Package 'casper'

November 20, 2025

Version 2.44.0

Date 2025-07-22

Title Characterization of Alternative Splicing based on Paired-End Reads

Author David Rossell, Camille Stephan-Otto, Manuel Kroiss, Miranda Stobbe, Victor Pena

Maintainer David Rossell <rosselldavid@gmail.com>

Depends R (>= 3.6.0), Biobase, IRanges, methods, GenomicRanges

Imports BiocGenerics (>= 0.31.6), coda, EBarrays, gaga, gtools, Seqinfo, GenomicFeatures, limma, mgcv, Rsamtools, rtracklayer, S4Vectors (>= 0.9.25), sqldf, survival, VGAM

Enhances parallel

Description Infer alternative splicing from paired-end RNA-seq data. The model is based on counting paths across exons, rather than pairwise exon connections, and estimates the fragment size and start distributions non-parametrically, which improves estimation precision.

License GPL (>=2)

LazyLoad yes

Collate GenericDefs.R ClassDefinitions.R asymmetryCheck.R calcDenovo.R calcExp.R casperVignettes.R createDenovoGenome.R genePlot.R getDistrs.R getRoc.R denovoExpr.R makeTranscriptDbFromGFF.R mergeBatches.R mergeExp.R modelPrior.R pathCounts.R probNonEquiv.R procBam.R procGenome.R qqnormGenomeWide.R relexprByGene.R rmShortInserts.R simMultSamples.R simPost.R simReads.R splitGenomeByLength.R truncnorm.R wrapKnown.R checks.R wrapDenovo.R

biocViews ImmunoOncology, GeneExpression, DifferentialExpression, Transcription, RNASeq, Sequencing

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

git_branch RELEASE_3_22

git_last_commit 9f5a6ac

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2025-11-20

Contents

annotatedGenome-class *Class "annotatedGenome"*

Description

The annotatedGenome class stores info about transcripts, usually created with procGenome from TxDb objects or user-provided .gtf files.

Objects from the Class

Objects are typically created with a call to procGenome (for known genomes) or to createDenovoGenome (for de novo genomes).

Slots

islands GRangesList object with elements corresponding to gene islands. It indicates the start/end/name of each exon contained in the island

transcripts Each element in the list corresponds to a gene island. It indicates the exons contained in each known variant.

exon2island data.frame indicating the chromosome, start and end of each exon, and its corresponding gene island.

exonsNI GRanges indicating the chromosome, start/end and id of each exon

aliases data.frame indicating the aliases for each known transcript, i.e. transcripts having the exact same sequence of exons.

genomeVersion Character indicating the genome version from which the object was build, e.g. "hg19"

dateCreated Character indicating the date when the object was created. UCSC genomes chance from time to time, so that an "hg19" genome from Jan 2012 may not be exactly the same as in Dec 2012.

denovo Logical variable. FALSE indicates that the object was created using available annotation only. TRUE indicates that new exons/islands were added based on the data observed in a particular RNA-seq experiment.

txLength Numeric vector storing transcript lengths.

knownVars List where each element corresponds to an island, and contains a character vector with names of isoforms that should be considered as known (i.e. always included in the model)

Methods

show signature(object = "annotatedGenome"): Displays general information about the object.

Author(s)

Camille Stephan-Otto Attolini

See Also

procGenome and createDenovoGenome to create annotatedGenome objects.

Examples

```
showClass("annotatedGenome")
```

asymmetryCheck — *Plot asymmetry coefficients for the observed data and compare to those expected under Normality.*

Description

Produces a boxplot for the asymmetry coefficients for each row in the input matrix. Normal observations are simulated using the observed sample means and variances, and their asymmetry coefficients are added to the plot.

Usage

```
asymmetryCheck(x, ...)
```

Arguments

x	ExpressionSet, matrix or data.frame with genes/isoforms in rows
...	Other arguments to be passed on to codeplot

Value

Boxplot with asymmetry coefficients for observed and simulated Normal data

Author(s)

David Rossell

Examples

```
mu <- rnorm(100)
x <- matrix(rnorm(100*5,mu),ncol=5)

asymmetryCheck(x)
```

calcDenovo — *Estimate expression of gene splicing variants de novo.*

Description

calcDenovo estimates expression of gene splicing variants, considering both known variants and variants that have not been previously described.

Usage

```
calcDenovo(distrs, targetGenomeDB, knownGenomeDB=targetGenomeDB, pc,
readLength, islandid, priorq=3, mprior, minpp=0.001, selectBest=FALSE,
searchMethod="submodels", niter, exactMarginal=TRUE,
integrateMethod="plugin", verbose=TRUE, mc.cores=1)
```

Arguments

distrs	List of fragment distributions as generated by the getDistrs function
targetGenomeDB	annotatedGenome object with isoforms we wish to quantify. By default these are the same as in knownGenomeDB, but more typically targetGenomeDB is imported from a .gtf file produced by some isoform prediction software.
knownGenomeDB	annotatedGenome object with known isoforms, e.g. from UCSC or GENCODE annotations. Used to set the prior probability that any given isoform is expressed. knownGenome should be the same genome annotations used to create argument mprior (when provided)
pc	Named vector of exon path counts as returned by pathCounts
readLength	Read length in bp, e.g. in a paired-end experiment where 75bp are sequenced on each end one would set readLength=75.
islandid	Name of the gene island to be analyzed. If not specified, all gene islands are analyzed.
priorq	Parameter of the prior distribution on the proportion of reads coming from each variant. The prior is Dirichlet with prior sample size for each variant equal to priorq. We recommend priorq=3 as this defines a non-local prior that penalizes falsely predicted isoforms that show low expression.
mprior	Prior on the model space returned by modelPrior, used to favor isoforms consistent with knownGenomeDB. If left missing it is estimated from knownGenomeDB. See details.
minpp	Models (i.e. splicing configurations) with posterior probability less than minpp are not reported. This argument can help reduce substantially the amount of required memory to store the results.
selectBest	If set to TRUE only the model with highest posterior probability is reported. While this can save memory, we do not recommend this option as it may ignore a substantial amount of uncertainty.
searchMethod	Method used to perform the model search. "allmodels" enumerates all possible models (warning: this is not feasible for genes with >5 exons). "rwmcmc" uses a random-walk MCMC scheme to focus on models with high posterior probability. "submodels" considers that some isoforms in targetGenomeDB may not be expressed, but does not search for new variants. "auto" uses "allmodels" for genes with up to 5 exons and "rwmcmc" for longer genes. See details.
niter	Number of MCMC iterations.
exactMarginal	Set to FALSE to estimate posterior model probabilities as the proportion of MCMC visits. Set to TRUE to use the integrated likelihoods (default). See details.
integrateMethod	Method to compute integrated likelihoods. The default ('plugin') evaluates likelihood*prior at the posterior mode and is the faster option. Set 'Laplace' for Laplace approximations and 'IS' for Importance Sampling. The latter increases computation cost very substantially.
verbose	Set to TRUE to display progress information.
mc.cores	Number of processors to be used for parallel computation. Can only be used if the package multicore is available for your system. Warning: using multiple processors substantially increases the memory requirements, so set this value carefully.

Details

calcDenovo explores which subset of the isoforms indicated in targetGenomeDB are truly expressed. It also adds new isoforms when some reads follow an exon path that is not possible under any of the isoforms in targetGenomeDB. calcDenovo the posterior probability of each model (i.e. configuration of expressed variants) via Bayes theorem.

P(model|y) "proportional to" m(y|model) P(model)

where m(y|model) is the integrated likelihood and P(model) is the prior probability of the model. For example, a gene with 20 predicted isoforms in targetGenome gives rise 2^20 - 1 possible models (configurations of expressed isoforms).

Importantly, P(model) can be set by analyzing available genome annotations in knownGenomeDB. For instance, genes with 20 exons have isoforms that tend to use most of the 20 exons. They also tend to express more isoforms than genes with 5 exons. The function modelPrior analyzes knownGenomeDB to set reasonable values for P(model).

An exhaustive enumeration of all possible models is not feasible unless the gene is very short (e.g. around 5 exons). For longer genes we use computational strategies to search a subset of "interesting" models. This is controlled by the argument searchMethod (see above).

In order to compute P(model|y) one can either use the computed m(y|model) P(model) (option exactMarginal==TRUE) or the proportion of MCMC visits (option exactMarginal==FALSE). Unless niter is large the former option typically provides more precise estimates.

Value

A denovoGenomeExpr object.

Author(s)

Camille Stephan-Otto Attolini, Manuel Kroiss, David Rossell

References

Rossell D, Stephan-Otto Attolini C, Kroiss M, Stocker A. Quantifying Alternative Splicing from Paired-End RNA-sequencing data. Annals of Applied Statistics, 8(1):309-330.

See Also

denovoExpr to obtain expression estimates from the calcDenovo output. plotExpr to produce a plot with splicing variants and estimated expression.

Examples

```
## See help(denovoExpr)
```

calcExp *Estimate expression of a known set of gene splicing variants.*

Description

Estimate expression of gene splicing variants, assuming that the set of variants is known. When rpkm is set to TRUE, fragments per kilobase per million are returned. Otherwise relative expression estimates are returned.

Usage

```
calcExp(distrs, genomeDB, pc, readLength, islandid, rpkm=TRUE, priorq=2,
priorqGeneExpr=2, citype="none", niter=10^3, burnin=100, mc.cores=1, verbose=FALSE)
```

Arguments

distrs	List of fragment distributions as generated by the getDistrs function
genomeDB	knownGenome object containing annotated genome, as returned by the procGenome function.
pc	Named vector of exon path counts as returned by pathCounts
readLength	Read length in bp, e.g. in a paired-end experiment where 75bp are sequenced on each end one would set readLength=75.
islandid	Name of the gene island to be analyzed. If not specified, all gene islands are analyzed.
rpkm	Set to FALSE to return relative expression levels, i.e. the proportion of reads generated from each variant per gene. These proportions add up to 1 for each gene. Set to TRUE to return fragments per kilobase per million (RPKM).
priorq	Parameter of the prior distribution on the proportion of reads coming from each variant. The prior is Dirichlet with prior sample size for each variant equal to priorq. We recommend priorq=2 for estimation, as it pools the estimated expression away from 0 and 1 and returned lower estimation errors than priorq=1 in our simulated experiments.
priorqGeneExpr	Parameter for prior distribution on overall gene expression. Defaults to 2, which ensures non-zero estimates for all genes
citype	Set to "none" to return no credibility intervals. Set to "asymp" to return approximate 95% CIs (obtained via the delta method). Set to "exact" to obtain exact CIs via Monte Carlo simulation. Options "asymp" and especially "exact" can increase the computation time substantially.
niter	Number of Monte Carlo iterations. Only used when citype=="exact".
burnin	Number of burnin Monte Carlo iterations. Only used when citype=="exact".
mc.cores	Number of processors to be used for parallel computation. Can only be used if the package multicore is available for your system.
verbose	Set to TRUE to display progress information.

