith_", kmer = 31, makeUnique = TRUE)
```

Arguments

`fastaFiles`	a character string or vector of FASTA transcriptomes
`fastaPath`	where to find the preceding FASTA files
`fastaTxDbLite`	boolean: should we try to annotate new FASTAs? (yes)
`collapse`	string to name multi-FASTA indices ("_mergedWith_")
`kmer`	integer, integer 3-31 of kmer size,default 31
`makeUnique`	boolean, true will auto-correct existing dupes

`KallistoExperiment` *Initializes a KallistoExperiment and performs some rudimentary checks. Many of the arguments CAN be NULL; determination of which is required is done at run-time. A KallistoExperiment must contain at least the est_counts and eff_length assays, because these are required for tpm estimates. HOWEVER, given raw counts along with informative lengths for each feature (row), we can deduce the effective length of each transcript or bundle from the total normalized count. Function SEtoKE(), which underlies as(SE, "KallistoExperiment"), does exactly that.*

Description

Initializes a KallistoExperiment and performs some rudimentary checks. Many of the arguments CAN be NULL; determination of which is required is done at run-time. A KallistoExperiment must contain at least the est_counts and eff_length assays, because these are required for tpm estimates. HOWEVER, given raw counts along with informative lengths for each feature (row), we can deduce the effective length of each transcript or bundle from the total normalized count. Function SEtoKE(), which underlies as(SE, "KallistoExperiment"), does exactly that.

Usage

```
KallistoExperiment(est_counts = NULL, eff_length = NULL,
  transcriptomes = NULL, covariates = DataFrame(),
  features = GRangesList(), kallistoVersion = "", est_counts_mad = NULL,
  ...)
```

Arguments

`est_counts`	matrix of estimated counts
`eff_length`	matrix of effective transcript lengths
`transcriptomes`	string or strings naming the target txomes
`covariates`	the column metadata (covariates) for each sample
`features`	the row-wise annotations for the object
`kallistoVersion`	version of Kallisto used to run the experiment
`est_counts_mad`	matrix of count MADs summarizing bootstrap runs

See Also

SEtoKE

KallistoExperiment-class *A SummarizedExperiment subclass that stores multiple Kallisto runs*

Description

FIXME: add RUVg-derived ERCC-calibrated normalization factors in metadata

Usage

```
## S4 method for signature 'KallistoExperiment'
counts(object)

## S4 method for signature 'KallistoExperiment'
covariates(object)

## S4 method for signature 'KallistoExperiment'
pData(object)

## S4 replacement method for signature 'KallistoExperiment'
covariates(object) <- value

## S4 replacement method for signature 'KallistoExperiment,DataFrame'
pData(object) <- value

## S4 method for signature 'KallistoExperiment'
features(object)

## S4 replacement method for signature 'KallistoExperiment'
features(object) <- value

## S4 method for signature 'KallistoExperiment'
eff_length(object)

## S4 method for signature 'KallistoExperiment'
tpm(object)

## S4 method for signature 'KallistoExperiment'
kallistoVersion(object)

## S4 method for signature 'KallistoExperiment'
transcriptomes(object)

## S4 method for signature 'KallistoExperiment'
transcriptsBy(x, by = "gene", ...)

## S4 method for signature 'KallistoExperiment'
```

```
mad(x)

