Package 'gCrisprTools'

April 10, 2023

Type Package

Title Suite of Functions for Pooled Crispr Screen QC and Analysis

Version 2.4.0

Date 2021-08-23

Author Russell Bainer, Dariusz Ratman, Steve Lianoglou, Peter Haverty

Maintainer Russell Bainer <russ.bainer@gmail.com>

Description Set of tools for evaluating pooled high-throughput screening experiments,
typically employing CRISPR/Cas9 or shRNA expression cassettes. Contains methods for interrogating
library and cassette behavior within an experiment, identifying differentially abundant
cassettes, aggregating signals to identify candidate targets for empirical validation,
hypothesis testing, and comprehensive reporting. Version 2.0 extends these applications to include
a variety of tools for contextualizing and integrating signals across many experiments, incorporates
extended signal enrichment methodologies via the ``sparrow'' package, and
streamlines many formal requirements to aid in interpretablity.

License Artistic-2.0

Imports Biobase, limma, RobustRankAggreg, ggplot2,
SummarizedExperiment, grid, rmarkdown, grDevices, graphics,
methods, ComplexHeatmap, stats, utils, parallel

Suggests edgeR, knitr, AnnotationDbi, org.Mm.eg.db, org.Hs.eg.db,
BiocGenerics, markdown, RUnit, sparrow, msigdbr, fgsea

RoxygenNote 7.2.1

VignetteBuilder knitr

Encoding UTF-8

biocViews ImmunoOncology, CRISPR, PooledScreens, ExperimentalDesign,
BiomedicalInformatics, CellBiology, FunctionalGenomics,
Pharmacogenomics, Pharmacogenetics, SystemsBiology,
DifferentialExpression, GeneSetEnrichment, Genetics,
MultipleComparison, Normalization, Preprocessing,
QualityControl, RNASeq, Regression, Software, Visualization

NeedsCompilation no

Depends R (>= 4.1)

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

git_branch RELEASE_3_16

git_last_commit 154bc8e

git_last_commit_date 2022-11-01

Date/Publication 2023-04-10

R **topics documented:**

---

gCrisprTools-package   *gCrisprTools*

Description

Pipeline for using CRISPR screen data

---

aln   *Precalculated alignment statistics of a crispr screen*

Description

Example alignment matrix file for the provided example Crispr screen. All sample, gRNA, and Gene information has been anonymized and randomized.

Source

Genentech, Inc.

See Also

Please see 'vignettes/Crispr_example_workflow.R' for details.

Examples

```
data('aln')
head(aln)
```

---

ann   *Annotation file for a mouse Crispr library*

Description

Example annotation file for the screen data provided in `es`. All sample, gRNA, and Gene informa-
tion has been anonymized and randomized.

Source

Genentech, Inc.

See Also

Please see 'vignettes/Crispr_example_workflow.R' for details.

Examples

```
data('ann')
head(ann)
```

---

ct.alignmentChart   *View a Barchart Summarizing Alignment Statistics for a Crispr Screen*

Description

This function displays the alignemnt statistics for a pooled Crispr screen, reported directly from an
alignment statistic matrix.

Usage

```
ct.alignmentChart(aln, sampleKey = NULL)
```

Arguments

aln   A numeric matrix of alignment statistics for a Crispr experiment. Corresponds
               to a 4xN matrix of read counts, with columns indicating samples and rows in-
               dicating the number of 'targets', 'nomatch', 'rejections', and 'double_match'
               reads. Details about these classes may be found in the best practices vignette or
               as part of the report generated with `ct.makeReport()`.

sampleKey   An optional ordered factor linking the samples to experimental variables. The
               `names` attribute should exactly match those present in `aln`.

Value

A grouped barplot displaying the alignment statistics for each sample included in the alignment
matrix, which usually corresponds to all of the samples in the experiment.

Author(s)

Russell Bainer

Examples

```
data('aln')
ct.alignmentChart(aln)
```

---

ct.applyAlpha   *Apply RRA 'alpha' cutoff to RRAalpha input*

Description

The 'alpha' part of RRAalpha is used to consider only the top guide-level scores for gene-level statistics. Practically, all guides failing the cutoff get a pvalue of 1. There are three ways of determining which guides fail. See 'scoring' below.

Usage

```
ct.applyAlpha(
  stats,
  RRAalphaCutoff = 0.1,
  scoring = c("combined", "pvalue", "fc")
)
```

Arguments

stats	three-column numeric matrix with pvalues for down and up one-sided test with guide-level fold changes (coefficients from the relevant contrast).
RRAalphaCutoff	A cutoff to use when defining gRNAs with significantly altered abundance during the RRAa aggregation step, which may be specified as a single numeric value on the unit interval or as a logical vector. When supplied as a logical vector (of length equal to `nrows(fit)`), this parameter directly indicates the gRNAs to include during RRAa aggregation. Otherwise, if `scoring` is set to `pvalue` or `combined`, this parameter is interpreted as the maximum nominal p-value required to consider a gRNA's abundance meaningfully altered during the aggregation step. If `scoring` is `fc`, this parameter is interpreted as the proportion of the list to be considered meaningfully altered in the experiment (e.g., if `RRAalphaCutoff` is set to 0.05, only consider the rankings of the 5 (or down-regulated) gRNAs for the purposes of RRAa calculations).
scoring	The gRNA ranking method to use in RRAa aggregation. May take one of three values: `pvalue`, `fc`, or 'combined'. `pvalue` indicates that the gRNA ranking statistic should be created from the (one-sided) p-values in the fit object. `fc` indicates that the ranks of the gRNA coefficients should be used instead, and `combined` indicates that that the coefficents should be used as the ranking statistic but gRNAs are discarded in the aggregation step based on the corresponding nominal p-value in the fit object.

Value

data.frame with guide-level pvals, fold change, and scores.deplete and scores.enrich which are the input the RRAalpha

Author(s)

Russell Bainer

Examples

```
fakestats <- matrix(runif(300), ncol = 3)
colnames(fakestats) = c('Depletion.P', 'Enrichment.P', 'lfc')
ct.applyAlpha(fakestats)
```

---

ct.buildSE   *Package Screen Data into a 'SummarizedExperiment' Object*

Description

Convenience function to package major components of a screen into a 'SummarizedExperiment' container for downstream visualization and analysis. All arguments are optional except for 'es'.

Usage

```
ct.buildSE(
  es,
  sampleKey = NULL,
  ann = NULL,
  vm = NULL,
  fit = NULL,
  summaryList = NULL
)
```

Arguments

es	An 'ExpressionSet' of screen data. Required.
sampleKey	a gCrisprTools 'sampleKey' object, to be added to the 'colData'.
ann	Annotation object to be packaged into the 'rowData'
vm	A 'voom'-derived normalized object
fit	a 'MArrayLM' object containing the contrast information and model results
summaryList	A named list of data.frames, returned by ct.generateResults. if you need to generate one of these by hand for some reason, see the example resultsDF object loaded in the example below.

Value

A 'SummarizedExperiment' object.

Author(s)

Russell Bainer

Examples

```
data('ann', 'es', 'fit', 'resultsDF')
ct.buildSE(es, ann = ann, fit = 'fit', summaryList = list('resA' = resultsDF, 'resB' = resultsDF))
```

---

ct.CAT   *Compare Two CRISPR Screens via a CAT plot*

Description

This is a function for comparing the results of two screening experiments. Given two summaryDF, the function places them in register with one another, generates a Concordance At The Top (CAT) plot, and returns an invisible dataframe containing the relevant gene-level signals. Signals are aggregated by P-value threshold, such that the concordance is represented as the portion of shared values meeting or exceeding that significance threshold.

This function is conceptually similar to the 'ct.ROC' and 'ct.PRC()' functions, but is appropriate when considering consistency of ranked values rather than an interchangeable set; the most common use case is for comparing primary and replication screens, where the underlying technology and selection criteria are expected to be highly similar. CAT plots are fundamentally about comparing rankings, and so only targets in common between the two provided screens are considered. If the totality of list overlap is important, consider using 'ct.PRC()' or 'ct.ROC()'.

Usage

```
ct.CAT(
  dflist,
  targets = c("geneSymbol", "geneID"),
  switch.dir = FALSE,
  plot.it = TRUE
)
```

Arguments

dflist	A list of results dataframes. Names will be preserved, and the enrichment calculation is conditioned on the first element of the list.
targets	Column of the provided summaryDF to consider. Must be geneID or geneSymbol.
switch.dir	Logical indicating whether to test overlap of signals in the same direction, or whether the directionality is expected to reverse. 'same.dir = FALSE' looks at the consistency between depleted signals in 'df1' and enriched signals in 'df2'.
plot.it	Logical indicating whether to compose the plots on the default device. Two CAT plots summarizing overlap in both enrichment directions are drawn.

Value

Invisibly, a data.frame containing the relevant summary stats for each target in both screens.