Value

Expression set with expression estimates. featureNames identify each transcript via RefSeq ids, and the featureData contains further information. If citype was set to a value other than "none", the featureData also contains the 95% credibility intervals (i.e. intervals that contain the true parameter value with 95% posterior probability).

Author(s)

Camille Stephan-Otto Attolini, Manuel Kroiss, David Rossell

References

Rossell D, Stephan-Otto Attolini C, Kroiss M, Stocker A. Quantifying Alternative Splicing from Paired-End RNA-sequencing data. Annals of Applied Statistics, 8(1):309-330.

See Also

relexprByGene

Examples

```
data(K562.r1l1)
data(hg19DB)

#Pre-process
bam0 <- rmShortInserts(K562.r1l1, isizeMin=100)
pbam0 <- procBam(bam0)
head(getReads(pbam0))

#Estimate distributions, get path counts
distrs <- getDistrs(hg19DB,bam=bam0,readLength=75)
pc <- pathCounts(pbam0, DB=hg19DB)

#Get estimates
eset <- calcExp(distrs=distrs, genomeDB=hg19DB, pc=pc, readLength=75, rpkm=FALSE)
head(exprs(eset))
head(fData(eset))

#Re-normalize relative expression to add up to 1 within gene_id rather
# than island_id
eset <- relexprByGene(eset)

#Add fake sample by permuting and combine
eset2 <- eset[sample(1:nrow(eset),replace=FALSE),]
sampleNames(eset2) <- '2' #must have a different name
esetall <- mergeExp(eset,eset2)

#After merge samples are correctly matched
head(exprs(esetall))
head(fData(esetall))
```

---

denovoExpr *Estimate expression for de novo splicing variants.*

Description

Obtains expression estimates from denovoGenomeExpr objects, as returned by calcDenovo. When rpkm is set to TRUE, fragments per kilobase per million are returned. Otherwise relative expression estimates are returned.

The estimates can be obtained by Bayesian model averaging (default) or by selecting the model with highest posterior probability. See details.

Usage

```
denovoExpr(x, pc, rpkm = TRUE, summarize = "modelAvg", minProbExpr = 0.5, minExpr = 0.05)
```

Arguments

x	denovoGenomeExpr object returned by calcExp
pc	Named vector of exon path counts as returned by pathCounts
rpkm	Set to FALSE to return relative expression levels, i.e. the proportion of reads generated from each variant per gene. These proportions add up to 1 for each gene. Set to TRUE to return fragments per kilobase per million (RPKM).
summarize	Set to "modelAvg" to obtain model averaging estimates, or to "bestModel" to select the model with highest posterior probability. We recommend the former, as even the best model may have low posterior probability.
minProbExpr	Variants with (marginal posterior) probability of being expressed below minProbExpr are omitted from the results. This argument is useful to eliminate variants that are not at least moderately supported by the data.
minExpr	Variants with relative expression minExpr are omitted from the results. This is useful to eliminate variants to which few reads are assigned, e.g. due to read miss-alignments or biases.

Value

Expression set with expression estimates. The featureData indicates the gene island id, posterior probability that each variant is expressed (column "probExpressed") and the number of aligned reads per gene island (column "explCnts").

Author(s)

David Rossell

References

Rossell D, Stephan-Otto Attolini C, Kroiss M, Stocker A. Quantifying Alternative Splicing from Paired-End RNA-sequencing data. Annals of Applied Statistics, 8(1):309-330.

Examples

```
## NOTE: toy example with few reads & genes to illustrate code usage
##       Results with complete data are much more interesting!

data(K562.r1l1)
data(hg19DB)

#Pre-process
bam0 <- rmShortInserts(K562.r1l1, isizeMin=100)
pbam0 <- procBam(bam0)

#Estimate distributions, get path counts
distrs <- getDistrs(hg19DB,bam=bam0,readLength=75)
pc <- pathCounts(pbam0, DB=hg19DB)

#Set prior distrib on model space
mprior <- modelPrior(hg19DB, maxExons=40, smooth=FALSE)
```

```
#Fit model
denovo <- calcDenovo(distrs,targetGenomeDB=hg19DB,pc=pc,readLength=75,priorq=3,mprior=mprior,minpp=0)

head(names(denovo))
denovo[['6499']]
posprob(denovo[['6499']])

#Get estimates
eset <- denovoExpr(denovo, pc=pc, rpkm=TRUE, minProbExpr=0.5)

head(exprs(eset))
head(fData(eset))
```

`denovoGeneExpr-class` *Class "denovoGeneExpr"*

Description

`denovoGeneExpr` stores inferred expression for de novo splicing variants for a single gene. `denovoGenomeExpr` stores the information for several genes (typically, the whole genome).

Objects from the Class

Objects are returned by `calcDenovo`. When running `calcDenovo` on multiple genes results are returned in a `denovoGenomeExpr` object. Results for a single gene can be retrieved using the [[operator as usual, which returns a `denovoGeneExpr` object.

Slots

posprob `data.frame` containing the posterior probability of each model

expression `data.frame` with the estimated expression of each variant under each model

variants `matrix` indicating the exons contained in each variant.

integralSum Sum of the log(integrated likelihood) + log(model prior probability) across all considered models.

npathDeleted Number of paths that had 0 probability under all considered variants and had to be excluded for model fitting purposes.

priorq Input parameter to `calcDenovo`

txLength Length of transcripts in bp (including new isoforms found by casper)

Methods

show `signature(object = "denovoGeneExpr")`: Displays general information about the object.

names Show names (island ids)

"[" Selects a subset of genes

"[[" Selects a single gene

posprob Accesses the posterior probabilities of each model (slot posprob)

variants Accesses the variant names and their respective exons

variants<- Replaces the value of the slot variants (can be useful for renaming variants, for instance)

Author(s)

David Rossell

See Also

`calcDenovo` to create objects from the class. `denovoExpr` to obtain expression estimates from `denovoGenomeExpr` objects.

Examples

```
showClass("denovoGeneExpr")
```

---

denovoGenomeExpr-class

Class "denovoGenomeExpr"

Description

`denovoGeneExpr` stores inferred expression for de novo splicing variants for a single gene. `denovoGenomeExpr` stores the information for several genes (typically, the whole genome).

Objects from the Class

Objects are returned by `calcDenovo`.

Slots

islands A list of `denovoGeneExpr` objects, with each element containing results for an individual gene.

Methods

show `signature(object = "denovoGenomeExpr")`: Displays general information about the object.

as.list Coerces the object to a list

"[" Selects a subset of genes

"[[" Selects a single gene

Author(s)

Camille Stephan-Otto Attolini

See Also

`procGenome` and `createDenovoGenome` to create `denovoGenomeExpr` objects.

Examples

```
showClass("denovoGeneExpr")
showClass("denovoGenomeExpr")
```

distrsGSE37704 *Estimated read start and insert size distributions from MiSeq data in GEO dataset GSE37704.*

Description

We downloaded the fastq files, aligned with TopHat and processed with `wrapKnown` to obtain the estimated distributions for each of the 6 samples. `distrsGSE37704` is a list with the 6 corresponding elements. The estimated distributions for HiSeq data were very similar, hence these distributions can be used as defaults for Illumina MiSeq and HiSeq experiments.

Usage

```
data(distrsGSE37704)
```

Format

An `list` with 6 elements of class `readDistrs`. See `help(getDistrs)` and `help(readDistrs-class)` for details.

Examples

```
data(distrsGSE37704)
distrsGSE37704
plot(distrsGSE37704[[1]],'readSt')
lines(distrsGSE37704[[2]], 'readSt', col=2)
plot(distrsGSE37704[[1]],'fragLength')
```

genePlot *Plot exon structure for each transcript of a given gene.*

Description

Plot exon structure for each transcript of a given gene. Optionally, aligned reads can be added to the plot.

Usage

```
genePlot(generanges, islandid, genomeDB, reads, exp, names.arg, xlab='',
ylab='', xlim, cex=1, yaxt='n', col, ...)
```

Arguments

`generanges`	Object containing the ranges with start/end of each exon.
`islandid`	If `generanges` is not specified, transcripts are obtained from island `islandid` from the annotated genome `genomeDB`.
`genomeDB`	Annotated genome produced with the "procGenome" function
`reads`	`pbam` object with aligned reads. This is an optional argument.

exp	ExpressionSet object with expression values, as returned by calcExp. This is an optional argument.
names.arg	Optionally, indicate the names of each transcript.
xlab	x-axis label
ylab	y-axis label
xlim	x-axis limits, defaults to start of 1st exon and end of last exon
cex	Character expansion
yaxt	The y-axis in the plot has no interpretation, hence by default it is not displayed.
col	Either single color or vector of colors to be used to draw each transcript. Defaults to rainbow colors.
...	Other arguments to be passed on to plot.

Value

A plot is produced.

Methods

signature(generanges="CompressedIRangesList", islandid="ANY", genomeDB="ANY", reads="ANY", exp="AN
: Plots a set of transcripts. Each element in the generanges corresponds to a transcript. Each transcript should contain exon start/end positions.

signature(generanges="IRanges", islandid="ANY", genomeDB="ANY", reads="ANY", exp="ANY")
: Plots a single transcript. Each range indicates the start/end of a single exon.

signature(generanges="IRangesList", islandid="ANY", genomeDB="ANY", reads="ANY", exp="ANY")
: Plots a set of transcripts. Each element in the generanges corresponds to a transcript. Each transcript should contain exon start/end positions.

signature(generanges="GRangesList", islandid="ANY", genomeDB="ANY", reads="ANY", exp="ANY")
: Plots a set of transcripts. Each element in the generanges corresponds to a transcript. Each transcript should contain exon start/end positions.

signature(generanges="GRanges", islandid="ANY", genomeDB="ANY", reads="ANY", exp="ANY")
: Plots a set of transcripts. Each space in generanges corresponds to a transcript. Each transcript should contain exon start/end positions.

signature(generanges="missing", islandid="character", genomeDB="annotatedGenome", reads="GRanges"
: Plots all transcripts stored in genomeDB for island with identifier islandid. Individual reads are added to the plot (reads contains start/end of individual read fragments).

signature(generanges="missing", islandid="character", genomeDB="annotatedGenome", reads="missing"
: Plots all transcripts stored in genomeDB for island with identifier islandid.

signature(generanges="missing", islandid="character", genomeDB="annotatedGenome", reads="procBam"
: Plots all transcripts stored in genomeDB for island with identifier islandid. Individual reads are added to the plot (reads contains start/end of individual read fragments).

signature(generanges="missing", islandid="character", genomeDB="annotatedGenome", reads="procBam"
: Plots all transcripts stored in genomeDB for island with identifier islandid. Individual reads and estimated expression are added to the plot (reads contains start/end of individual read fragments).