## S4 method for signature 'ANY'
covariates(object)

## S4 replacement method for signature 'ANY'
covariates(object) <- value

## S4 method for signature 'SummarizedExperiment0'
features(object)

## S4 replacement method for signature 'RangedSummarizedExperiment'
features(object) <- value

## S4 replacement method for signature 'SummarizedExperiment0'
features(object) <- value
```

Arguments

`object:`	something from which to retrieve covariates
`object:`	something to which covariates should be assigned
`value:`	the covariates to assign (usually a data.frame or DataFrame)
`object:`	something from which features should be obtained
`object:`	something to which features should be assigned
`value:`	the features to assign (usually a GRanges or GRangesList)

Value

the object, perhaps with updated covariates

a GRanges or GRangesList of feature annotations

the object, perhaps with updated feature annotations

Methods (by generic)

- `counts`: Retrieve the estimated count matrix from a KallistoExperiment.
- `covariates`: Retrieve the sample covariates from a KallistoExperiment.
- `pData`: Retrieve the sample covariates from a KallistoExperiment.
- `covariates<-`: Assign the sample covariates for a KallistoExperiment.
- `pData<-`: Convenience method for people used to ExpressionSet, to set per-sample data.
- `features`: Retrieve the per-row annotations for a KallistoExperiment.
- `features<-`: Assign per-row annotations to a KallistoExperiment.
- `eff_length`: Retrieve the matrix of effective transcript lengths from a KallistoExperiment

- tpm: Obtain tpm from the precomputed matrix (computed as shown in https://haroldpimentel.wordpress.com/2014/05/0 the-fpkm-a-review-rna-seq-expression-units/), specifically,
  `rate <- log(counts(object)) - log(eff_length(object));`
  `tpm <- exp(rate - log(sum(exp(rate))) + log(1e6))` @import SummarizedExperiment
- kallistoVersion: Retrieve the version of Kallisto used for alignment from a KallistoExperiment
- transcriptomes: Retrieve the transcriptomes used for annotation from a KallistoExperiment
- transcriptsBy: Fetch transcripts for a gene, or all transcripts bundled by gene.
- mad: Fetch the matrix of MADs for estimated counts, if bootstraps were run.
- covariates:
- covariates<-:
- features:
- features<-:
- features<-:

Slots

transcriptomes	Transcriptomes against which reads were pseudoaligned
kallistoVersion	The version of Kallisto used to pseudoalign the reads

mergeKallisto *Merge multiple Kallisto results, yielding a KallistoExperiment.*

Description

If no transcriptomes are specified (or the ones specified are not found), the resulting KallistoExperiment will have a rowData/rowRanges object which consists entirely of unannotated transcripts from chromosome "Unknown". If transcriptomes are specified and annotations for those transcriptomes are found, the transcripts described in the annotations will be fully annotated for transcript ID, gene ID, gene name, entrez ID, and transcript biotype, provided that these fields are supported in the annotation resources.

Usage

```
mergeKallisto(outputDirs = NULL, outputPath = ".", covariates = NULL,
  annotate = FALSE, collapse = "_mergedWith_", parallel = TRUE, ...)
```

Arguments

outputDirs	character: directories holding Kallisto results (NULL)
outputPath	character: base path to the outputDirs (default is .)
covariates	data.frame or DataFrame: per-sample covariates (NULL)
annotate	boolean: automatically annotate the transcripts? (FALSE)
collapse	string: collapsing string for indices ("_mergedWith_")
parallel	boolean: try to run the merge in parallel? (TRUE)

Details

FIXME: automatically determine which transcriptomes were used (in process!)

`pairFastqFiles` *figure out how a sample's runs can be properly paired (no SE support yet)*

Description

figure out how a sample's runs can be properly paired (no SE support yet)

Usage

```
pairFastqFiles(path = ".", extension = "_*", readPrefix = "R")
```

Arguments

`path`	a character string specifying where the FASTQ files are
`extension`	what is the file extension? default is ".fastq.gz"
`readPrefix`	usually pairs are _R1_/_R2_, so this defaults to "R"

`pathwayPlot` *Use pathview to plot a pathway or pathways colored by gene-wise contrasts. The default is to plot the effect-size-signed, -log10(p.value) for each row.*

Description

Use pathview to plot a pathway or pathways colored by gene-wise contrasts. The default is to plot the effect-size-signed, -log10(p.value) for each row.

Usage