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
cat <- ct.CAT(list('first' = resultsDF, 'second' = resultsDF[1:2000,]))
head(cat)
```

---

ct.compareContrasts   *Identify Replicated Signals in Pooled Screens Using Conditional Scoring*

Description

This function identifies signals that are present in one or more screening experiment contrasts using a conditional strategy. Specifically, this function identifies all significant signals (according to user definitions) in a set of provided results DF and returns a 'simplifiedResult' dataframe derived from the first provided contrast with an appended logical column indicating whether there is evidence for signal replication in the other provided resultsDFs.

Signals are considered replicated if they cross the specified stringent threshold (default: $Q = 0.1$) in one or more of the provided contrasts, and are similarly enriched or depleted at the relaxed threshold (default: $P = 0.1$) in all of the remaining contrasts. If a single contrast is provided, all signals crossing the stringent threshold are considered replicated.

Signals are compared across screens on the basis of ct.regularizeContrasts, so users must provide an identifier with which to standardize targets ('geneID' by default).

Usage

```
ct.compareContrasts(
  dflist,
  statistics = c("best.q", "best.p"),
  cutoffs = c(0.1, 0.1),
  same.dir = rep(TRUE, length(dflist)),
  return.stats = FALSE,
  nperm = 10000,
  ...
)
```

Arguments

dflist	A list of (possibly simplified) results data.frames produced by [ct.generateResults]().
statistics	Statistics to use to define congruence; may be a single value, but internally coerced to a vector of length 2 where the first value corresponds to the stringent cutoff annd the second value is used for the relaxed cutoff. Must be 'best.p' or 'best.q'.
cutoffs	Numeric value(s) corresponding to the significance cutoff(s) used to define stringent and relaxed values of 'statistics'. Internally coerced to a vector of length 2.
same.dir	Logical vector of the same length as 'dflist' indicating whether replicating signals are expected to go in the same direction (e.g., enrich/deplete in their respective screens). For example, a 'dflist' of length 3 could be specified as c(TRUE, TRUE, FALSE), indicating that replicating signals should be enriched in both of the first two contrasts and depleted in the third to be considered replicated (or vise-versa). Default is 'rep(TRUE, length(dflist))'.
return.stats	When TRUE, return the significance of overlap instead of the logical vector (by permutation).
nperm	numeric indicating number of permutations when 'return.stats' is true (default 10000).
...	Other arguments to 'ct.simpleResult()', especially 'collapse'.

Value

If 'return.stats' is 'FALSE', returns the first contrast as a 'simplifiedResult' data.frame, with a 'replicated' logical column indicating whether each signal replicates in all of the provided screens according to the specified logic.

If 'return.stats' is 'TRUE', returns a dataframe indicating the permutation-based test statistics summarizing the evidence for significantly enriched signal replication across the provided contrasts (enrich, deplete, and all together).

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
summary(ct.compareContrasts(list(resultsDF, resultsDF[1:5000,]))$replicated)
ct.compareContrasts(list(resultsDF, resultsDF[1:5000,]), return.stats = TRUE)
```

---

ct.contrastBarchart   *Visualize Signal Across A List of Contrasts*

Description

Given a list of provided results 'data.frame's summarizing a series of contrasts from one or more pooled screens, this function visualizes their respective signals as a series of stacked barcharts. Enriched signals are represented in the positive direction, and depleted signals are represented in the negative direction. Note that the provided contrast results are not regularized by this function.

This function may be used to compare signals across different screen contrasts, or to compare signals within interesting subsets of targets ascertained within a single experiment.

Usage

```
ct.contrastBarchart(
  dflist,
  background = TRUE,
  statistic = c("best.q", "best.p"),
  ...
)
```

Arguments

dflist	A named list of 'data.frame's summarizing the results of one or more screen contrasts, returned by the function `ct.generateResults`.
background	Logical indicating whether to represent the nonsignificant hits in the barchart.
statistic	Should cutoffs be calculated based on FDR ('best.q') or P-value ('best.p')?
...	Other parameters to lower functions, especially 'ct.simpleResult()'

Value

A summary plot on the current device. Invisibly, the data.frame tallying signals at various thresholds.

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
ct.contrastBarchart(list('FirstResult' = resultsDF, 'SecondResult' = resultsDF))
ct.contrastBarchart(list('FirstResult' = resultsDF, 'SecondResult' = resultsDF), background = FALSE)
ct.contrastBarchart(list('FirstResult' = resultsDF[1:1000,], 'SecondResult' = resultsDF))
```

---

ct.DirectionalTests   *Compute Directional P-values from eBayes Output*

Description

This function produces two sets of one-sided P-values derived from the moderated t-statistics produced by eBayes.

Usage

```
ct.DirectionalTests(fit, contrast.term = NULL)
```

Arguments

fit   An object of class MArrayLM containing, at minimum, a df.residual slot containing the appropriate degres of freedom for each test, and a t slot containing t statistics.

contrast.term   If a fit object with multiple coefficients is passed in, a string indiating the coefficient of interest.

Value

A matrix object with two numeric columns, indicating the p-values quantifying the evidence for enrichment and depletion of each feature in the relevant model contrast.

Author(s)

Russell Bainer

Examples

```
data('fit')
ct.DirectionalTests(fit)
```

---

ct.expandAnnotation   *Expand an annotation object to allow annotations of reagents to multiple targets*

Description

This function takes a gCrisprTools annotation object and expands it to allow 1:many mappings of reagents. This mostly is used for internal processing, and users should interact with the wrapper functions that call it (e.g., 'ct.generateResults').

Libraries targeting ambiguous biological elements (e.g., alternative promoters to a gene where the boundaries between elements is contested) may contain reagents that are plausibly annotated to a finite set of possible targets. To accomodate this, users may supply an alternative reagent annotation in the form of a named list of vectors, where the names correspond to reagent 'ID's in the annotation object and each list element corresponds something coercible to a to a character vector of associated targets that will ultimately be assembled into the 'geneSymbol' column of the annotation object. It is assumed that the 'geneID' values are assigned unambiguously to the reagents, and are passed through directly.

Usage

```
ct.expandAnnotation(ann, alt.annotation)
```

Arguments

ann   A `data.frame` containing an annotation object with gRNA-level information
                 encoded as rows, typically produced by 'ct.prepareAnnotation'. The 'ID' col-
                 umn should correspond to the individual reagent identifiers.

alt.annotation   A named list of character vectors, which should be named identically to a value
                 in the 'ID' column of the supplied annotation object. The values in the character
                 vectors will eventually form the 'geneSymbol' column of the annotation file.

Value

A new annotation data frame, expanded as described above.

Author(s)

Russell Bainer

Examples

```
data('ann')
alt.annotation <- list('Target2089_b' = c('foo', 'bar'), 'Target2089_c' = 'foo', 'Target2089_a' = 'bar')
ct.expandAnnotation(ann, alt.annotation)
```

---

ct.filterReads *Remove low-abundance elements from an ExpressionSet object*

Description

This function removes gRNAs only present in very low abundance across all samples of a pooled Crispr screening experiment. In most cases very low-abundance guides are the result of low-level contamination from other libraries, and often distort standard normalization approaches. This function trims gRNAs in a largely heuristic way, assuming that the majority of 'real' gRNAs within the library are comparably abundant in at least some of the samples (such as unexpanded controls), and that contaminants are present at negligible levels. Specifically, the function trims the trim most abundant guides from the upper tail of each log-transformed sample distribution, and then omits gRNAs whose abundances are always less than $1/(2^{log2.ratio})$ of this value.

Usage

```
ct.filterReads(
  eset,
  trim = 1000,
  log2.ratio = 4,
  sampleKey = NULL,
  plot.it = TRUE,
  read.floor = NULL
)
```

Arguments

eset	An unnormalized ExpressionSet object containing, at minimum, a matrix of gRNA counts accessible with exprs().
trim	The number of gRNAs to be trimmed from the top of the distribution before estimating the abundance range. Empirically, this usually should be equal to about 2 to 5 percent of the guides in the library.
log2.ratio	Maximum abundance of contaminant gRNAs, expressed on the log2 scale from the top of the trimmed range of each sample. That is, log2.ratio = 4 means to discard all gRNAs whose abundance is $(1/2)^4$ of the trimmed maximum.
sampleKey	An (optional) sample key, supplied as an ordered factor linking the samples to experimental variables. The names attribute should exactly match those present in eset, and the control set is assumed to be the first level.
plot.it	Logical value indicating whether to plot the adjusted gRNA densities on the default device.
read.floor	Optionally, the minimum number of reads required for each gRNA.

Value

An ExpressionSet object, with trace-abundance gRNAs omitted.

Author(s)

Russell Bainer

Examples

```
data('es')
ct.filterReads(es)
```

---

ct.GCbias   *Visualization of gRNA GC Content Trends*

Description

This function visualizes relationships between gRNA GC content and their measured abundance or various differential expression model estimates.

Usage

```
ct.GCbias(data.obj, ann, sampleKey = NULL, lib.size = NULL)
```

Arguments

data.obj   An ExpressionSet or fit (MArrayLM) object to be analyzed for the presence of GC content bias.

ann   An annotation data.frame, used to estimate GC content for each guide. Guides are annotated by row, and the object must minimally contain a target column containing a character vector that indicates the corresponding nucleotide sequences.

sampleKey   An optional sample key, supplied as a factor linking the samples to experimental variables. The names attribute should exactly match those present in eset, and the control set is assumed to be the first level. Ignored in the analysis of model fit objects.

lib.size   An optional vector of voom-appropriate library size adjustment factors, usually calculated with [calcNormFactors](#) and transformed to reflect the appropriate library size. These adjustment factors are interpreted as the total library sizes for each sample, and if absent will be extrapolated from the columnwise count sums of the exprs slot of the eset.

Value

An image relating GC content to experimental observations on the default device. If the provided data.obj is an ExpressionSet, this takes the form of a scatter plot where the GC with a smoothed trendline within each sample. If data.obj is a fit object describing a linear model contrast, then four panels are returned describing the GC content distribution and its relationship to guide-level fold change, standard deviation, and P-value estimates.

Author(s)

Russell Bainer

Examples