Author(s)

Camille Stephan-Otto Attolini, David Rossell

Examples

```
data(hg19DB)

#Plot an IRangesList
txs <- transcripts(txid="NM_005158",genomeDB=hg19DB)
genePlot(txs)

#Equivalently, indicate islandid
islandid <- getIsland(txid="NM_005158",genomeDB=hg19DB)
genePlot(islandid=islandid, genomeDB=hg19DB)
```

---

getDistrs *Compute fragment start and fragment length distributions*

Description

Compute fragment start distributions by using reads aligned to genes with only one annotated variant. Estimate fragment length distribution using fragments aligned to long exons (>1000nt). Fragment length is defined as the distance between the start of the left-end read and the end of the right-end read.

Usage

```
getDistrs(DB, bam, pbam, islandid=NULL, verbose=FALSE, nreads=4*10^6,
readLength, min.gt.freq = NULL, tgroups=5, mc.cores=1)
```

Arguments

DB	Annotated genome. Object of class knownGenome as returned by procGenome.
bam	Aligned reads, as returned by scanBam. It must be a list with elements 'qname', 'rname', 'pos' and 'mpos'. Ignored when argument pbam is specified.
pbam	Processed BAM object of class procBam, as returned by function procBam. Arguments bam and readLength are ignored when pbam is specified.
islandid	Island IDs of islands to be used in the read start distribution calculations (defaults to genes with only one annotated variant)
verbose	Set to TRUE to print progress information.
nreads	To speed up computations, only the first nreads are used to obtain the estimates. The default value of 4 milions usually gives highly precise estimates.
readLength	Read length in bp, e.g. in a paired-end experiment where 75bp are sequenced on each end one would set readLength=75.
min.gt.freq	The target distributions cannot be estimated with precision for gene types that are very unfrequent. Gene types with relative frequency below min.gt.freq are merged, e.g. min.gt.freq=0.05 means gene types making up for 5% of the genes in DB will be combined and a single read start and length distribution will be estimated for all of them.
tgroups	As an alternative to min.gt.freq you may specify the maximum number of distinct gene types to consider. A separate estimate will be obtained for the tgroups with highest frequency, all others will be combined.
mc.cores	Number of cores to use for parallel processing

Value

An object of class readDistrs with slots:

lenDis	Table with number of fragments with a given length
stDis	Cumulative distribution function (object of type closure) for relative start position

Author(s)

Camille Stephan-Otto Attolini, David Rossell

Examples

```
data(K562.r1l1)
data(hg19DB)
bam0 <- rmShortInserts(K562.r1l1, isizeMin=100)

distrs <- getDistrs(hg19DB,bam=bam0,readLength=75)

#Fragment length distribution
plot(distrs,'fragLength')

#Fragment start distribution (relative to transcript length)
plot(distrs,'readSt')
```

---

getIsland *getIsland returns the island id associated to a given entrez or transcript id in an annotatedGenome object. getChr indicates the chromosome for a given Entrez, transcript or island id.*

Description

annotatedGenome objects store information regarding genes and transcripts. When there's an overlap in exons between several genes, these genes are grouped into gene islands. getIsland retrieves the island to which each gene or transcript was assigned, while getChr indicates the chromosome.

Usage

```
getIsland(entrezid, txid, genomeDB)
getChr(entrezid, txid, islandid, genomeDB)
```

Arguments

entrezid	Character indicating single Entrez identifier. Can be left missing and specify another identifier instead.
txid	Character indicating a single RefSeq transcript identifier. Can be left missing and specify another identifier instead.
islandid	Character indicating the gene island indentifier. Can be left missing and specify another identifier instead.
genomeDB	Object of class annotatedGenome

Value

Character with island identifier

Methods

`signature(entrezid='character',txid='missing',genomeDB='annotatedGenome')` Return island id for given Entrez identifier

`signature(entrezid='missing',txid='character',genomeDB='annotatedGenome')` Return island id for given transcript identifier (RefSeq)

`signature(entrezid='character',txid='missing',islandid='missing',genomeDB='annotatedGenome')` Return chromosome for given Entrez identifier (RefSeq)

`signature(entrezid='missing',txid='character',islandid='missing',genomeDB='annotatedGenome')` Return chromosome for given transcript identifier (RefSeq)

`signature(entrezid='missing',txid='missing',islandid='character',genomeDB='annotatedGenome')` Return chromosome for given island identifier

`signature(entrezid='character',txid='missing',islandid='missing')` Return chromosome for given Entrez identifier

`signature(entrezid='missing',txid='character',islandid='missing')` Return chromosome for given transcript identifier (RefSeq)

`signature(entrezid='missing',txid='character',islandid='missing')` Return chromosome for given island identifier

Examples

```
data(hg19DB)
getIsland(entrezid="27",genomeDB=hg19DB)
getIsland(txid="NM_005158",genomeDB=hg19DB)

getChr(entrezid="27",genomeDB=hg19DB)
getChr(txid="NM_005158",genomeDB=hg19DB)
```

---

getNreads *Get total number of paths in each island from a pathCounts object.*

Description

getNreads returns a numeric vector with the total number of path counts in each island from a pathCounts object.

Usage

```
getNreads(pc)
```

Arguments

pc	pathCounts object generated by pathCounts()

Value

Numeric vector with total number of path counts in each island of pc.

Methods

`signature(pathCounts='pathCounts')` Returns numeric vector with total number of path counts for each island in the pathCounts object.

Author(s)

Camille Stephan-Otto Attolini

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##-- or do  help(data=index)  for the standard data sets.
```

---

getReads *getReads returns the reads stored in a* procBam *object.*

Description

`procBam` objects store reads that have been split according to their CIGAR codes. `getReads` accesses these reads.

Usage

```
getReads(x)
```

Arguments

`x`	Object of class `procBam`

Value

`RangedData` object with reads stored in `x`.

Methods

`signature(x='procBam')` Return reads stored in `x`.

Examples

```
#See example in calcExp
```

getRoc *Operating characteristics of differential expression analysis*

Description

getRoc compares simulation truth and data analysis results to determine False Positives (FP), False Negatives (FP), True Positives (TP), True Negatives (TN), Positives (FP+TP), False Discovery Proportion (FP/P) and Power (TP/(TP+FN)).

Usage

```
getRoc(simTruth, decision)
```

Arguments

simTruth	Binary vector or matrix indicating simulation truth (FALSE or 0 for non differential expression, TRUE or 1 for differential expression)
decision	Binary vector or matrix with differential expression calls based on some data analysis.

Value

data.frame with TP, FP, TN, FN, P, FDR and Power.

Methods

signature(simTruth='logical',decision='logical') Operating characteristics are computed for a single simulation

signature(simTruth='numeric',decision='numeric') Operating characteristics are computed for a single simulation

signature(simTruth='matrix',decision='matrix') simTruth and decision contain truth and calls for several simulations (in columns). getRoc returns a data.frame with operating characteristics in each simulation.

Author(s)

David Rossell

Examples

```
## See help(probNonEquiv) for an example
```

hg19DB *Subset of human genome (UCSC hg19 version)*

Description

We downloaded the human genome hg19 via procGenome and selected a few genes from chromosome 1 to use as a toy data for the vignette and examples.

Usage

```
data(hg19DB)
```

Format

An annotatedGenome object. See help(procGenome) and help(annotatedGenome-class) for details.

Examples

```
data(hg19DB)
hg19DB
slotNames(hg19DB)
```

K562.r1l1 *Toy RNA-seq data from RGASP project.*

Description

The paired-end RNA-seq data is from the RGASP project sample K562_2x75 (replicate 1, lane 1) and was obtained at ftp://ftp.sanger.ac.uk/pub/gencode/rgasp/RGASP1/inputdata/human_fastq. Reads were aligned against hg19 with tophat 2.0.2 and bowtie 0.12.5, setting the insert size at -r 200, and imported into R using scanBam from package Rsamtools. For illustration purposes, we selected reads mapping to a few genes only (namely, the genes that were also selected for the toy genome annotation in data(hg19DB).

Usage

```
data(K562.r1l1)
```

Format

A list indicating read id, chromosome, start and end locations and the position of the pair, as returned by scanBam.

Source

ftp://ftp.sanger.ac.uk/pub/gencode/rgasp/RGASP1/inputdata/human_fastq

References

C Trapnell, L Pachter, SL Salzberg. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics, 2009, 25, 1105-1111. doi=10.1093/bioinformatics/btp120.

B Langmead, C Trapnell, M Pop, SL Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology, 2009, 10:R25.

Examples

```
data(K562.r1l1)
names(K562.r1l1)
```

`mergeBatches` *Merge two ExpressionSet objects by doing quantile normalization and computing partial residuals (i.e. substracting group mean expression in each batch). As currently implemented the method is only valid for balanced designs, e.g. each batch has the same number of samples per group.*

Description

`mergeBatches` combines `x` and `y` into an `ExpressionSet`, performs quantile normalization and adjusts for batch effects by subtracting the mean expression in each batch (and then adding the grand mean so that the mean expression per gene is unaltered).

Usage

```
mergeBatches(x, y, mc.cores=1)
```

Arguments

`x`	`ExpressionSet` object with data from batch 1.
`y`	Either `ExpressionSet` object with data from batch 2, or `simulatedSamples` object with data from multiple simulations.
`mc.cores`	Number of processors to be used (ignored when `y` is an `ExpressionSet`)

Value

When `y` is an `ExpressionSet`, `mergeBatches` returns an `ExpressionSet` with combined expressions. Its `featureData` contains a variable `"batch"` indicating the batch that each sample corresponded to.

When `y` is a `simulatedSamples` object, `mergeBatches` is applied to combine `x` with each dataset in `y` and a list of `ExpressionSet` objects is returned.

Author(s)

David Rossell

Examples

```
#Fake data from 2 batches
x <- matrix(rnorm(6),nrow=2)
colnames(x) <- paste('x',1:3,sep='')
y <- matrix(1+rnorm(6),nrow=2)
colnames(y) <- paste('y',1:3,sep='')
x <- new("ExpressionSet",exprs=x)
y <- new("ExpressionSet",exprs=y)
exprs(x)
exprs(y)

#Merge & adjust
z <- mergeBatches(x,y)
exprs(z)
```

`mergeExp` *Merge splicing variant expression from multiple samples*

Description

`mergeExp` combines the output of `calcExp` from multiple samples, i.e. multiple `ExpressionSet` objects, into a single `ExpressionSet`

Usage

```
mergeExp(..., sampleNames,  keep=c('transcript','gene_id','island_id'))
```

Arguments

`...`	`ExpressionSet` objects to be combined.
`sampleNames`	Character vector indicating the name of each sample. Defaults to 'Sample1', 'Sample2', etc.
`keep`	Variables in the featureData of each individual `ExpressionSet` to keep in the merged output.

Details

`mergeExp` runs some checks to ensure that object can be combined (e.g. making sure that measurements are obtained on same set of genes), then sorts and formats each input `ExpressionSet`.

A label with the sample name is appended to variables in the featureData that appear in multiple samples, e.g. variable 'se' reporting standard errors (obtained by setting citype='asymp' in calcExp).

Value

Object of class `ExpressionSet` combining the input ExpressionSets. Its `featureData` contains the columns indicated in the keep argument, plus a column `readCount` with the total number of reads mapped to each gene (or gene island, when multiple genes have overlapping exons).

Author(s)

David Rossell

See Also

calcExp to obtain an ExpressionSet for an individual sample.

Examples