```
pathwayPlot(pathway, kexp = NULL, results = NULL, design = NULL,
  coef = 2, IDtype = "ENSEMBL", path = ".", addData = NULL,
  how = "signed", species = c("Homo sapiens", "Mus musculus"), ...)
```

Arguments

`pathway:` which pathway (or keywords to find the appropriate pathway)

`kexp:` a KallistoExperiment (can be null, if `results` is OK)

`results:` cached results from running fitBundles to avoid rerunning it

`design:` a design matrix to compute contrasts within the experiment

`coef:` which column in the design matrix to extract coefficients (2)

`IDtype:` type of identifier for gene/transcript annotations (ENSEMBL)

`path:` the working directory for downloaded/generated files (".")

`species:` species for pathway extraction (defaults to Homo sapiens)

`addData:` optional per-annotation DNA methylation or copy number data

`how:` how to display effects at each node? (signed -log10(p[j]))

`...:` additional arguments to be passed to pathview

Value

a list with the human-readable pathway name(s) and the plot file(s)

See Also

enrichmentAnalysis

EnrichmentBrowser

pathview

`pcaGGFrame` *plots pca of each kexp assay*

Description

plots pca of each kexp assay

Usage

```
pcaGGFrame(kexpAssays, firstComponent = c("first", "second", "third",
  "fourth", "fifth", "sixth"), secondComponent = c("first", "second", "third",
  "fourth", "fifth", "sixth"), assayInterested = c("cpm", "tpm", "length",
  "mad"))
```

Arguments

`kexpAssays,` kallisto experiment assay

`firstComponent` , the principal component you wish to compare with second

`secondComponent,` the principal component you wish to compare with first component

`assayInterested,` the type of assay data, cpm, etc

Value

a data Frame of the selected principal components used to pass into pcaPlot, input attributes are under attributes(ggFrame) for each PC

---

pcaPlot *plots pca of each kexp assay*

Description

plots pca of each kexp assay

Usage

```
pcaPlot(ggFrame)
```

Arguments

`ggFrame,`	a data frame of desired components to plot

Value

a pca plot of an cpm, tpm, length, or median abs. deviations

---

remapSymbols *update transcript-/gene-level annotations for a KallistoExperiment or matrix*

Description

update transcript-/gene-level annotations for a KallistoExperiment or matrix

Usage

```
remapSymbols(x, what = c("transcript", "gene"))
```

Arguments

`x`	the matrix or kexp
`what`	"transcript" or "gene" level reannotation

Value

x, with updated $gene_name or rownames

`rpkmToTpm` *Converts RPKM or FPKM estimates to TPM. Per Colin Dewey,*

Description

tpm[i, j] == (rpkm[i, j]/(sum(rpkm[i,]))) * 1e6

Usage

```
rpkmToTpm(rpkm, ...)

.tpmBySample(rpkm)
```

Arguments

`rpkm`	a matrix of RPKM or FPKM estimates
`rpkm`	a vector of possibly-zero RPKM estimates

Details

For nonzero rpkm, this becomes

tpm[i, nonzero] == exp(log(rpkm[i, nonzero]) - log(sum(rpkm[i, nonzero])) + log(1e6))

and of course if RPKM == 0 then TPM == 0 as well.

Value

a matrix of TPM estimates

Functions

- `.tpmBySample`:

See Also

http://bioinformatics.oxfordjournals.org/content/26/4/493.full

`runKallisto` *run kallisto with fastq.gz files on a freshly generated or existing index*

Description

run kallisto with fastq.gz files on a freshly generated or existing index

Usage

```
runKallisto(sampleDir, indexName = NULL, fastaPath = ".",
  fastaFiles = NULL, fastqPath = ".", outputPath = ".",
  bootstraps = 100, threads = 1, bias = TRUE, pseudobam = FALSE,
  singleEnd = FALSE, lengthMean = 150, lengthDev = 0.001,
  collapse = "_mergedWith_", extension = ".fastq.gz", ...)
```

Arguments

`sampleDir`	character, subdirectory for sample's FASTQ files
`indexName`	character or NULL, optional name of the index
`fastaPath`	character, where FASTA files are underneath (".")
`fastaFiles`	character vector of FASTA transcriptomes, or NULL
`fastqPath`	character, where sampleDir is located under (".")
`outputPath`	character, output in outputPath/sampleDir (".")
`bootstraps`	integer, how many bootstrap replicates to run? (100)
`threads`	integer, how many threads to use for bootstraps? (4)
`bias`	boolean, perform bias correction? (TRUE)
`pseudobam`	boolean, produce pseudoBAM output? (FALSE)
`singleEnd`	boolean, produce single end quantification, mean and std.dev required
`lengthMean`	integer, length mean used only for single end quantification
`lengthDev`	integer, length std used only for single end quantification
`collapse`	string to name multi-FASTA indices ("_mergedWith_")
`extension`	string, to pass the extension into pairFastqFiles()

saveArtemisPlots *Saving plots in PDF format in a single output file*

Description

Saving plots in PDF format in a single output file

Usage