```
data('es')
data('ann')
data('fit')

ct.GCbias(es, ann)
ct.GCbias(fit, ann)
```

---

ct.generateResults   *Calculate results of a crispr screen from a contrast*

Description

This is a wrapper function that enables direct generation of target-level p-values from a crispr screen.

Usage

```
ct.generateResults(
  fit,
  annotation,
  RRAalphaCutoff = 0.1,
  permutations = 1000,
  contrast.term = NULL,
  scoring = c("combined", "pvalue", "fc"),
  alt.annotation = NULL,
  permutation.seed = NULL
)
```

Arguments

fit   An object of class MArrayLM containing, at minimum, a t slot with t-statistics
                from the comparison, a df.residual slot with the corresponding residuals fo
                the model fits, and an Amean slot with the respective mean abundances.

annotation   An annotation file for the experiment. gRNAs are annotated by row, and must
                minimally contain columns geneSymbol and geneID.

RRAalphaCutoff A cutoff to use when defining gRNAs with significantly altered abundance dur-
                ing the RRAa aggregation step, which may be specified as a single numeric
                value on the unit interval or as a logical vector. When supplied as a logical
                vector (of length equal to nrows(fit)), this parameter directly indicates the
                gRNAs to include during RRAa aggregation. Otherwise, if scoring is set to
                pvalue or combined, this parameter is interpreted as the maximum nominal p-
                value required to consider a gRNA's abundance meaningfully altered during the

aggregation step. If scoring is fc, this parameter is interpreted as the proportion of the list to be considered meaningfully altered in the experiment (e.g., if RRAalphaCutoff is set to 0.05, only consider the rankings of the 5 (or downregulated) gRNAs for the purposes of RRAa calculations).

Note that this function uses directional tests to identify enriched or depleted targets, and when RRAalphaCutoff is provided as a logical vector the interpretation of the various aggregation statistics is going to be dependent on the specific criteria used to select reagents for inclusion.

permutations   The number of permutations to use during the RRAa aggregation step.

contrast.term   If a fit object with multiple coefficients is passed in, a string indiating the coefficient of interest.

scoring   The gRNA ranking method to use in RRAa aggregation. May take one of three values: pvalue, fc, or 'combined'. pvalue indicates that the gRNA ranking statistic should be created from the (one-sided) p-values in the fit object. fc indicates that the ranks of the gRNA coefficients should be used instead, and combined indicates that that the coefficents should be used as the ranking statistic but gRNAs are discarded in the aggregation step based on the corresponding nominal p-value in the fit object.

alt.annotation   Libraries targeting ambiguous biological elements (e.g., alternative promoters to a gene where the boundaries between elelments is contested) may contain reagents that are plausibly annotated to a finite set of possible targets. To accomodate this, users may supply an alternative reagent annotation in the form of a named list of vectors, where each list element corresponds something coercible to a to a character vector of associated targets that will ultimately be assembled into the 'geneSymbol' column of the 'resultsDF' object. Each of these character vectors should be named identically to a row of the supplied fit object (e.g., the 'row.names'). It is assumed that the 'geneID' values are assigned unambiguously to the reagents, and are passed through directly.

permutation.seed
                 numeric seed for permutation reproducibility. Default: NULL means to not set any seed. This argument is passed through to ct.RRAaPvals.

Value

A dataframe containing gRNA-level and target-level statistics. In addition to the information present in the supplied annotation object, the returned object indicates P-values and Q-values for the depletion and enrichment of each gRNA and associated target, the median log2 fold change estimate among all gRNAs associated with the target, and Rho statistics that are calculated internally by the RRAa algorithm that may be useful in ranking targets that are considered significant at a given alpha or false discovery threshold.

A 'resultsDF' formatted dataframe containing gene-level statistics.

Author(s)

Russell Bainer

Examples

```
data('fit')
data('ann')
output <- ct.generateResults(fit, ann, permutations = 10)
head(output)
  p = seq(0, 1, length.out=20)
  fc = seq(-3, 3, length.out=20)
  fc[2] = NA
  fc[3] = -20
  stats = data.frame(
    Depletion.P=p,
    Enrichment.P=rev(p),
    fc=fc
  )
  ct.applyAlpha(stats,scoring='combined')
```

---

ct.GREATdb	*Update a gene-centric msdb object for GREAT-style enrichment analysis using a specified CRISPR annotation.*

Description

Update a gene-centric 'GeneSetDb' object for GREAT-style enrichment analysis using a specified annotation.

Often, pooled screening libraries are constructed such that the gene targets of interest are associated with variable numbers of semi-independent screen signals (associated with, e.g., sets of alternative promoters or cis regulatory units). Such an arrangement is often unavoidable but produces to complications when performing gene set enrichent analyses. This function conforms a standard 'GeneSetDb' object to appropriately consider this form of ultiple testing during ontological enrichment analyses according to the GREAT strategy outlined by [McLean et al. (2009)](https://doi.org/10.1038/nbt.1630).

Operationally, this means that genewise sets in the provided object will be translated to the corresponding 'geneSymbol' sets provided in the annotation file.

Usage

```
ct.GREATdb(
  annotation,
 gsdb = sparrow::getMSigGeneSetDb(collection = c("h", "c2"), species = "human", id.type
    = "ensembl"),
  minsize = 10,
  ...
)
```

Arguments

annotation   an annotation object returned by `ct.prepareAnnotation()`.

gsdb   A gene-centric `GeneSetDb` object to conform to the relevant peakwise dataset.

minsize   Minimum number of targets required to consider a geneset valid for analysis.

...   Additional arguments to be passed to 'ct.prepareAnnotation()'.

Value

A new `GeneSetDb` object with the features annotated genewise to pathways.

Examples

```
data(resultsDF)
data(ann)
gsdb <- ct.GREATdb(ann, gsdb = sparrow::getMSigGeneSetDb(collection = 'h', species = 'human', id.type = 'entrez'))
show(sparrow::featureIds(gsdb))
```

---

ct.gRNARankByReplicate

Visualization of Ranked gRNA Abundances by Replicate

Description

This function median scales and log2 transforms the raw gRNA count data contained in an ExpressionSet, and then plots the ordered expression values within each replicate. The curve colors are assigned based on a user- specified column of the pData contained in the ExpressionSet. Optionally, this function can plot the location of Nontargeting control guides (or any guides, really) within the distribution.

Usage

```
ct.gRNARankByReplicate(
  eset,
  sampleKey,
  annotation = NULL,
  geneSymb = NULL,
  lib.size = NULL
)
```

Arguments

eset   An ExpressionSet object containing, at minimum, count data accessible by ex-
               prs() and some phenoData.

sampleKey   A sample key, supplied as a (possibly ordered) factor linking the samples to
               experimental variables. The `names` attribute should exactly match those present
               in `eset`, and the control set is assumed to be the first `level`.

annotation   An annotation dataframe indicating the nontargeting controls in the geneID col-
               umn.

geneSymb   The geneSymbol identifier(s) in annotation that corresponds to gRNAs to be
                  plotted on the curves. If the provided value is not present in the geneSymbol,
                  nontargeting controls will be plotted instead.

lib.size   An optional vector of voom-appropriate library size adjustment factors, usually
                  calculated with [calcNormFactors](#) and transformed to reflect the appropriate
                  library size. These adjustment factors are interpreted as the total library sizes
                  for each sample, and if absent will be extrapolated from the columnwise count
                  sums of the exprs slot of the eset.

Value

A waterfall plot as specified, on the default device.

Author(s)

Russell Bainer

Examples

```
data('es')
data('ann')

#Build the sample key
library(Biobase)
sk <- ordered(relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference'))
names(sk) <- row.names(pData(es))

ct.gRNARankByReplicate(es, sk, ann, 'Target1377')
```

---

ct.guideCDF   *View CDFs of the ranked gRNAs or Targets present in a crispr screen*

Description

This function generates a plot relating the cumulative proportion of reads in each sample of a crispr
screen to the abundance rank of the underlying guides (or Targets). The purpose of this algorithm
is to detect potential distortions in the library composition that might not be properly controlled by
sample normalization (see also: ct.stackedGuides()).

Usage

```
ct.guideCDF(eset, sampleKey = NULL, plotType = "gRNA", annotation = NULL)
```

Arguments

eset	An ExpressionSet object containing, at minimum, a matrix of gRNA abundances extractable with the exprs() function.
sampleKey	An optional sample key, supplied as an ordered factor linking the samples to experimental variables. The names attribute should exactly match those present in eset, and the control set is assumed to be the first level.
plotType	A string indicating whether the individual guides should be displayed ('gRNA'), or if they should be aggregated into target-level estimates ('Target') according to the geneSymbol column in the annotation object.
annotation	An optional data.frame containing an annotation object to be used to aggregate the guides into targets. gRNAs are annotated by row, and must minimally contain a column geneSymbol indicating the target elements.

Value

A CDF plot displaying the appropriate CDF curves on the default device.

Author(s)

Russell Bainer

Examples

```
data('es')
ct.guideCDF(es)
```

---

ct.inputCheck   *Check compatibility of a sample key with a supplied object*

Description

For many gCrisprTools functions, a sample key must be provided that specifies sample mapping to experimental groups. The sample key should be provided as a single, named factor whose names exactly correspond to the 'colnames()' of the 'ExpressionSet' containing the count data to be processed (or coercible as such). By convention, the first level corresponds to the control sample group.

This function checks whether the specified sample key is of the proper format and has properties consistent with the specified object.

Usage