```
#See example in calcExp
```

---

modelPrior *Set prior distribution on expressed splicing variants.*

Description

Set prior on expressed splicing variants using the genome annotation contained in a knownGenome object.

The prior probability of variants V1,...,Vn being expressed depends on n, on the number of exons in each variant V1,...,Vn and the number of exons in the gene. See the details section.

Usage

```
modelPrior(genomeDB, maxExons=40, smooth=TRUE, verbose=TRUE)
```

Arguments

genomeDB	Object of class knownGenome
maxExons	The prior distribution is estimated for genes with 1 up to maxExons exons. As there are fewer genes with many exons, the prior parameters are estimated poorly. To avoid this common estimate is used for all genes with more than maxExons exons
smooth	If set to TRUE the estimated prior distribution parameters for the number of exons in a gene are smoothed using Generalized Additive Models. This step typically improves the precision of the estimates, and is only applied to genes with 10 or more exons.
verbose	Set to TRUE to print progress information.

Details

The goal is to set a prior that takes into account the number of annotated variants for genes with E exons, as well as the number of exons in each variant.

Suppose we have a gene with E exons. Let V_1,...,V_n be n variants of interest and let |V_1|,...,|V_n| be the corresponding number of exons in each variant. The prior probability of variants V_1,...,V_n being expressed is modeled as

P(V_1,...,V_n|E)= P(n|E) P(|V_1| |E) ... P(|V_n| |E)

where P(n|E)= NegBinom(n; k_E, r_E) I(0 < n < 2^E) and P(|V_i| |E)= BetaBinomial(|V_i|-1; E-1, alpha_E, beta_E).

The parameters k_E, r_E, alpha_E, beta_E depend on E (the number of exons in the gene) and are estimated from the available annotation via maximum likelihood. Parameters are estimated jointly for all genes with E>= maxExons in order to improve the precision.

For smooth==TRUE, alpha_E and beta_E are modeled as a smooth function of E by calling gam and setting the smoothing parameter via cross-validation. Estimates for genes with E>=10 are substituted by their smooth versions, which typically helps improve stability in the estimates.

Value

List with 2 components.

nvarPrior	List with prior distribution on the number of expressed variants for genes with 1,2,3... exons. Each element contains the truncated Negative Binomial parameters, observed and predicted frequencies (counting the number of genes with a given number of variants).
nexonPrior	List with prior distribution on the number of exons in a variant for genes with 1,2,3... exons. Each element contains the Beta-Binomial parameters, observed and predicted frequencies (counting the number of variants with a given number of exons)

Author(s)

David Rossell, Camille Stephan-Otto Attolini

Examples

```
data(hg19DB)
mprior <- modelPrior(hg19DB, maxExons=10)

##Prior on number of expressed variants
##Genes with 2 exons
##mprior$nvarPrior[['2']]
##Genes with 3 exons
##mprior$nvarPrior[['3']]

##Prior on the number of exons in an expressed variant
##Genes with 2 exons
##mprior$nexonPrior[['2']]
##Genes with 3 exons
##mprior$nexonPrior[['3']]
```

---

modelPriorAS-class *Class "modelPriorAS"*

Description

modelPriorAS stores parameters for the prior distribution on all possible alternative splicing configuration (i.e. prior on model space). This information is used for de novo reconstruction of splicing variants.

Objects from the Class

Objects are created by function modelPrior.

Slots

nvarPrior Prior on the number of variants per gene. A list with components "nbpar" containing the parameters of the Negative Binomial distribution, "obs" containing the observed counts and "pred" the Negative Binomial predicted counts.

nexonPrior Prior on the number of exons in an expressed variant. A list with components "bbpar" containing Beta-Binomial parameters, "obs" containing the observed counts and "pred" the Beta-Binomial predicted counts.

Methods

show `signature(object = "modelPriorAS")`: Displays general information about the object.

"[" Selects prior parameters for genes with the specified number of exons

coef Selects a single gene

Author(s)

David Rossell

See Also

`procGenome` and `createDenovoGenome` to create `modelPriorAS` objects.

Examples

```
showClass("modelPriorAS")
```

---

pathCounts *Compute exon path counts*

Description

Compute counts for exon paths visited by aligned reads

Usage

```
pathCounts(reads, DB, mc.cores = 1, verbose=FALSE)
```

Arguments

`reads`	Object of class `procBam` containing aligned reads, as returned by `procBam`.
`DB`	Object of class `annotatedGenome` containing either a known or de novo annotated genome.
`mc.cores`	Number of processors to be used for parallel computing. Requires having package `multicore` installed and loaded.
`verbose`	Set to `TRUE` to print progress information.

Value

Named integer vector with counts of exon paths. Names are character strings built as ".exon1.exon2-exon3.exon4.", with dashes making the split between exons visited by left and right-end reads correspondingly.

Methods

`signature(reads='list')` Computes counts for exon paths from a list of procBam objects (usually reads processed and split by chromosome).

`signature(reads='procBam')` Compute counts for exon paths from a procBam object of processed reads.

Author(s)

Camille Stephan-Otto Attolini

See Also

procGenome to create an annotated genome object, createDenovoGenome to create a de novo annotated genome. See help(getNreads) to get number of fragments mapping to each island.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##-- or do  help(data=index)  for the standard data sets.
```

---

pathCounts-class *Class "pathCounts"*

Description

Stores exon path counts.

Objects from the Class

Objects are created with a call to pathCounts.

Slots

counts List with one element per gene island. For each island, it contains a named vector with exon path counts. The names indicate the visited exons.

For instance, consider that for gene '1' with 2 exons we observe 10 reads in which the left end falls completely in exon 1 and the right end in exon 2. Suppose that for 5 reads the left end bridges exons 1-2 and the right end falls in exon 2. Then pc[['1']] would contain c(10,5) and names(pc[['1']]) would contain c(".1-2.",".1.2-2.")

denovo Logical variable. FALSE indicates that the counts correspond to a known genome (i.e. created with procGenome), and TRUE to a de novo annotated genome (i.e. created with createDenovoGenome).

stranded Logical variable. TRUE indicates that the path counts were obtained from and RNA-seq experiment where strand information was preserved.

Methods

show signature(object = "pathCounts"): Displays general information about the object.

Author(s)

Camille Stephan-Otto Attolini

Examples

```
showClass("pathCounts")
```

plot-methods *Plot estimated read start and fragment length distributions.*

Description

Plots the estimated fragment length (insert size) distribution and the relative read start distribution (0 indicating transcription start, 1 transcription end). The former checks that the insert size distribution matches that described in the experimental protocol. The latter checks the extent to which reads are non-uniformly distributed (note: casper does NOT assume reads to be uniformly distributed, so a lack of uniformity is not a problem per se).

Arguments

x	Object of type readDistrs, as returned by getDistrs.
y	Set to "fragLength" to plot the estimated insert size ditribution. Set to "readSt" to plot a histogram of the estimated read start distribution.
...	Further arguments to be passed on to plot.

Methods

signature(x = "readDistrs", y = "ANY") x is an object of type readDistrs, as returned by getDistrs. The plot allows to visualize the fragment length and read start distributions in a given sample.

signature(x = "readDistrs") x is an object of type readDistrs, as returned by getDistrs. The plot allows to visualize the fragment length and read start distributions in a given sample.

signature(x = "readDistrsList", y = "ANY") x is an object of type readDistrsList storing fragment length and read start distributions for multiple samples.

signature(x = "readDistrsList") x is an object of type readDistrsList storing fragment length and read start distributions for multiple samples.

Examples

```
#See getDistrs examples
```

plotExpr *Plot inferred gene structure and expression.*

Description

Plots variants with sufficiently large posterior probability of being expressed along with their (marginal) estimated expression.

Usage

```
plotExpr(gene, minProbExpr = 0.5, minExpr = 0.1,
    xlab = "(kb)", ylab = "", xlim, cex = 1, yaxt = "n", col, ...)
```

Arguments

gene	denovoGeneExpr object containing results for a single gene, as returned by calcDenovo.
minProbExpr	Variants with marginal posterior probability of expression below minProbExpr are not reported
minExpr	Variants with (marginal) estimated expression below minExpr are not reported. Can be useful to remove sequence preference artifacts.
xlab	x-axis label, passed on to plot
ylab	y-axis label, passed on to plot
xlim	x-axis limits, passed on to plot
cex	Character expansion, passed on to plot
yaxt	Type of y-axis, passed on to plot
col	Colors for each variant, defaults to rainbow colors. It is possible to specify a single color.
...	Other arguments to be passed on to plot

Details

The marginal posterior probability that a variant is expressed is the sum of the posterior probabilities of all models containing that variant.

The marginal estimated expression is the average expression across all models (including those where the variant has 0 expression) weighted by the posterior probability of each model.

Methods

signature(gene = "denovoGeneExpr") gene contains the results from a de novo isoform expression analysis for a single gene, as returned by calcDenovo. When calcDenovo is run on multiple genes simultaneously, the desired gene can be selected using the "[[" operator as usual.

Examples

```
#See calcDenovo examples
```

---

plotPriorAS *Plot prior distribution on set of expressed variants (i.e. the model space).*

Description

Plots the prior distribution on the number of expressed variants and the number of exons per variant in genes with exons exons (as returned by function modelPrior). The prior distribution is compared to the observed frequencies to check that the assumed distributional forms are reasonable.

Usage

```
plotPriorAS(object, type="nbVariants", exons=1:9, xlab,
ylab="Probability", col=c("red","blue"))
```

Arguments

object	modelPriorAS object with prior distribution on model space.
type	Set to "nbVariants" to plot the prior on the number of variants per gene. Set to "nbExons" to plot the prior on the number of exons.
exons	Vector with integers. The plot is only produced with number of exons indicated in exons.
xlab	x-axis label, passed on to plot
ylab	y-axis label, passed on to plot
col	Colors for bars showing prior probabilities and frequencies in the known genome

Methods

signature(object = "modelPriorAS") object contains the prior distribution on the model space, as returned by function modelPrior

Examples

```
#See modelPrior examples
```

---

probNonEquiv

probNonEquiv *performs a Bayesian hypothesis test for equivalence between group means. It returns the posterior probability that* |*mu1-mu2*|>*logfc.* pvalTreat *is a wrapper to* treat *in package* limma*, which returns P-values for the same hypothesis test.*

Description

probNonEquiv computes v_i=P(|theta_i| > logfc | data), where theta_i is the difference between group means for gene i. This posterior probability is based on the NNGCV model from package EBarrays, which has a formulation similar to limma in an empirical Bayes framework. Notice that the null hypothesis here is that |theta_i|<logfc, e.g. isoforms with small fold changes are regarded as uninteresting.