```
saveArtemisPlots(res, outName)
```

Arguments

`res`	An output from limma created by fitBundles
`outName`	a string for saving the file

Value

returns a pdf in the working directory

SEtoKE *Convert a properly annotated SummarizedExperiment with 'counts', OR a matrix of counts and a GRanges of annotations for each count, to a KallistoExperiment (thereby providing tpm estimates on demand).*

Description

Note that the code here is based upon Harold Pimentel's code, cf. https://haroldpimentel.wordpress.com/2014/05/08/what-the-fpkm-a-review-rna-seq-expression-units/

Usage

```
SEtoKE(SE = NULL, counts = NULL, features = NULL, covariates = NULL,
  transcriptomes = NULL, fraglen = 200, ...)

SummarizedExperimentToKallistoExperiment(SE, transcriptomes)

CountsAndFeaturesToKallistoExperiment(counts, features, transcriptomes, ...)
```

Arguments

`SE`	a properly-annotated SummarizedExperiment
`counts`	if SE is not supplied, a matrix of counts
`features`	if SE is not supplied, GRanges with lengths
`covariates`	optional data.frame or DataFrame if SE is not present
`transcriptomes`	mandatory string or strings naming the transcriptomes
`fraglen`	optional mean fragment length estimate for PE runs
`...`	Other stuff (arguments passed to KallistoExperiment)
`SE`	SummarizedExperiment w/fully annotated rows (features)
`transcriptomes`	mandatory string or strings naming the transcriptomes
`counts`	matrix of transcript or bundle counts
`features`	GRanges of features with valid lengths
`transcriptomes`	mandatory string or strings naming the transcriptomes
`...`	Other stuff (such as covariates=covs and the like)

Details

FIXME: add bias correction to eff_lengths, or derive from TPM instead? FIXME: allow for metadata and per-sample metadata (i.e. MultiAssayExperiment) FIXME: add some sort of KallistoExperiment method to correct bias post-hoc

Value

a KallistoExperiment

a KallistoExperiment

a KallistoExperiment

Functions

- `SummarizedExperimentToKallistoExperiment`:
- `CountsAndFeaturesToKallistoExperiment`:

See Also

KallistoExperiment

SEtoKE

SEtoKE

`strpop` *handy string splitting function that operates on basename(x)*

Description

handy string splitting function that operates on basename(x)

Usage

```
strpop(x, y = " ", z = NULL)
```

Arguments

`x`	a string
`y`	a split character (" ")
`z`	an element or elements to return (the last one)

Value

a string, possibly concatenated from multiple elements

`transcriptWiseAnalysis` *Analysis of raw transcript abundance estimates.*

Description

Analysis of raw transcript abundance estimates.

Usage

```
transcriptWiseAnalysis(kexp, design, p.cutoff = 0.05, fold.cutoff = 1,
  coef = 2, tx_biotype = NULL, gene_biotype = NULL,
  biotype_class = NULL, ...)
```

Arguments

`kexp`	a KallistoExperiment or SummarizedExperiment-like object
`design`	a design matrix w/contrast or coefficient to test in col2
`p.cutoff`	where to set the p-value cutoff for plots, etc. (0.05)
`fold.cutoff`	where to set the log2-FC cutoff for plots, etc. (1 == 2x)
`coef`	which column of the design matrix to test on (2nd)
`tx_biotype`	optionally restrict to one or more tx_biotype classes
`gene_biotype`	optionally restrict to one or more gene_biotype classes
`biotype_class`	optionally restrict to one or more biotype_class ...es

Index