```
ct.inputCheck(sampleKey, object)
```

Arguments

sampleKey	A named factor, where the levels indicate the experimental replicate groups and the names match the colnames of the expression matrix contained in object. The first level should correspond to the control samples, but obviously there is no way to algorithmically control this.
object	An ExpressionSet, EList, or matrix.

Value

A logical indicating whether the objects are compatible.

Author(s)

Russell Bainer

---

ct.keyCheck	*Check compatibility of a sample key with a supplied ExpressionSet or similar object*

Description

For many gCrisprTools functions, a sample key must be provided that specifies sample mapping to experimental groups. The sample key should be provided as a single, named factor whose names exactly correspond to the 'colnames()' of the 'ExpressionSet' containing the count data to be processed (or coercible as such). By convention, the first level corresponds to the control sample group.

This function checks whether the specified sample key is of the proper format and has properties consistent with the specified object.

Usage

```
ct.keyCheck(sampleKey, object)
```

Arguments

sampleKey	A named factor, where the levels indicate the experimental replicate groups and the names match the colnames of the expression matrix contained in object. The first level should correspond to the control samples, but obviously there is no way to algorithmically control this.
object	An ExpressionSet, EList, or other matrix-like object with defined 'colnames()'.

Value

Invisibly, a properly formatted 'sampleKey'.

Author(s)

Russell Bainer

Examples

```
data('es')
library(limma)
library(Biobase)

#Build the sample key
sk <- relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference')
names(sk) <- row.names(pData(es))
ct.keyCheck(sk, es)
```

---

ct.makeContrastReport   *Generate a Contrast report from a pooled CRISPR screen*

Description

This is a function to generate an html Contrast report for a CRISPR screen, focusing on contrast-level analyses collected from other functions in `gCrisprTools`. It is designed to be used 'as-is', and analysts interested in using different functionalities of the various functions should do that outside of this wrapper script.

Usage

```
ct.makeContrastReport(
  eset,
  fit,
  sampleKey,
  results,
  annotation,
  comparison.id,
  identifier,
  contrast.subset = colnames(eset),
  outdir = NULL
)
```

Arguments

eset   An ExpressionSet object containing, at minimum, a matrix of gRNA abundances
              extractable with the exprs() function and some named phenodata extractable
              with pData().

fit   A fit object for the contrast of interest, usually generated with lmFit.

sampleKey   A sample key, supplied as an ordered factor linking the samples to experimental
              variables. The names attribute should exactly match those present in eset, and
              the control set is assumed to be the first level.

results   A data.frame summarizing the results of the screen, returned by the function
              [ct.generateResults](#).

annotation
: An annotation object for the experiment. See the man page for `ct.prepareAnnotation()` for details and example format.

comparison.id
: character with a name of the comparison.

identifier
: A character string to name the report and corresponding subdirectories. If provided, the final report will be called 'identifier.html' and will be located in a directory called `identifier` in the `outdir`. If NULL, a generic name

contrast.subset
:

    character vector containing the sample labels to be used in the analysis; all elements must be contained in the `colnames` of the specified `eset`. including the timestamp will be generated. Default: `colnames(eset)`.

outdir
: An optional character string indicating the directory in which to generate the report. If NULL, a temporary directory will be automatically generated.

Value

The path to the generated html report.

Author(s)

Russell Bainer, Dariusz Ratman

Examples

```
data('es')
data('fit')
data('ann')
data('resultsDF')

##' #Build the sample key
library(Biobase)
sk <- ordered(relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference'))
names(sk) <- row.names(pData(es))

path2report <- ct.makeContrastReport(es, fit, sk, resultsDF, ann, comparison.id = NULL, outdir = '.')
```

---

ct.makeQCReport *Generate a QC report from a pooled CRISPR screen*

Description

This is a function to generate an html QC report for a CRISPR screen, focusing on experiment-level and library-level analyses collected from other functions in `gCrisprTools`. It is designed to be used 'as-is', and analysts interested in using different functionalities of the various functions should do that outside of this wrapper script.

Usage

```
ct.makeQCReport(
  eset,
  trim,
  log2.ratio,
  sampleKey,
  annotation,
  aln,
  identifier = NULL,
  lib.size,
  geneSymb = NULL,
  outdir = NULL
)
```

Arguments

eset	An ExpressionSet object containing, at minimum, a matrix of gRNA abundances extractable with the exprs() function and some named phenodata extractable with pData().
trim	The number of gRNAs to be trimmed from the top of the distribution before estimating the abundance range. Empirically, this usually should be equal to about 2 to 5 percent of the guides in the library.
log2.ratio	Maximum abundance of contaminant gRNAs, expressed on the log2 scale from the top of the trimmed range of each sample. That is, log2.ratio = 4 means to discard all gRNAs whose abundance is (1/2)^4 of the trimmed maximum.
sampleKey	A sample key, supplied as an ordered factor linking the samples to experimental variables. The names attribute should exactly match those present in eset, and the control set is assumed to be the first level.
annotation	An annotation object for the experiment. See the man page for ct.prepareAnnotation for details and example format.
aln	A numeric alignment matrix, where rows correspond to 'targets', 'nomatch', 'rejections', and 'double_match', and where columns correspond to experimentasl samples. May be 'NULL', to suppress alignment evaluation.
identifier	A character string to name the report and corresponding subdirectories. If provided, the final report will be called 'identifier.html' and will be located in a directory called identifier. If NULL, a generic name including the timestamp will be generated.
lib.size	An optional vector of voom-appropriate library size adjustment factors, usually calculated with [calcNormFactors](#) and transformed to reflect the appropriate library size. These adjustment factors are interpreted as the total library sizes for each sample, and if absent will be extrapolated from the columnwise count sums of the exprs slot of the eset.
geneSymb	The geneSymbol identifier(s) in annotation that corresponds to gRNAs to be plotted on the curves. Passed through to [ct.gRNARankByReplicate](#), [ct.viewControls](#) and [ct.prepareAnnotation](#) (as controls argument if it's not NULL). Default NULL.

outdir An optional character string indicating the directory in which to generate the report. If NULL, a temporary directory will be automatically generated.

Value

The path to the generated html report.

Author(s)

Russell Bainer, Dariusz Ratman

Examples

```
data('es')
data('ann')
data('aln')

##' #Build the sample key
library(Biobase)
sk <- ordered(relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference'))
names(sk) <- row.names(pData(es))

path2report <- ct.makeQCReport(es, trim = 1000, log2.ratio = 0.0625, sk, ann, aln, identifier = NULL, lib.size = NUL
```

---

ct.makeReport *Generate a full experimental report from a pooled CRISPR screen*

Description

This is a function to generate an html report for a CRISPR screen, incorporating information about a specified contrast. The report contains a combination of experiment-level and contrast-specific analyses, largely collected from other functions in gCrisprTools. It is designed to be used 'as-is', and analysts interested in using different functionalities of the various functions should do that outside of this wrapper script.

Usage

```
ct.makeReport(
  fit,
  eset,
  sampleKey,
  annotation,
  results,
  aln,
  outdir = NULL,
  contrast.term = NULL,
  identifier = NULL
)
```

Arguments

fit	An object of class MArrayLM containing, at minimum, a coefficents slot with coefficients from the comparison, and a stdev.unscaled slot with the corresponding standard deviation of the coefficent estimates. The row.names attribute should ideally match that which is found in annotation, but this will be checked internally.
eset	An ExpressionSet object containing, at minimum, a matrix of gRNA abundances extractable with the exprs() function and some named phenodata extractable with pData().
sampleKey	A sample key, supplied as an ordered factor linking the samples to experimental variables. The names attribute should exactly match those present in eset, and the control set is assumed to be the first level.
annotation	An annotation object for the experiment. See the man page for ct.prepareAnnotation() for details and example format.
results	A data.frame summarizing the results of the screen, returned by the function [ct.generateResults](#).
aln	A numeric alignment matrix, where rows correspond to 'targets', 'nomatch', 'rejections', and 'double_match', and where columns correspond to experimentasl samples. Users may also pass 'NULL' to suppress evaluation of alignment.
outdir	A directory in which to generate the report; if NULL, a temporary directory will be automatically generated. The report will be located in a subdirectory whose name is internally generated (see below). The path to the report itself is returned by the function.
contrast.term	A parameter passed to ct.preprocessFit in the event that the fit object contains data from multiple contrasts. See that man page for further details.
identifier	A character string to name the report and corresponding subdirectories. If provided, the final report will be called 'identifier.html' and will be located in a directory called identifier in the outdir. If NULL, a generic name including the timestamp will be generated.

Value

The path to the generated html report.

Author(s)

Russell Bainer

Examples

```
data('fit')
data('es')

##' #Build the sample key
library(Biobase)
sk <- relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference')
names(sk) <- row.names(pData(es))
```

```
data('ann')
data('resultsDF')
data('aln')
path2report <- ct.makeReport(fit, es, sk, ann, resultsDF, aln, outdir = '.')
```

---

ct.makeRhoNull *Make null distribution for RRAalpha tests*

Description

Makes random distribution of Rho value by taking nperm random samples of n rank stats, p.

Usage

```
ct.makeRhoNull(n, p, nperm)
```

Arguments

n	single integer, number of guides per gene
p	numeric vector of rank statistics
nperm	single integer, how many random samples to take.

Value

numeric vector of Rho values

Examples