Subsequent differential expression calls are based on selecting large v_i. For instance, selecting v_i >= 0.95 guarantees that the posterior expected false discovery proportion (a Bayesian FDR analog) is below 0.05.

Usage

```
probNonEquiv(x, groups, logfc = log(2), minCount, method = "plugin", mc.cores=1)

pvalTreat(x, groups, logfc = log(2), minCount, p.adjust.method='none', mc.cores = 1)
```

Arguments

x	ExpressionSet containing expression levels, or list of ExpressionSets
groups	Variable in fData(x) indicating the two groups to compare (the case with more than 2 groups is not implemented).
logfc	Biologically relevant threshold for the log fold change, i.e. difference between groups means in log-scale

minCount	If specified, probabilities are only computed for rows with fData(x)$readCount >= minCount
method	Set to 'exact' for exact posterior probabilities (slower), 'plugin' for plug-in approximation (much faster). Typically both give very similar results.
mc.cores	Number of parallel processors to use. Ignored unless x is a list.
p.adjust.method	P-value adjustment method, passed on to p.adjust

Value

If x is a single ExpressionSet, probNonEquiv returns a vector with posterior probabilities (NA for rows with less than minCount reads). pvalTreat returns TREAT P-values instead.

If x is a list of ExpressionSet, the function is applied to each element separately and results are returned as columns in the output matrix.

Author(s)

Victor Pena, David Rossell

References

Rossell D, Stephan-Otto Attolini C, Kroiss M, Stocker A. Quantifying Alternative Splicing from Paired-End RNA-sequencing data. Annals of Applied Statistics, 8(1):309-330

McCarthy DJ, Smyth GK. Testing significance relative to a fold-change threshold is a TREAT. Bioinformatics, 25(6):765-771

See Also

treat in package limma, p.adjust

Examples

```
#Simulate toy data
p <- 50; n <- 10
x <- matrix(rnorm(p*2*n),nrow=p)
x[(p-10):p,1:n] <- x[(p-10):p,1:n] + 1.5
x <- new("ExpressionSet",exprs=x)
x$group <- rep(c('group1','group2'),each=n)

#Posterior probabilities
pp <- probNonEquiv(x, groups='group', logfc=0.5)
d <- rowMeans(exprs(x[,1:n])) - rowMeans(exprs(x[,-1:-n]))
plot(d,pp,xlab='Observed log-FC')
abline(v=c(-.5,.5))

#Check false positives
truth <- rep(c(FALSE,TRUE),c(p-11,11))
getRoc(truth, pp>.9)
getRoc(truth, pp>.5)
```

procBam *Process BAM object*

Description

Process paired-end data stored in BAM object generated by scanBam. Outputs GRanges objects for reads and junctions.

Usage

```
procBam(bam, stranded=FALSE, seed=as.integer(1), verbose=FALSE, rname='null',
keep.junx=FALSE, keep.flag=FALSE, ispaired=TRUE,...)
```

Arguments

bam	BAM object generated by scanBam
stranded	Set to TRUE to indicate that the RNA-seq experiment preserved the strand information.
seed	Seed for random number generator
verbose	Set to TRUE to print progress information.
rname	Chromosome to process be combined with the which argument in the scanBam function
keep.junx	Option to store junction information. Only useful for finding denovo exons and transcripts.
keep.flag	Option to store aligment flag information.
ispaired	Set to TRUE is reads are paired.
...	Other arguments

Details

In case of multihits with same start position for both reads but different insertions/deletions patterns only one alignment is chosen at random.

Value

An object of class procBam containing reads with both ends correctly aligned and split according to the corresponding CIGAR. Unique identifiers by fragment are stored. Junctions spanned by reads are also stored in GRanges object if the argument \'keep.junx\' is set to TRUE.

Methods

signature(bam='list',stranded='logical',seed='integer',verbose='logical', rname='character',keep
: Process paired-end data stored in BAM object generated by scanBam. Outputs GRanges objects for reads and (optionally) junctions.

Author(s)

Camille Stephan-Otto Attolini

See Also

scanBam from package Rsamtools, help("procBam-class"), getReads.

Examples

```
##See example in calcExp
```

---

procBam-class *Class "procBam"*

Description

Stores processed bam files in a RangedData format. Each read is split into disjoint ranges according to its cigar code.

Objects from the Class

Objects are created with a call to procBam.

Slots

pbam GRanges indicating chromosome, start and end of each disjoint range. The pair id and read id within the pair are also stored.

junx GRanges indicating chromosome, start and end of junctions spanned by reads.

stranded Logical variable. TRUE indicates that the reads were obtained from and RNA-seq experiment where strand information was preserved.

In the case of stranded experiments:

plus GRanges indicating chromosome, start and end of each disjoint range for fragments originated from the positive strand. The pair id and read id within the pair are also stored.

minus GRanges indicating chromosome, start and end of each disjoint range for fragments originated from the negative strand. The pair id and read id within the pair are also stored.

pjunx GRanges indicating chromosome, start and end of junctions spanned by reads originated from the positive strand.

mjunx GRanges indicating chromosome, start and end of junctions spanned by reads originated from the negative strand.

Methods

show signature(object = "procBam"): Displays general information about the object.

getReads signature(x = "procBam"): Extracts the aligned reads stored in x.

Author(s)

Camille Stephan-Otto Attolini, David Rossell

See Also

getReads

Examples

```
showClass("procBam")
```

procGenome *Create an annotatedGenome object that stores information about genes and transcripts*

Description

procGenome processes annotations for a given transcriptome, either from a TxDb object created by GenomicFeatures package (e.g. from UCSC) or from a user-provided GRanges object (e.g. by importing a gtf file).

createDenovoGenome creates a de novo annotated genome by combining UCSC annotations and observed RNA-seq data.

Usage

```
procGenome(genDB, genome, mc.cores=1, verbose=TRUE)

createDenovoGenome(reads, DB, minLinks=2,
maxLinkDist=1e+05, maxDist=1000, minConn=2, minJunx=3, minLen=12, mc.cores=1)
```

Arguments

genDB	Either a TxDb object with annotations (e.g. from UCSC or a gtf file or a GRanges object as returned by import from rtracklayer package). See details.
genome	Character indicating genome version (e.g. "hg19", "dm3")
mc.cores	Number of cores to use in parallel processing (multicore package required)
verbose	Set to TRUE to print progress information
DB	annotatedGenome object, as returned by procGenome
minLinks	Minimum number of reads joining two exons to merge their corresponding genes
maxLinkDist	Maximum distance between two exons to merge their correspondin genes. A value of 0 disables this option.
maxDist	Maximum distance between two exons with reads joining them to merge their corresponding genes.
minConn	Minimum number of fragments connecting a new exon to an annotated one to add to denovo genome.
minJunx	Minimum number of junctions needed to redefine an annotated exon's end or start.
minLen	Minimum length of a junction to consider as a putative intron.
reads	Processed reads stored in a RangedData, as returned by procBam

Details

These functions create the annotation objects that are needed for subsequent functions. Typically these objects are created only once for a set of samples.

If interested in quantifying expression for known transcripts only, one would typically use procGenome with a TxDb from the usual Bioconductor annotations, e.g. genDB<-makeTxDbFromUCSC(genome="hg19",tablename= or imported from a gtf file e.g. genDB<-makeTxDbFromGFF('transcripts.gft',format='gtf'). GRanges

object (e.g. genDB <- import('transcripts.gtf')). Package GenomicFeatures contains more info about how to create TxDb objects. Alternatively, one can provide annotations as a GRanges object whith is returned when importing a gtf file with function import (package rtracklayer).

The output from procGenome can be used in combination with wrapKnown, which quantifies expression for a set of known transcripts, or wrapDenovo, which uses Bayesian model selection methods to assess which transcripts are truly expressed. When using wrapDenovo, you should create a single annotatedGenome object that combines information from all samples (e.g. from a gtf file produced by running your favorite isoform prediction software jointly on all samples), as this increases the power to detect new exons and isoforms.

Value

Object of class annotatedGenome.

Methods

signature(genDB = "transcriptDb") genDB is usually obtained with a call to makeTxDbFromUCSC (package GenomicFeatures), e.g. genDB<-makeTxDbFromUCSC(genome="hg19", tablename="refGene")

signature(genDB = "GRanges") genDB stores information about all transcripts and their respective exons. Chromosome, start, end and strand are stored as usual in GRanges objects. genDB must have a column named "type" taking the value "transcript" for rows corresponding to transcript and "exon" for rows corresponding to exons. It must also store transcript and gene ids. For instance, Cufflinks RABT module creates a gtf file with information formatted in this manner for known and de novo predicted isoforms.

Author(s)

Camille Stephan-Otto Attolini

See Also

See annotatedGenome-class for a description of the class. See methods transcripts to extract exons in each transcript, getIsland to obtain the island id corresponding to a given transcript id See splitGenomeByLength for splitting an annotatedGenome according to gene length.

Examples

```
## Known transcripts from Bioconductor annotations
## library(TxDb.Hsapiens.UCSC.hg19.knownGene)
## hg19DB <- procGenome(TxDb.Hsapiens.UCSC.hg19.knownGene, genome='hg19')

## Alternative using makeTxDbFromUCSC
## genDB<-makeTxDbFromUCSC(genome="hg19", tablename="refGene")
## hg19DB <- procGenome(genDB, "hg19")

## Alternative importing .gtf file
## genDB.Cuff <- import('transcripts.gtf')
## hg19DB.Cuff <- procGenome(genDB.Cuff, genome='hg19')
```

qqnormGenomeWide *Genome-wide qq-normal and qq-gamma plots*

Description

qqnormGenomeWide overlays quantile-quantile normal plots (qqnorm) for a series of genes (rows in the input matrix), to provide an overall assessment of Normality. Similarly, qqgammaGenomeWide overlays quantile-quantile gamma plots.

Note that the theoretical quantiles for z-scores under a Normal are the same for all genes, but the gamma theoretical quantiles depend on the Gamma parameter estimates for each gene and hence the theoretical quantiles are different for each gene (resulting in different x-values in each qq-plot)

Usage

```
qqnormGenomeWide(x, ngenes=min(1000, nrow(x)), ...)

qqgammaGenomeWide(x, ngenes=min(1000, nrow(x)), ...)
```

Arguments

x	ExpressionSet, matrix or data.frame with genes/isoforms in rows
ngenes	A qqnorm plot is produced for the first ngenes rows in x
...	Other arguments to be passed on to codeplot

Value

Produces a figure overlaying qq-normal or qq-gamma plots for ngenes comparing observed vs. theoretical quantiles

Author(s)

David Rossell

Examples

```
mu <- rnorm(100)
x <- matrix(rnorm(100*5,mu),ncol=5)

qqnormGenomeWide(x)
qqgammaGenomeWide(exp(x))
```

quantileNorm *Apply quantile normalization*

Description

Perform quantile normalization on the columns of a matrix or ExpressionSet

Usage

```
quantileNorm(x)
```

Arguments

x	ExpressionSet or matrix

Value

Returns x with quantile normalized columns

Author(s)

David Rossell

Examples

```
x <- cbind(rnorm(1000),rnorm(1000,2,4))
boxplot(x)

xnorm <- quantileNorm(x)
boxplot(xnorm)
```

relexprByGene *Compute relative expressions within each gene*

Description

Transforms relative expressions that add up to 1 within each gene island (the default output of casper) to relative expressions that add up to 1 per gene.

Usage

```
relexprByGene(x, normbylength=FALSE, genomeDB)
```

Arguments

x	ExpressionSet containing relative expressions. (typically, adding up to 1 for each island_id) Column gene_id in fData(x) should contain a unique gene identifier.
normbylength	If set to TRUE, isoform expressions are divided by isoform length before re-normalizing. This is useful for taking into account that longer isoforms produce more reads than shorter isoforms.
genomeDB	If normbylength==TRUE, genomeDB should be an annotatedGenome object containing the annotated genome (see procGenome)

Value

ExpressionSet with relative expressions adding up to one for each gene_id.

Author(s)

David Rossell

Examples

```
#See help(calcExp)
```

---

rmShortInserts *Remove reads with short insert sizes from imported BAM files.*

Description

In paired-end experiments short inserts (i.e. the 2 ends being very close to each other), may indicate RNA degradation or that a short RNA (e.g. miRNA) is being sequenced. Typically the goal is not to study alternative splicing for such short/degraded RNA; in this case it is recommendable to remove such short inserts to avoid biasing the insert size distribution. Requiring a minimum insert size can also result in significantly faster computations when quantifying alternative splicing via calc or calcDenovo.

Usage

```
rmShortInserts(bam, isizeMin=100)
```

Arguments

bam	Object with aligned reads, as returned by scanBam
isizeMin	Reads with insert size smaller than isizeMin will be removed.

Value

Named list, in the same format as that returned by scanBam.

Note

The insert size is stored in objects imported with scanBam in the element named isize.

Author(s)

David Rossell

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##-- or do  help(data=index)  for the standard data sets.
```

simMAE *Simulate Mean Absolute Error (MAE) in estimating isoform expression under various experimental settings.*

Description

Simulate several future RNA-seq data under various experimental settings (sequencing depth, read length, insert sizes), estimate isoform expression and assess the MAE incurred in the estimation process. The function is a wrapper combining functions simReads and calcExp.

Usage

```
simMAE(nsim, islandid, nreads, readLength, fragLength, burnin=1000, pc, distr, readLength.pilot=rea
```

Arguments

nsim	Number of RNA-seq datasets to generate (often as little as nsim=10 suffice)
islandid	When specified this argument indicates to run the simulations only for gene islands with identifiers in islandid. When not specified genome-wide simulations are performed.
nreads	Vector indicating the target number of read pairs for each experimental setting. The actual number of reads differs from nreads to account for non-mappability and random read yield (see details)
readLength	Vector indicating the read length in each experimental setting
fragLength	Vector indicating the mean insert size in each experimental setting
burnin	Number of MCMC burn-in samples (passed on to calcExp)
pc	Observed path counts in pilot data. When not specified, these are simulated from eset.pilot
distr	Estimated read start and insert size distributions in pilot data
readLength.pilot	Read length in pilot data
eset.pilot	ExpressionSet with pilot data expression in log2-RPKM, used to simulate pc when not specified by the user. See details
usePilot	By default casper assumes that the pilot data is from a related experiment rather than the current tissue of interest (usePilot=FALSE). Hence, the pilot data is used to simulate new RNA-seq data but not to estimate its expression. However, in some cases we may be interested in re-sequencing the pilot sample at deeper length, in which case one would want to combine the pilot data with the new data to obtain more precise estimates. This can be achieved by setting usePilot=TRUE
retTxsError	If retTxsError=TRUE, simMAE returns posterior expected MAE for each individual isoform. This option is not available when eset.pilot is specified instead of pc. Else the output is a data.frame with overall MAE across all isoforms
genomeDB	annotatedGenome object, as returned by procGenome
mc.cores	Number of cores to use in the expression estimation step, passed on to calcExp
mc.cores.int	Number of cores to simulate RNA-seq datasets in parallel

verbose	Set verbose=TRUE to print progress information
writeBam	Set to TRUE to write simulated reads to a .bam file
bamFile	Name of the .bam file

Details

simMAE simulates nsim datasets under each experimental setting defined by nreads, readLength, fragLength. For each dataset the following steps are performed:

1. The number of reads is nreads * readYield * pmapped, where readYield= runif(1,0.8,1.2) accounts for deviations in read yield and pmapped= runif(1,0.6,0.9)*pmappable is the proportion of mapped reads (60%-90% of the mappable reads according to the piecewise-linear power law of Li et al (2014))

2. True expression levels pi are generated from their posterior distribution given the pilot data.

3. Conditional on pi, RNA-seq data are generated and expression estimates pihat are obtained using calcExp

4. The mean absolute estimation error sum(abs(pihat-pi)) across all isoforms is computed

Ideally simMAE should use pilot data from a relevant related experiment to simulate what future data may look like for the current experiment of interest. The recommened way to do this is to download a .bam file from such a related experiment and processing it in casper with function wrapKnown, as then both gene and isoform expression can be estimated accurately. The object output by wrapKnown is a list with elements named 'pc', 'distr' which can be given as input to simMAE.

As an alternative to specifying pc, simMAE allows setting eset.pilot as pilot data. Gene and isoform expression are then simulated as follows:

1. The number of reads per gene is generated from a Multinomial distribution with success probabilities proportional to 2^exprs{eset.pilot}.

2. Relative isoform expression within each gene are generated from a symmetric Dirichlet distribution with parameter 1/Ig, where Ig is the number of isoforms in gene g.

We emphasize that relative isoform expressions are not trained from the pilot data, and that while the distribution of gene expression levels resembles that in eset.pilot, no attempt is made to match gene identifiers and hence the results for individual genes should not be trusted (hence this option is only available when retTxsError==FALSE.

Value

If retTxsError==TRUE, simMAE returns posterior expected MAE for each individual isoform. Else the output is a data.frame with overall MAE across all isoforms

References

Stephan-Otto Attolini C., Pena V., Rossell D. Bayesian designs for personalized alternative splicing RNA-seq studies (2014)

Li, W. and Freudenberg, J. and Miramontes, P. Diminishing return for increased Mappability with longer sequencing reads: implications of the k-mer distributions in the human genome. BMC Bioinformatics, 15, 2 (2014)

See Also

wrapKnown,simReads,calcExp

Examples

```
## maybe str(simMAE) ; plot(simMAE) ...
```

---

simMAEcheck *Model checking for One Sample Problems.*

Description

Simulates RNA-seq data under the same experimental setting as in the observed data, and compares the observed vector of number of reads per gene with the simulations.

Usage

```
simMAEcheck(nsim, islandid, burnin=1000, pc, distr, readLength.pilot, eset.pilot, usePilot=FALSE, r
```

Arguments

nsim	Number of RNA-seq datasets to generate (often as little as nsim=10 suffice)
islandid	When specified this argument indicates to run the simulations only for gene islands with identifiers in islandid. When not specified genome-wide simulations are performed.
burnin	Number of MCMC burn-in samples (passed on to calcExp)
pc	Observed path counts in pilot data. When not specified, these are simulated from eset.pilot
distr	Estimated read start and insert size distributions in pilot data
readLength.pilot	Read length in pilot data
eset.pilot	ExpressionSet with pilot data expression in log2-RPKM, used to simulate pc when not specified by the user. See details
usePilot	By default casper assumes that the pilot data is from a related experiment rather than the current tissue of interest (usePilot=FALSE). Hence, the pilot data is used to simulate new RNA-seq data but not to estimate its expression. However, in some cases we may be interested in re-sequencing the pilot sample at deeper length, in which case one would want to combine the pilot data with the new data to obtain more precise estimates. This can be achieved by setting usePilot=TRUE
retTxsError	If retTxsError=TRUE, simMAE returns posterior expected MAE for each individual isoform. This option is not available when eset.pilot is specified instead of pc. Else the output is a data.frame with overall MAE across all isoforms
genomeDB	annotatedGenome object, as returned by procGenome
mc.cores	Number of cores to use in the expression estimation step, passed on to calcExp
mc.cores.int	Number of cores to simulate RNA-seq datasets in parallel
verbose	Set verbose=TRUE to print progress information

Details

simMAEcheck simulates nsim datasets under the same experimental setting as in the observed data. For more details, please check the documentation for simMAE, which is the basis of this function.

Value

The output is a list with 2 entries. The first entry is a `data.frame` with overall MAE across all isoforms in the simulations (see `simMAE` for details). The second entry contains the expected number of genes for which the number of reads in the data lies in the range of the posterior predictive simulations (under the hypothesis that they have the same distribution) and the actual number of genes for which the condition is satisfied.

References

Stephan-Otto Attolini C., Pena V., Rossell D. Bayesian designs for personalized alternative splicing RNA-seq studies (2014)

Li, W. and Freudenberg, J. and Miramontes, P. Diminishing return for increased Mappability with longer sequencing reads: implications of the k-mer distributions in the human genome. BMC Bioinformatics, 15, 2 (2014)

See Also

wrapKnown,simReads,calcExp

Examples

```
#Run casperDesign() to see full manual with examples
```

`simMultSamples` *Simulate paired end reads for multiple future samples based on pilot data, and obtain their expression estimates via casper*

Description

Simulate true expression levels and observed data (casper expression estimates) for future samples within each group.

These simulations serve as the basis for sample size calculation: if one were to sequence `nsamples` new RNA-seq samples, what data would we expect to see? The simulation is posterior predictive, i.e. based on the current available data `x`.

Usage