```
ct.makeRhoNull(3, 1:9, 5)
```

---

ct.normalizeBySlope *Normalize sample abundance estimates by the slope of the values in the central range*

Description

This function normalizes Crispr gRNA abundance estimates by equalizing the slopes of the middle (logged) values of the distribution across samples. Specifically, the algorithm ranks the gRNA abundance estimates within each sample and determines a relationship between rank change and gRNA within a trimmed region of the distribution via a linear fit. It then adjusts each sample such that the center of the logged abundance distribution is strictly horizontal and returns these values as median-scaled counts in the appropriate slot of the input ExpressionObject.

Usage

```
ct.normalizeBySlope(ExpressionObject, trim = 0.25, lib.size = NULL, ...)
```

Arguments

ExpressionObject

> An ExpressionSet containing, at minimum, count data accessible by exprs, or an EList object with count data in the $E slot (usually returned by [voom](#)).

trim

> The proportion to be trimmed from each end of the distributionbefore performing the linear fit; algorithm defaults to 25 fit is performed on the interquartile range.

lib.size

> An optional vector of size factor adjusted library size. Default: NULL means to use sum of column counts as a lib.size.

...

> Other arguments to be passed to ct.normalizeMedians(), if desired.

Value

A renormalized object of the same type as the provided object.

Author(s)

Russell Bainer

Examples

```
data('es')
data('ann')

#Build the sample key and library sizes for visualization
library(Biobase)
sk <- ordered(relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference'))
names(sk) <- row.names(pData(es))
ls <- colSums(exprs(es))

es.norm <- ct.normalizeBySlope(es, lib.size= ls)
ct.gRNARankByReplicate(es, sk, lib.size= ls)
ct.gRNARankByReplicate(es.norm, sk, lib.size= ls)
```

---

ct.normalizeFQ   *Apply Factored Quantile Normalization to an eset*

Description

This function applies quantile normalization to subsets of samples defined by a provided factor, correcting for library size. It does this by converting raw count values to log2 counts per million and optionally adjusting further in the usual way by dividing these values by user-specified library size factors; then this matrix is split into groups according to the provided factor that are quantile normalized, and then the groups are median scaled to each other before conversion back into raw counts. This method is best used in comparisons for long timecourse screens, where groupwise differences in growth rate cause uneven intrinsic dialation of construct distributions.

Note that this normalization strategy is not appropriate for experiments where significant distortion of the libraries is expected as a consequence of the screening strategy (e.g., strong selection screens).

Usage

```
ct.normalizeFQ(eset, sets, lib.size = NULL)
```

Arguments

eset
: An `ExpressionSet` containing, at minimum, count data accessible by `exprs`.

sets
: A character or factor object delineating which samples should be grouped together during the normalization step. Must be the same length as the number of columns in the provided eset, and cannot contain 'NA' or 'NULL' values.

lib.size
: An optional vector of voom-appropriate library size adjustment factors, usually calculated with [calcNormFactors](#) and transformed to reflect the appropriate library size. These adjustment factors are interpreted as the total library sizes for each sample, and if absent will be extrapolated from the columnwise count sums of the `exprs` slot of the `eset`.

Value

A renormalized ExpressionSet object of the same type as the provided object.

Author(s)

Russell Bainer

Examples

```
data('es')

#Build the sample key and library sizes for visualization
library(Biobase)
sk <- relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference')
names(sk) <- row.names(pData(es))
ls <- colSums(exprs(es))

es.norm <- ct.normalizeFQ(es, sets = gsub('(Death|Control)', '', pData(es)$TREATMENT_NAME), lib.size= ls)
ct.gRNARankByReplicate(es, sampleKey = sk, lib.size= ls)
ct.gRNARankByReplicate(es.norm, sampleKey = sk, lib.size= ls)
```

---

ct.normalizeGuides   *Normalize an ExpressionSet Containing a Crispr Screen*

Description

This function normalizes Crispr gRNA abundance estimates contained in an `ExpressionSet` object. Currently four normalization methods are implemented: median scaling (via `normalizeMedianValues`), slope-based normalization (via `ct.normalizeBySlope()`), scaling to the median of the nontargeting control values (via `ct.normalizeNTC()`), factored quantile normalization (via `ct.normalizeFQ()`), and spline fitting to the distribution of selected gRNAs (via `ct.normalizeSpline()`). Because of

the peculiarities of pooled Crispr screening data, these implementations may be more stable than the endogenous methods used downstream by [voom](#). See the respective man pages for further details about specific normalization approaches.

Usage

```
ct.normalizeGuides(
  eset,
  method = c("scale", "FQ", "slope", "controlScale", "controlSpline"),
  annotation = NULL,
  sampleKey = NULL,
  lib.size = NULL,
  plot.it = FALSE,
  ...
)
```

Arguments

eset	An ExpressionSet object with integer count data extractable with exprs().
method	The normalization method to use.
annotation	The annotation object for the library, required for the methods employing non-targeting controls.
sampleKey	An (optional) sample key, supplied as an ordered factor linking the samples to experimental variables. The names attribute should exactly match those present in eset, and the control set is assumed to be the first level. If 'method' = 'FQ', the sampleKey is taken as the 'sets' argument (and its format requirements are similarly relaxed; see '?ct.normalizeFC').
lib.size	An optional vector of voom-appropriate library size adjustment factors, usually calculated with [calcNormFactors](#) and transformed to reflect the appropriate library size. These adjustment factors are interpreted as the total library sizes for each sample, and if absent will be extrapolated from the columnwise count sums of the exprs slot of the eset.
plot.it	Logical indicating whether to plot the ranked log2 gRNA count distributions before and after normalization.
...	Other parameters to be passed to the individual normalization methods.

Value

A renormalized ExpressionSet. If specified, the sample level counts will be scaled so as to maintain the validity of the specified lib.size values.

Author(s)

Russell Bainer

See Also

[ct.normalizeMedians](#), [ct.normalizeBySlope](#), [ct.normalizeNTC](#), [ct.normalizeSpline](#)

Examples

```
data('es')
data('ann')

#Build the sample key as needed
library(Biobase)
sk <- ordered(relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference'))
names(sk) <- row.names(pData(es))

es.norm <- ct.normalizeGuides(es, 'scale', annotation = ann, sampleKey = sk, plot.it = TRUE)
es.norm <- ct.normalizeGuides(es, 'slope', annotation = ann, sampleKey = sk, plot.it = TRUE)
es.norm <- ct.normalizeGuides(es, 'controlScale', annotation = ann, sampleKey = sk, plot.it = TRUE, geneSymb = 'NoT
es.norm <- ct.normalizeGuides(es, 'controlSpline', annotation = ann, sampleKey = sk, plot.it = TRUE, geneSymb = 'No
```

---

ct.normalizeMedians   *Normalize sample abundance estimates by median gRNA counts*

Description

This function normalizes Crispr gRNA abundance estimates by equalizing the median gRNA abundance values after correcting for library size. It does this by converting raw count values to log2 counts per million and optionally adjusting further in the usual way by dividing these values by user-specified library size factors. THis method should be more stable than the endogenous scaling functions used in voom in th especific case of Crispr screens or other cases where the median number of observed counts may be low.

Usage

```
ct.normalizeMedians(eset, lib.size = NULL)
```

Arguments

eset   An ExpressionSet containing, at minimum, count data accessible by exprs.

lib.size   An optional vector of voom-appropriate library size adjustment factors, usually calculated with [calcNormFactors](#) and transformed to reflect the appropriate library size. These adjustment factors are interpreted as the total library sizes for each sample, and if absent will be extrapolated from the columnwise count sums of the exprs slot of the eset.

Value

A renormalized ExpressionSet object of the same type as the provided object.

Author(s)

Russell Bainer

Examples

```
data('es')

#Build the sample key and library sizes for visualization
library(Biobase)
sk <- ordered(relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference'))
names(sk) <- row.names(pData(es))
ls <- colSums(exprs(es))

es.norm <- ct.normalizeMedians(es, lib.size= ls)
ct.gRNARankByReplicate(es, sampleKey = sk, lib.size= ls)
ct.gRNARankByReplicate(es.norm, sampleKey = sk, lib.size= ls)
```

---

ct.normalizeNTC   *Normalize sample abundance estimates by the median values of non-targeting control guides*

Description

This function normalizes Crispr gRNA abundance estimates by equalizing the median abundances of the nontargeting gRNAs within each sample. The normalized values are returned as normalized counts in the 'exprs' slot of the input eset. Note that this method may be unstable if the screening library contains relatively few nontargeting gRNAs.

Usage

```
ct.normalizeNTC(eset, annotation, lib.size = NULL, geneSymb = NULL)
```

Arguments

eset   An ExpressionSet object containing, at minimum, count data accessible by exprs.

annotation   An annotation dataframe indicating the nontargeting controls in the geneID column.

lib.size   An optional vector of voom-appropriate library size adjustment factors, usually calculated with [calcNormFactors](#) and transformed to reflect the appropriate library size. These adjustment factors are interpreted as the total library sizes for each sample, and if absent will be extrapolated from the columnwise count sums of the exprs slot of the eset.

geneSymb   The geneSymbol identifier in annotation that corresponds to nontargeting gR-NAs. If absent, ct.gRNARankByReplicate will attempt to infer nontargeting guides by searching for 'no_gid' or NA in the appropriate columns via ct.prepareAnnotation().

Value

A normalized eset.

Author(s)

Russell Bainer

Examples

```
data('es')
data('ann')

#Build the sample key and library sizes for visualization
library(Biobase)
sk <- ordered(relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference'))
names(sk) <- row.names(pData(es))
ls <- colSums(exprs(es))

es.norm <- ct.normalizeNTC(es, ann, lib.size = ls, geneSymb = 'NoTarget')

ct.gRNARankByReplicate(es, sk, lib.size = ls)
ct.gRNARankByReplicate(es.norm, sk, lib.size = ls)
```

---

ct.normalizeSpline	*Normalize sample abundance estimates by a spline fit to specific shared elements*

Description

This function normalizes Crispr gRNA abundance estimates by fiting a smoothed spline to a subset of the gRNAs within each sample and then equalizing these curves across the experiment. Specifically, the algorithm ranks the gRNA abundance estimates within each sample and uses a smoothed spline to determine a relationship between the ranks of the "anchor" guides and their abundance estimates. It then adjusts the spline trends from each sample to the mean of all of the sample spline fits in a manner analogous to quantile normalization, interpolating the gRNA abundance values between the anchor points; these values are returned as normalized counts in the 'exprs' slot of the input eset.

Usage

```
ct.normalizeSpline(eset, annotation, geneSymb = NULL, lib.size = NULL)
```

Arguments

eset   An ExpressionSet object containing, at minimum, count data accessible by exprs.

annotation   An annotation dataframe indicating the nontargeting controls in the geneID column.

geneSymb   The geneSymbol identifier(s) in annotation that corresponds to the "anchor" gRNAs. If absent, the method will attempt to infer nontargeting guides by searching for 'no_gid' or NA in the appropriate columns.

lib.size   An optional vector of voom-appropriate library size adjustment factors, usually
              calculated with [calcNormFactors](#) and transformed to reflect the appropriate
              library size. These adjustment factors are interpreted as the total library sizes
              for each sample, and if absent will be extrapolated from the columnwise count
              sums of the exprs slot of the eset.

Value

A normalized eset.

Author(s)

Russell Bainer

Examples

```
data('es')
data('ann')

#Build the sample key and library sizes for visualization
library(Biobase)
sk <- (relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference'))
names(sk) <- row.names(pData(es))
ls <- colSums(exprs(es))

es.norm <- ct.normalizeSpline(es, ann, 'NoTarget', lib.size = ls)
ct.gRNARankByReplicate(es, sk, lib.size = ls)
ct.gRNARankByReplicate(es.norm, sk, lib.size = ls)
```

---

ct.parseGeneSymbol   *Parse 'geneSymbol' values to construct an alternative annotation list*

Description

This is an accessory function to 'ct.expandAnnotation()' function, which enables users to expand
annotation objects to accomodate reagent libraries where reagents are expected to impact a set of
known targets. See documentation for that function for additional details.

Often, libraries that contain multiply-targeting reagents are annotated using a structured format that
can be decomposed by regex matching. This function takes in an annotation object containing
an 'ID' column indicating the reagent ID and a 'geneSymbol' column containing the target map-
pings, and parses the target mappings according to a known annotation format. Currently supported
formats are 'cellecta' (e.g., 'TARGET_P1P2P3' indicating multiple promoters associated with a
known target), and 'underscore', where different targets are concatenated using the underscore sep-
arator (e.g., 'TARGET1_TARGET2_TARGET3').

Returns an 'alt.annotation'-type list of character vectors encoding the target mappings for each
reagent.

Usage

```
ct.parseGeneSymbol(ann, format = c("cellecta", "underscore"))
```

Arguments

ann
: A `data.frame` containing reagent-level information encoded as rows. The 'ID' column should correspond to the individual reagent identifiers, and the 'geneSymbol' column should contain target annotation strings to be parsed (both are coerced to strings). Does not, strictly speaking, need to be a proper annotation object, but one of those will work.

format
: Format of the geneSymbol column strings.

Value

A named 'alt.annotation'-type list of character vectors encoding the target mappings for each reagent

Author(s)

Russell Bainer

Examples

```
fakeann <- data.frame('ID' = LETTERS[1:4], 'geneSymbol' = c('T1_P1', 'T1_P1P2', 'T1_P2P1', 'T1_P2'))
ct.parseGeneSymbol(fakeann, 'cellecta')
ct.parseGeneSymbol(fakeann, 'underscore')
```

---

ct.PRC   *Generate a Precision-Recall Curve from a CRISPR screen*

Description

Given a set of targets of interest, this function generates a Precision Recall curve from the results of a CRISPR screen. Specifically, it orders the target elements in the screen in the specified direction, and then plots the recall rate (proportion of true targets identified) against the precision (proportion of identified targets that are true targets).

Note that ranking statistics in CRISPR screens are (usually) permutation-based, and so some granularity in the rankings is expected. This function does a little extra work to ensure that hits are counted as soon as the requisite value of the ranking statistic is reached regardless of where the gene is located within the block of equally-significant genes. Functionally, this means that the drawn curve is somewhat anticonservative in cases where the gene ranks are not well differentiated.

Usage

```
ct.PRC(
  summaryDF,
  target.list,
  direction = c("enrich", "deplete"),
  plot.it = TRUE
)
```

Arguments

`summaryDF`	A dataframe summarizing the results of the screen, returned by the function `ct.generateResults`.
`target.list`	A character vector containing the names of the targets to be tested; by default these are assumed to be 'geneID's, but specifying 'collapse=geneSymbol' enables setting on 'geneSymbol' by passing that value through to 'ct.simpleResult'.
`direction`	Direction by which to order target signals ('enrich' or 'deplete').
`plot.it`	Logical value indicating whether to plot the curves.

Value

A list containing the the x and y coordinates of the curve.

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
data('essential.genes') #Note that this is an artificial example.
pr <- ct.PRC(resultsDF, essential.genes, 'enrich')
str(pr)
```

---

ct.prepareAnnotation   *Check and optionally subset an annotation file for use in a Crispr*
                        Screen

Description

This function processes a supplied annotation object for use in a pooled screening experiment. Originally this was processed into something special, but now it essentially returns the original annotation object in which the geneSymbol column has been factorized. This is primarily used internally during a call to the `ct.generateResults()` function. Also performs some minor functionality checking, and ensures that the reagent identifiers are present as an 'ID' column (if absent, the row.names are used).

Valid annotations contain both 'geneID' and 'geneSymbol' columns. This is because there is often a distinction between the official gene that is being targeted and a coherent set of gRNAs that make up a testing cohort. For example, multiple sets of guides may target distinct promoters, exons, or other entities that are expected to produce distinct biological phenomena related to the gene that should be interpreted separately. For this reason, the 'geneID' column encodes the official gene designation (typically an ensembl or entrez gene identifier) while the 'geneSymbol' column contains a human-readable descriptor of the gRNA target (such as a gene symbol or promoter name). This mapping can be further expanded to incorporate mapping ambiguity via the 'ct.expandAnnotation()' function.

Usage

```
ct.prepareAnnotation(ann, object = NULL, controls = TRUE, throw.error = TRUE)
```

Arguments

`ann`	A `data.frame` containing an annotation object with gRNA-level information encoded as rows. The `row.names` attribute should correspond to the individual gRNAs, and it should at minimum contain columns named 'geneID' and 'geneSymbol' indicating the corresponding gRNA target gene ID and symbol, respectively.
`object`	If supplied, an object with `row.names` to be used to subset the supplied annotation frame for downstream analysis.
`controls`	The name of a value in the `geneSymbol` column of ann that corresponds to non-targeting control gRNAs. May also be supplied as a logical value, in which case the function will try to identify and format nontargeting guides.
`throw.error`	Logical indicating whether to throw an error when `controls` is `TRUE` but no nontargeting gRNAs are detected.

Value

A new annotation data frame, usually with nontargeting controls and NA values reformatted to `NoTarget` (and geneID set to `'no_gid'`), and the 'geneSymbol' column of ann factorized. If supplied with an `object`, the gRNAs not present in the `object` will be omitted.

Author(s)

Russell Bainer

Examples

```
data('ann')
data('es')
es <- ct.filterReads(es)
newann <- ct.prepareAnnotation(ann, es)
```

---

ct.rankSimple *Rank Signals in a Simplified Pooled Screen Result Object*

Description

This function takes in a supplied results data.frame, optionally transforms it into a 'simplifiedResult', and returns the ranks of the target-level signals.

Usage

```
ct.rankSimple(df, top = c("enrich", "deplete"))
```

Arguments

df   A results data.frame, in either raw or simplified form. Will be converted to simplified form if necessary.

top   Determines the directionality of the ranking. 'enrich' defines ranks from the most enriched to the most depleted target; 'deplete' does the opposite

Value

A numeric vector of ranks, with length equal to the number of rows in the simplified data.frame.