```
simMultSamples(nsim, nsamples, nreads, readLength, fragLength, x,
groups='group', distrs, genomeDB, model='LNNMV', verbose=TRUE, mc.cores=1)
```

Arguments

`nsim`	Number of simulations to obtain
`nsamples`	Vector indicating number of future samples per group, e.g. `nsamples=c(5,5)` to simulate 5 new samples for 2 groups.
`nreads`	Desired number of paired-end reads per sample. The actual number of aligned reads for any given sample differs from this amount, see details.
`readLength`	Read length, i.e. in an experiment with paired reads at 100bp each, `readLength=100`.
`fragLength`	Desired average insert size (size of RNA molecules after fragmentation). If missing, insert sizes are as obtained from `distrs`

x	ExpressionSet containing pilot data. x[[group]] indicates groups to be compared
groups	Name of column in pData(x) indicating the groups
distrs	Fragment start and length distributions. It can be either an object or a list of objects of class readDistrs. In the latter case, an element is chosen at random for each individual sample to consider uncertainty in these distributions. If not specified, it defaults to data(distrsGSE37704).
genomeDB	annotatedGenome object
model	Set to 'LNNMV' to simulate from log-normal normal with modified variance model (Yuan and Kendsiorski, 2006), or to 'GaGa' to simulate from the GaGa model (Rossell, 2009). See details.
verbose	Set to TRUE to print progress
mc.cores	Number of cores to use in function. mc.cores>1 requires package parallel

Details

The posterior predictive simulations is based on four steps: (1) simulate true expression for each group (mean and SD), (2) simulate true expression for future samples, (3) simulate paired reads for each future sample, (4) estimate expression from the reads via Casper. Below are some more details.

1. Simulate true mean expression in each group and residual variance for each gene. If model=='LNNMV' this is based on the log-normal normal with modified variance model in package EBarrays (Yuan & Kendziorski 2006), if model=='GaGa' this is based on the GaGa model (Rossell, 2009). adapted to take into account that the expression estimates in the pilot data x are noisy (which is why simMultSamples requires the SE / posterior SD associated to exprs(x)). The simulated values are returned in component "simTruth" of the simMultSamples output.

2. Simulate true isoform expression for each of the future samples. These are independent Normal draws with mean and variance generated in step 1. True gene expression is derived from the isoform expressions.

3. Determine the number of reads to be simulated for each gene based on its true expression (generated in step 2) and a Multinomial sampling model. For each sample:

- The number of reads yielded by the experiment is Unif(.8*nreads,1.2*nreads) - A proportion of non-mappable reads is discarded using the power law in Li et al (2014) - Amongst remaining reads, we assume that a proportion Unif(0.6,0.9) were aligned (consistenly with reports from ENCODE project)

The final number of simulated reads is reported in component "simExpr" of the simMultSamples output.

4. Obtain expression estimates from the path counts produced in step 3 via calcExp. These are reported in component "simExpr" of the simMultSamples output.

Value

Object of class simulatedSamples, which extends a list of length nsim. See the class documentation for some helpful methods (e.g. coef, exprs, mergeBatches). Each element is itself a list containing an individual simulation.

simTruth	data.frame indicating the mean and standard deviation of the Normal distribution used to generate data from each group

simExpr	ExpressionSet with Casper expression estimates, as returned by calcExp. pData(simExpr) indicates group information, and fData(simExpr) the number of simulated reads for each sample (in columns 'explCnts') and across all samples (in column 'readCount')

Author(s)

Victor Pena, David Rossell

References

Rossell D. (2009) GaGa: a Parsimonious and Flexible Model for Differential Expression Analysis. Annals of Applied Statistics, 3, 1035-1051.

Stephan-Otto Attolini C., Pena V., Rossell D. Bayesian designs for personalized alternative splicing RNA-seq studies (2015)

Yuan, M. and Kendziorski, C. (2006). A unified approach for simultaneous gene clustering and differential expression identification. Biometrics, 62, 1089-1098.

Examples

```
#Run casperDesign() to see full manual with examples
```

---

simReads

Function to simulate paired end reads following given read start and fragment length distributions and gene and variant expressions.

Description

This function generates path counts and bam files with simulated paired end reads according to given read start distribution, fragment length distribution and gene and variant expressions.

Usage

```
simReads(islandid, nSimReads, pis, rl, seed, writeBam, distrs, genomeDB,
repSims=FALSE, bamFile=NULL, stranded=FALSE, verbose=TRUE, chr=NULL, mc.cores=1)
```

Arguments

islandid	Island ID's from the genomeDB object to simulate reads
nSimReads	Named numeric vector with number of fragments to simulate in each island.
pis	Named numeric vector with relative expression of transcripts. Expressions add up to one for each island to simulate.
rl	Read length
seed	Seed of the random numbers generator
writeBam	Set to 1 to generate bam files with the simulated reads
distrs	Object of class 'readDistrs' with read start and fragment length distributions
genomeDB	Object of class 'annotatedGenome' with the genome to genererate reads from
repSims	Set to TRUE to return relative read starts and fragment lengths from the simulation

bamFile	Name of the bam file to write reads to. Must end with '.bam'
stranded	Set to TRUE to preserve gene strand when generating reads. The 'XS' tag will be added to reads in the bam file and the returned 'pc' object will be stranded
verbose	Set to TRUE to print progress
chr	Characters vector with chromosomes to simulate. Defaults to whole genome simulations.
mc.cores	Number of cores to use in function

Value

Nsim	Numerical vector with the number of reads simulated for each island.
pc	Object of class 'pathCounts' with simulated path counts
sims	Only if 'repSims' is set to TRUE. List with vectors of length 'n' with the following elements: -'varl': Length of variant for corresponding read -'st' Start of fragment relative to variant start (not in genomic coordinates) -len:Fragment length -'strand':Strand of gene for simulated read

Author(s)

Camille Stephan-Otto Attolini

Examples

```
data(hg19DB)
data(K562.r1l1)
distrs <- getDistrs(hg19DB,bam=K562.r1l1,readLength=75)

islandid <- c('10319','463')
txs <- unlist(lapply(hg19DB@transcripts[islandid], names))
pis <- vector(mode='numeric', length=length(txs))
npis <- sapply(hg19DB@transcripts[islandid],length)
pis[1:npis[1]] <- rep(1/npis[1],npis[1])
pis[-1:-npis[1]] <- rep(1/npis[2],npis[2])
names(pis) <- txs
nSimReads <- c(100, 100)
names(nSimReads) <- islandid

simpc <- simReads(islandid=islandid, nSimReads=nSimReads, pis=pis,
rl=75, repSims=TRUE, seed=1, writeBam=FALSE, distrs=distrs,genomeDB=hg19DB)
```

---

simulatedSamples-class

Class "simulatedSamples"

Description

simulatedSamples stores multiple simulated isoform expression datasets. Each dataset contains the (simulation) true mean expression in each group and residual variance, as well as the estimated expression in each individual sample.

Objects from the Class

Objects are returned by `simMultSamples`.

Slots

The class extends a list directly.

.Data A list, each element containing a different simulated dataset

Methods

show `signature(object = "simulatedSamples")`: Displays general information about the object.

coef `signature(object = "simulatedSamples")`: Returns a matrix with difference between group means (simulation truth) in all simulated datasets

exprs `signature(object = "simulatedSamples")`: Returns a list of ExpressionSets containing the estimated expressions in each simulation.

mergeBatches `signature(x="ExpressionSet",y="simulatedSamples")`: Combines `x` with each element in `exprs` in `y`, and returns a list. See `help(mergeBatches)` for more details.

"[" `x[i]` selects a subset of simulations, `x[,j]` a subset of the samples in each simulation

Author(s)

David Rossell

See Also

`mergeBatches`

Examples

```
showClass("simulatedSamples")
```

---

splitGenomeByLength *Split an annotatedGenome object into subsets according to gene length*

Description

`splitGenomeByLength` splits an annotatedGenome according to gene length (bp), which allows estimating the fragment start and length distribution for each subset separately.

Usage

```
splitGenomeByLength(DB, breaks=c(0,3000,5000,Inf))
```

Arguments

`DB`	Object containing annotated genome. Must be of class `annotatedGenome`, as returned by `procGenome` or `createDenovoGenome`.
`breaks`	Breakpoints to define gene subgroups.

Details

By default groups are <3000bp, 3000-5000bp, >5000bp, which work well for the human genome. Further sub-dividisions may result in unstable estimates of fragment start and length distributions.

Value

List where each component is of class `annotatedGenome`.

Author(s)

David Rossell

See Also

`procGenome` and `createDenovoGenome` for creating `annotatedGenome` objects. `getDistrs` for estimating fragment start and length distribution.

Examples

```
##Not run
## genDB<-makeTranscriptDbFromUCSC(genome="hg19", tablename="refGene")
## hg19DB <- procGenome(genDB, "hg19")
## hg19split <- splitGenomeByLength(hg19DB)
```

---

subsetGenome — *subsetGenome subsets an object of class annotatedGenome for a set of island IDs or chromosome names.*

Description

~~ Methods for function `subsetGenome` in package **casper** ~~ Subset an annotatedGenome object by islands or chromosomes.

Usage

```
subsetGenome(islands, chr, genomeDB)
```

Arguments

`islands`	Vector of characters with the island IDs to retrieve from genome.
`chr`	Vector of characters with the names of chromosomes to retrieve from genome.
`genomeDB`	annotatedGenome object with genome to subset.

Methods:

`signature(islands = "character", chr = "missing", genomeDB = "annotatedGenome")` Subset annotatedGenome object by a set of island IDs.

`signature(islands = "missing", chr = "character", genomeDB = "annotatedGenome")` Subset annotatedGenome object by chromosomes.

`transcripts` *Extracts transcript information (exon start and ends) from an* `annotatedGenome` *object, either for all transcripts or only those corresponding to a given island or transcript.*

Description

annotatedGenome objects store information regarding genes and transcripts. When there's an overlap in exons between several genes, these genes are grouped into gene islands.

`transcripts` retrieves all stored transcripts for a given transcript or island.

`matchTranscripts` finds transcripts in queryDB matching a transcript in subjectDB. The best match for each transcript in subjectDB is returned, unless difference in bp is >maxbp

Usage

```
transcripts(genomeDB, txid, islandid)

matchTranscripts(queryDB, subjectDB, maxbp=10)
```

Arguments

`genomeDB`	Object of class `annotatedGenome`
`txid`	Character indicating transcript identifier (optional)
`islandid`	Character indicating island identifier (optional)
`queryDB`	`annotatedGenome` with query transcripts
`subjectDB`	`annotatedGenome` with potentially matching transcripts
`maxbp`	Maximum difference in bp for transcripts to be matched

Value

`IRangesList` where each element in the list corresponds to a different transcript.

Methods

`signature(genomeDB = "annotatedGenome", txid="missing", islandid="missing")` Return exons for all transcripts in `genomeDB`

`signature(genomeDB = "annotatedGenome", txid="character", islandid="missing")` Return exons for transcript `txid`

`signature(genomeDB = "annotatedGenome", txid="missing", islandid="character")` Return exons for all transcripts in island `islandid`

See Also

`genePlot` to plot the resulting transcripts

Examples

```
data(hg19DB)
txs <- transcripts(txid="NM_005158",genomeDB=hg19DB)
txs
```

txLength ~~ *Methods for Function* txLength *in Package* **casper** ~~

Description

~~ Methods for function txLength in package **casper** ~~ Function to retrieve transcript lengths from annotated genome (class genomeDB).

Usage