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
df.simple <- ct.simpleResult(resultsDF)
sr <- ct.rankSimple(resultsDF)
all((df.simple$best.p[sr == 1] == 0), (df.simple$direction[sr == 1] == 'enrich'))

sr <- ct.rankSimple(resultsDF, 'deplete')
all((df.simple$best.p[sr == 1] == 0), (df.simple$direction[sr == 1] == 'deplete'))
```

---

ct.rawCountDensities   *Visualization of Raw gRNA Count Densities*

Description

This function plots the per-sample densities of raw gRNA read counts on the log10 scale. The curve colors are assigned based on a user- specified sampleKey. This function is primarily useful to determine whether libraries are undersequenced (low mean raw gRNA counts), contaminated (many low-abundance gRNAs present), or if PCR artifacts may be present (subset of extremely abundant guides, multiple gRNA distribution modes). In most well-executed experiments the majority of gRNAs will form a tight distribution around some reasonably high average read count (hundreds of reads), at least among the control samples. Excessively low raw count values can compromise normalization steps and subsequent estimation of gRNA levels, especially in screens in which most gRNAs have minimal effects on cell viability.

Usage

```
ct.rawCountDensities(eset, sampleKey = NULL, lib.size = NULL)
```

Arguments

eset	An ExpressionSet object containing, at minimum, count data accessible by ex-prs() and some phenoData.
sampleKey	A sample key, supplied as a (possibly ordered) factor linking the samples to experimental variables. The names attribute should exactly match those present in eset, and the control set is assumed to be the first level.
lib.size	Optional named vector of library sizes (total reads within the library) to enable normalization

Value

A density plot as specified on the default device.

Author(s)

Russell Bainer

Examples

```
data('es')

#Build the sample key
library(Biobase)
sk <- relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference')
names(sk) <- row.names(pData(es))

ct.rawCountDensities(es, sk)
```

---

ct.regularizeContrasts

Regularize Two Screening Results Objects

Description

This function prepares multiple 'gCrisprTools' results dataframes for comparison. Specifically, it checks that all provided data frames are valid result objects, converts each to the target-wise 'simpleResult' format, removes signals that are not shared by all objects, places their rows in identical order, and then returns the simplified dataframes as a list.

This function is largely meant to be used by other gCrisprtools functions, although there are occasions when an analyst may want to call it directly. Often, it is useful to pass the 'collapse' argument to 'ct.simpleresult()' in cases where libraries and technologies differ between screens.

Usage

```
ct.regularizeContrasts(dflist, collapse = c("geneSymbol", "geneID"))
```

Arguments

`dflist`	A list of results dataframes. Names will be preserved.
`collapse`	Column of the provided resultsDFs on which to collapse values; should be 'geneSymbol' or 'geneID'.

Value

A list of the in-register 'simpleResult' objects, with length and names identical to 'dflist'.

Examples

```
data('resultsDF')
lapply(ct.regularizeContrasts(list('df1' = resultsDF[1:300,], 'df2' = resultsDF[200:400,])), nrow)
```

---

ct.resultCheck	*Determine whether a supplied object contains the results of a Pooled Screen*

Description

Many gCrisprTools functions operate on a `data.frame` of results generated by a CRISPR screen. This function takes in a supplied object and returns a logical indicating whether the object can be treated as one of these data.frames for the purposes of downstream analyses. This is largely used internally, but can be useful if a user needs to build a result object for some reason.

Usage

```
ct.resultCheck(summaryDF)
```

Arguments

`summaryDF`	A `data.frame`, usually returned by `ct.generateResults`. if you need to generate one of these by hand for some reason, see the example `resultsDF` object loaded in the example below.

Value

A logical indicating whether the object is of the appropriate format.

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
ct.resultCheck(resultsDF)
```

---

ct.ROC   *Generate a Receiver-Operator Characteristic (ROC) Curve from a*
                                CRISPR screen

Description

Given a set of targets of interest, this function generates a ROC curve and associated statistics from
the results of a CRISPR screen. Specifically, it orders the elements targeted in the screen in the
specified direction, and then plots the cumulative proportion of positive hits on the y-axis. The
corresponding vectors and Area Under the Curve (AUC) statistic are returned as a list.

Note that ranking statistics in CRISPR screens are (usually) permutation-based, and so some gran-
ularity is expected. This function does a little extra work to ensure that hits are counted as soon as
the requisite value of the ranking statistic is reached regardless of where the gene is located within
the block of equally-significant genes. Functionally, this means that the drawn curve is somewhat
anticonservative in cases where the gene ranks are not well differentiated.

Usage

```
ct.ROC(
  summaryDF,
  target.list,
  direction = c("enrich", "deplete"),
  condense = TRUE,
  plot.it = TRUE,
  ...
)
```

Arguments

summaryDF	A dataframe summarizing the results of the screen, returned by the function [ct.generateResults]().
target.list	A character vector containing the names of the targets to be tested. Only targets contained in the geneSymbol column of the provided summaryDF are considered.
direction	Direction by which to order target signals ('enrich' or 'deplete').
condense	Logical indicating whether the returned x and y coordinates should be 'con-densed', returning only the points at which the detected proportion of target.list changes. If set to FALSE, the returned x and y vectors will explicitly indicate the curve value at every position (useful for performing curve arithmetic down-stream).
plot.it	Logical value indicating whether to plot the curves.
...	Additional parameters for 'ct.simpleResult()'

Value

A list containing the the x and y coordinates of the curve, and the AUC statistic (invisibly).

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
data('essential.genes') #Note that this is an artificial example.
roc <- ct.ROC(resultsDF, essential.genes, direction = 'deplete')
str(roc)
```

---

ct.scatter   *Compare Two CRISPR Screen Contrasts via a Scatter Plot*

Description

This is a function for comparing the results of two screening experiments. Given two summaryDF, the function places them in register with one another, generates a simplified scatter plot where enrichment or depletion in each contrast is represented by the associated "signed" log10 (*P*/*Q*)-value (where enriched signals are represented in the positive direction and depleted signals are shown in the negative direction), and returns an invisible 'data.frame' containing the target X-axis and Y-axis coordinates and corresponding quadrant.

This is a target-level analysis, and some minor simplifications are introduced to screen signals for the sake of clarity. Principal among these is the decision to collapse gene signals to a single directional enrichment statistic. Target-level signals are typically aggregates of many guide-level signals, it is formally possible for targets to be both significantly enriched and significantly depleted within a single screen contrast as a result of substantially divergent reagent activity. This behavior is uncommon, however, and so targets are represented by selecting the direction of enrichment or depletion associated with the most significant (*P*/*Q*)-value. This directionality is then encoded into the X-axis and Y-axis position of the target as the sign of the signal as described above.

Usage

```
ct.scatter(
  dflist,
  targets = c("geneSymbol", "geneID"),
  statistic = c("best.p", "best.q"),
  cutoff = 0.05,
  plot.it = TRUE
)
```

Arguments

dflist   A (named) list of results dataframes, of length 2. See `ct.generateResults`.

targets   Column of the provided summaryDF to consider. Must be geneID or geneSymbol.

statistic   Statistic to plot on each axis (after -log10 transformation). Must be 'p', 'q', or 'rho'.

cutoff	significance cutoff used to define the significance quadrants (cannot be exactly zero).
plot.it	Logical indicating whether to compose the plot on the default device.

Value

Invisibly, a list of length 4 containing the genes passing significance for the respective quadrants.

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
scat <- ct.scatter(list('FirstResult' = resultsDF[100:2100,], 'SecondResult' = resultsDF[1:2000,]))
head(scat)
```

---

ct.seas   *Geneset Enrichment within a CRISPR screen using 'sparrow'*

Description

This function is a wrapper for the 'sparrow::seas()' function, which identifies differentially enriched/depleted ontological categories within the hits identified by a pooled screening experiment, given a provided 'GenseSetDb()' object and a list of results objects created by 'ct.generateResults()'. By default testing is performed using 'fgsea' and a hypergeometric test ('sparrow::ora()'), and results are returned as a 'SparrowResult' object.

This function will attempt to coerce them into inputs appropriate for the above analyses via 'ct.seasPrep()', after checking the relevant parameters within the provided 'GeneSetDb'. This is generally easier than going through the individual steps yourself, especially when the user is minimally postprocessing the contrast results in question.

Sometimes, it can be useful to directly indicate the set of targets to be included in an enrichment analysis (e.g., if you wish to expand or contract the set of active targets based on signal validation or other secondary information about the experiment). To accomodate this use case, users may include a logical column in the provided result object(s) indicating which elements should be included among the positive signals exposed to the test.

Note that many pooled libraries specifically target biased sets of genes, often focusing on genes involved in a particular pathway or encoding proteins with a shared biological property. Consequently, the enrichment results returned by this function represent the disproportionate enrichment or depletion of targets annotated to pathways *within the context of the screen*, and may or may not be informative of the underlying biology in question. This means that pathways not targeted by a library will obviously never be enriched a positive target set regardless of their biological relevance, and pathways enriched within a focused library screen are similarly expected to partially reflect the composition of the library and other confounding issues (e.g., number of targets within a pathway). Analysts should therefore use this function with care. For example, it might be unsurprising to detect pathways related to histone modification within a screen employing a crispr library primarily targeting epigenetic regulators.

Usage

```
ct.seas(dflist, gdb, active = "replicated", ...)
```

Arguments

dflist   A result object created by 'ct.generateResults()', or a named list containing
                many of them; will be passed as a list to 'ct.seasPrep()' with the associated
                '...' arguments.

gdb   A 'GenseSetDb' object containing annotations for the targets specified in 're-
                sult'.

active   Name of a column in the supplied result(s) that should be used to indicate ac-
                tive/selected targets.

...   Additional arguments to pass to 'ct.seasPrep()' or 'sparrow::seas()'.

Value

A named list of 'SparrowResults' objects.

Author(s)

Steve Lianoglou for seas; Russell Bainer for GeneSetDb processing and wrapping functions.