```
txLength(islandid, txid, genomeDB)
```

Arguments

islandid	Retrieve length for transcripts in island islandid.
txid	Retrieve length for txid transcripts.
genomeDB	Annotated genome of class genomeDB.

Details

When called for the first time lengths are calculated and stored in the object genomeDB. Subsequent calls refer to these computed values.

Value

Named numeric vector with transcript lengths.

Methods:

signature(islandid = "character", txid = "missing", genomeDB = "annotatedGenome") Retrieve lengths from genomeDB for transcripts in islandid islands.

signature(islandid = "missing", txid = "character", genomeDB = "annotatedGenome") Retrieve lengths from genomeDB for txid transcripts.

signature(islandid = "missing", txid = "missing", genomeDB = "annotatedGenome") Retrieve or calculate lengths for all transcripts in the annotated genome genomeDB.

wrapDenovo *Run all necessary steps to get expression estimates from multiple bam files with the casper pipeline.*

Description

Function to analyze bam files to generate an ExpressionSet with expression estimates for all samples, read start and fragment length distributions, path counts and optinally processed reads.

Usage

```
wrapDenovo(bamFile, output_wrapKnown, knownGenomeDB, targetGenomeDB, readLength,
  rpkm=TRUE, keep.multihits=TRUE, searchMethod="submodels",
  exactMarginal=TRUE,  integrateMethod = "plugin", maxExons=40,
  islandid, chroms=NULL, keep.pbam=FALSE,  keepPbamInMemory=FALSE,
  niter=10^3, priorq=3, priorqGeneExpr=2,
  mc.cores.int=1, mc.cores=1, verbose=TRUE, seed=1)
```

Arguments

bamFile	Names of bam files with the sample to analyze. These must sorted and indexed, and the index must be in the same directory.
output_wrapKnown	Optional argument containing the output of an earlier call to wrapKnown. If provided, path counts, read start and insert size distributions are loaded from this output rather than being re-computed. Better leave this argument missing unless you know what you're doing.
knownGenomeDB	annotatedGenome object with known isoforms, e.g. from UCSC or GENCODE annotations. Used to set the prior probability that any given isoform is expressed. See help(calcDenovo) for details.
targetGenomeDB	annotatedGenome object with isoforms we wish to quantify. By default these are the same as in knownGenomeDB, but more typically targetGenomeDB is imported from a .gtf file produced by some isoform prediction software.
readLength	Read length in bp, e.g. in a paired-end experiment where 75bp are sequenced on each end one would set readLength=75.
rpkm	Set to TRUE to return reads per kilobase per million (RPKM), FALSE for relative expression levels. Important, relative expression adds up to 1 within gene island, NOT within gene. To get relative expressions within gene run relexprByGene afterwards. See help(wrapKnown).
keep.multihits	Set to FALSE to discard reads aligned to multiple positions.
searchMethod	Method used to perform the model search. "allmodels" enumerates all possible models (warning: this is not feasible for genes with >5 exons). "rwmcmc" uses a random-walk MCMC scheme to focus on models with high posterior probability. "submodels" considers that some isoforms in targetGenomeDB may not be expressed, but does not search for new variants. "auto" uses "allmodels" for genes with up to 5 exons and "rwmcmc" for longer genes. See help("calcDenovo").
exactMarginal	Set to FALSE to estimate posterior model probabilities as the proportion of MCMC visits. Set to TRUE to use the integrated likelihoods (default). See details.
integrateMethod	Method to compute integrated likelihoods. The default ('plugin') evaluates likelihood*prior at the posterior mode and is the faster option. Set 'Laplace' for Laplace approximations and 'IS' for Importance Sampling. The latter increases computation cost very substantially.
maxExons	Prior probabilities of isoform expression are estimated for genes with 1 up to maxExons exons separately, for genes with more than maxExons exons a combined estimate is used. See help("modelPrior")
islandid	Names of the gene island to be analyzed. If missing all gene islands are analyzed
chroms	Names of the chromosomes to be analyzed. If missing all chromosomes are analyzed.

keep.pbam	Set to TRUE to save processed bam object, as returned by procBam. This object can require substantial memory during execution and disk storage upon saving and is not needed for a default analysis.
keepPbamInMemory	Set to TRUE to keep processed bam objects in memory to speed up some computations.
niter	Number of MCMC iterations in the model search algorithm.
priorq	Parameter of the Dirichlet prior for the proportion of reads coming from each variant. We recommend priorq=3 as this defines a non-local prior that penalizes falsely predicted isoforms.
priorqGeneExpr	Parameter of the Dirichlet prior distribution on overall gene expression. Defaults to 2 to ensure non-zero estimates.
mc.cores	Number of cores to use in expression estimation.
mc.cores.int	Number of cores to use when loading bam files. Be careful as this is a memory intensive step.
verbose	Set to TRUE to display progress information.
seed	Set seed of random number generator.

Details

The function executes the functions procBam, getDistrs, pathCounts calcDenovo and denovoExpr and formats the output nicely. Running wrapDenovo is much more efficient in cpu speed and memory usage than running these functions separately.

When rpkm is false the function returns the estimated proportion of reads arising from each isoform within a gene island. See the details in help("wrapKnown") for more information on this.

Value

denovoGenomeDB	annotatedGenome that contains the isoforms in targetGenomeDB plus any new isoforms predicted by casper
.	
exp	Object of class ExpressionSet containing Bayesian model averaging expression estimates. See the fData for the posterior probability that each isoform is expressed.
distr	Object of class readDistrs
pbam	List of objects of class procBam with one element per chromosome

Author(s)

Miranda Stobbe, David Rossell

References

Rossell D, Stephan-Otto Attolini C, Kroiss M, Stocker A. Quantifying Alternative Splicing from Paired-End RNA-sequencing data. Annals of Applied Statistics, 8(1):309-330.

See Also

calcDenovo, wrapKnown, relexprByGene

Examples

```
## not run
## Known isoforms
##  library(TxDb.Hsapiens.UCSC.hg19.knownGene)
##  hg19DB <- procGenome(TxDb.Hsapiens.UCSC.hg19.knownGene), genome='hg19')

## gtf with known & de novo predictions
##  mygtf <- import('hg19_denovo.gtf')
##  hg19denovoDB <- procGenome(mygtf, genome='hg19')

## bamFile="/path_to_bam/sorted.bam"
##  ans <- wrapDenovo(bamFile=bamFile, targetGenomeDB=hg19denovoDB, knownGenomeDB=hg19DB, readLength=101)

## Estimated expression via BMA
##  head(exprs(ans[['exp']]))

## Posterior probability that each isoform is expressed
##  head(fData(ans[['exp']]))
```

`wrapKnown` *Run all necessary steps to get expression estimates from multiple bam files with the casper pipeline.*

Description

Function to analyze bam files to generate an ExpressionSet with expression estimates for all samples, read start and fragment length distributions, path counts and optinally processed reads.

Usage

```
wrapKnown(bamFile, verbose=FALSE, seed=1, mc.cores.int=1,
mc.cores=1, genomeDB, readLength, rpkm=TRUE, priorq=2, priorqGeneExpr=2,
citype='none', niter=10^3, burnin=100, keep.pbam=FALSE,
keep.multihits=TRUE, chroms=NULL)
```

Arguments

`bamFile`	Names of bam files with the sample to analyze. These must sorted and indexed, and the index must be in the same directory.
`verbose`	Set to `TRUE` to display progress information.
`seed`	Set seed of random number generator.
`mc.cores.int`	Number of cores to use when loading bam files. This is a memory intensive step, therefore number of cores must be chosen according to available RAM memory.
`mc.cores`	Number of cores to use in expression estimation.
`genomeDB`	`annotatedGenome` object containing annotated genome, as returned by the `procGenome` function.
`readLength`	Read length in bp, e.g. in a paired-end experiment where 75bp are sequenced on each end one would set `readLength=75`.

rpkm	Set to TRUE to return reads per kilobase per million (RPKM). Set to FALSE to return relative expression levels. Important, relative expression adds up to 1 within gene island, NOT within gene. To get relative expressions within gene run relexprByGene afterwards. See details.
priorq	Parameter of the prior distribution on the proportion of reads coming from each variant. The prior is Dirichlet with prior sample size for each variant equal to priorq. We recommend priorq=2 for estimation, as it pools the estimated expression away from 0 and 1 and returned lower estimation errors than priorq=1 in our simulated experiments.
priorqGeneExpr	Parameter for prior distribution on overall gene expression. Defaults to 2, which ensures non-zero estimates for all genes
citype	Set to "none" to return no credibility intervals. Set to "asymp" to return approximate 95% CIs (obtained via the delta method). Set to "exact" to obtain exact CIs via Monte Carlo simulation. Options "asymp" and especially "exact" can increase the computation time substantially.
niter	Number of Monte Carlo iterations. Only used when citype=="exact".
burnin	Number of burnin Monte Carlo iterations. Only used when citype=="exact".
keep.pbam	Set to TRUE to save processed bam object, as returned by procBam. This object can require substantial memory during execution and disk storage upon saving and is not needed for a default analysis.
keep.multihits	Set to FALSE to discard reads aligned to multiple positions.
chroms	Manually set chromosomes to be processed. By default only main chromosomes are considered (except 'chrM')

Details

The function executes the functions procBam, getDistrs and pathCounts in parallel for each chromosome, but is much more efficient in cpu speed and memory usage than running these functions separately. Data from multiple samples are then combined using mergeExp. Note that further normalization (e.g. quantileNorm) may be needed preliminary to actual data analysis.

When rpkm is false the function returns the estimated proportion of reads arising from each isoform within a gene island. casper groups two or more genes into a gene island whenever these genes share an exon (or part of an exon). Because exons are shared, isoform quantification must be done simultaneously for all those genes.

That is, the output from wrapKnown when rpkm is FALSE are proportions that add up to 1 within each island. If you would like to re-normalize these expressions so that they add up to 1 within each gene, see the help for function relexprByGene.

One last remark: casper returns the estimated proportion of reads generated by each isoform, which is not the same as relative isoform expressions. Longer isoforms tend to produce more reads than shorter isoforms. This is easily accounted for by dividing relative expressions by isoform length, see relexprByGene.

Value

distr	Object of class readDistrs
pbam	List of objects of class procBam with one element per chromosome
pc	Object of class pathCounts
exp	Object of class ExpressionSet

Author(s)

Camille Stephan-Otto Attolini, David Rossell

References

Rossell D, Stephan-Otto Attolini C, Kroiss M, Stocker A. Quantifying Alternative Splicing from Paired-End RNA-sequencing data. Annals of Applied Statistics, 8(1):309-330.

See Also

procGenome, relexprByGene, quantileNorm

Examples

```
## genDB<-makeTranscriptDbFromUCSC(genome="hg19", tablename="refGene")
## hg19DB <- procGenome(genDB, "hg19")
##  bamFile="/path_to_bam/sorted.bam"
## ans <- wrapKnown(bamFile=bamFile, mc.cores.int=4, mc.cores=3, genomeDB=hg19DB, readLength=101)
##  names(ans)
##  head(exprs(ans\$exp))
```

Index