Examples

```
data('resultsDF')
gdb <- sparrow::getMSigGeneSetDb(collection = 'h', species = 'human', id.type = 'entrez')
ct.seas(list('longer' = resultsDF, 'shorter' = resultsDF[1:10000,]), gdb)
```

---

ct.seasPrep   *Prepare one or more resultsDF objects for analysis via Sparrow.*

Description

Take in a list of results objects and return an equivalently-named list of input 'data.frames' appropri-
ate for 'sparrow::seas()'. By construction, the relevant target unit is extracted from the 'geneSym-
bol' column of the provided results objects, which may. Note that the genewise '@logFC' slot in
the returned object will contain the appropriately-signed Z transformation of the P-value assigned
to the target. In most applications this is arguably more interpretable than e.g., the median gRNA
log2 fold change.

Usage

```
ct.seasPrep(
  dflist,
  collapse.on = c("geneID", "geneSymbol"),
  cutoff = 0.1,
  statistic = c("best.q", "best.p"),
  regularize = FALSE,
  gdb = NULL,
  active = "replicated"
)
```

Arguments

`dflist`	A list of gCrisprTools results 'data.frames' to be formatted.
`collapse.on`	Should targets be annotated as 'geneSymbol's or 'geneID's (default)?
`cutoff`	Numeric maximum value of 'statistic' to define significance.
`statistic`	Should cutoffs be calculated based on FDR ('best.q') or P-value ('best.p')?
`regularize`	Logical indicating whether to regularize the result objects in 'dflist' (e.g., use intersection set of all targets), or keep as-is.
`gdb`	Optionally, a 'GeneSetDb' object to enable proper registration of the output. If provided, the collapsing features in the provided 'simpleDF's must be present in the 'gsd@db$feature_id' slot. Note that a GREAT-style 'GeneSetDb' that has been conformed via 'ct.GREATdb()' will use 'geneID's as the 'feature_id'.
`active`	Optionally, the name of a logical column present in the provided result that will be used to define significant signals. This is set to 'replicated' by default to If a valid column name is provided, this overrides the specification of 'cutoff' and 'statistic'.

Value

A list of 'data.frames' formatted for evaluation with 'sparrow::seas()'.

Examples

```
data(resultsDF)
ct.seasPrep(list('longer' = resultsDF, 'shorter' = resultsDF[1:10000,]), collapse.on = 'geneSymbol')
```

---

ct.signalSummary	*Generate a Figure Summarizing Overall Signal for One or More Targets*

Description

Given one or more targets of interest, this function generates a summary image contextualizing the corresponding signals within the provided contrast. This takes the form of an annotated ranking curve of target-level signals, supplemented with horizontal Q-value cutoffs and an inset volcano plot of gRNA behavior.

Limited annotation is provided for the specified targets using the following logic:

- If a character vector is provided, up to five targets are annotated; longer lists are highlighted without specifying individual elements. - If a list is provided, the 'names' element is used as the annotation. This is similarly constrained to a total of 5 annotated elements.

Usage

```
ct.signalSummary(summaryDF, targets, callout = FALSE, ...)
```

Arguments

summaryDF   A dataframe summarizing the results of the screen, returned by the function
                [ct.generateResults](#).

targets   A list or character vector containing the names of the targets to be displayed.
                Only targets contained in the column specified by the 'collapse' parameter to
                'ct.simpleResult()' will be displayed; default is 'geneSymbol'. Plotting priority
                (e.g., the points to plot last in the case of overlapping signals) is given to earlier
                elements in the list.

callout   Logical indicating whether lines should be plotted indicating individual gene
                sets to augment the point highlighting.

...   Additional optional arguments to 'ct.simpleResult()'

Value

A summary plot on the current device.

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
ct.signalSummary(resultsDF, list('CandidateA' = 'Target229', 'Pathway3' = resultsDF$geneSymbol[c(42,116,1138,550
```

---

ct.simpleResult | *Convert a verbose results DF object to a gene-level result object*

Description

Convenience function to reduce a full results object to a gene-level object that retains minimal statistics (or alternatively, check that a provided simple result object is valid).

Usage

```
ct.simpleResult(summaryDF, collapse = c("geneSymbol", "geneID"))
```

Arguments

summaryDF   A data.frame, usually returned by ct.generateResults. if you need to generate one of these by hand for some reason, see the example resultsDF object loaded in the example below.

collapse   Column of the provided resultsDF on which to collapse values; in most cases this should be 'geneSymbol' or 'geneID'.

Value

A gene-level 'data.frame', with guide-level information omitted

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
ct.simpleResult(resultsDF)
```

---

ct.softLog | *Log10 transform empirical P-values with a pseudocount*

Description

This function -log10 transforms empirical P-values by adding a pseudocount of 1/2 the minimum nonzero value.

Usage

```
ct.softLog(x)
```

Arguments

x   numeric vector.

Value

-log10-transformed version of X.

Examples

```
ct.softLog(runif(20))
```

---

ct.stackGuides   *View a stacked representation of the most variable targets or individ-*
                        ual guides within an experiment, as a percentage of the total aligned
                        reads

Description

This function identifies the gRNAs or targets that change the most from sample to sample within an experiment as a percentage of the entire library. It then plots the abundance of the top nguides as a stacked barplot for all samples in the experiment. The purpose of this algorithm is to detect potential distortions in the library composition that might not be properly controlled by sample normalization, and so the most variable entites are defined by calculating the percent of aligned reads that they contribute to each sample, and then ranking each entity by the range of these percentages across all samples. Consequently, gRNAs or Targets that are highly abundant in at least one condition will be are more likely to be identified.

Usage

```
ct.stackGuides(
  eset,
  sampleKey = NULL,
  nguides = 20,
  plotType = "gRNA",
  annotation = NULL,
  ylimit = NULL,
  subset = NULL
)
```

Arguments

eset   An ExpressionSet object containing, at minimum, a matrix of gRNA abundances
              extractable with the exprs() function, and a metadata object containing a col-
              umn named SAMPLE_LABEL containing unique identifers for each sample. The
              colnames should be syntactically

sampleKey   An optional sample key, supplied as an ordered factor linking the samples to
              experimental variables. The names attribute should exactly match those present
              in eset, and the control set is assumed to be the first level.

nguides	The number of guides (or targets) to display.
plotType	A string indicating whether the individual guides should be displayed ('gRNA'), or if they should be aggregated into target-level estimates ('Target') according to the geneSymbol column in the annotation object.
annotation	An optional data.frame containing an annotation object to be used to aggregate the guides into targets. gRNAs are annotated by row, and must minimally contain a column geneSymbol indicating the target elements.
ylimit	An optional numeric vector of length 2 specifying the y limits for the plot, useful in comparin across studies.
subset	An optional character vector containing the sample labels to be used in the analysis; all elements must be contained in the colnames of the specified eset.

Value

A stacked barplot displaying the appropriate entities on the default device.

Author(s)

Russell Bainer

Examples

```
data('es')
data('ann')
ct.stackGuides(es, nguides = 20, plotType = 'Target', annotation = ann, ylimit = NULL, subset = NULL)
```

---

ct.topTargets	*Display the log2 fold change estimates and associated standard deviations of the guides targeting the top candidates in a crispr screen*

Description

This is a function for displaying candidates from a crispr screen, using the information summarized in the corresponding fit and the output from ct.generateResults(). The fold change and standard deviation estimates for each gRNA associated with each target (extracted from the coefficients and stdev.unscaled slot of fit) are plotted on the y axis. Targets are selected on the basis of their gene-level enrichment or depletion P-values; in the case of ties, they are ranked on the basis of their corresponding Rho statistics.

Usage

```
ct.topTargets(
  fit,
  summaryDF,
  annotation,
  targets = 10,
  enrich = TRUE,
  contrast.term = NULL
)
```

Arguments

fit	An object of class MArrayLM containing, at minimum, a coefficents slot with coefficients from the comparison, and a stdev.unscaled slot with the corresponding standard deviation of the coeffecient estimates. The row.names attribute should ideally match that which is found in annotation.
summaryDF	A data.frame summarizing the results of the screen, returned by the function ct.generateResults.
annotation	An annotation object for the experiment. gRNAs are annotated by row, and must minimally contain a column geneSymbol.
targets	Either the number of top targets to display, or a list of geneSymbols contained in the geneSymbol slot of the annotation object.
enrich	Logical indicating whether to display guides that are enriched (default) or depleted within the screen. If a vector of geneSymbols is specified, this controls the left-t0-right ordering of the corresponding gRNAs.
contrast.term	If a fit object with multiple coefficients is passed in, a string indiating the coefficient of interest.

Value

An image on the default device indicating each gRNA's log2 fold change and the unscaled standard deviation of the effect estimate, derived from the MArrayLM object.

Author(s)

Russell Bainer

Examples

```
data('fit')
data('resultsDF')
data('ann')

ct.topTargets(fit, resultsDF, ann)
```

---

ct.upSet   *Consolidate shared signals across many contrasts in an UpSet Plot*

Description

This function takes in a named list of 'results' dataframes produced by 'ct.generateResults()' or similar, harmonizes them, and identifies overlaps between them using the logic implemented in 'ct.compareContrasts()'. It then uses the overlaps of these sets to compose an UpSet plot summarizing shared overlaps of the provided contrasts. These overlaps can be specified with some detail via arguments passed to the 'ct.compareContrasts()' function; see documentation for more details.

Note that the UpSet plot is constructed to respect signal directionality, and by default constructs overlaps conditionally, but in a *bidirectional* manner. That is, a signal is considered observed in

two (or more) contrasts regardless of the contrast from which the stringent signal is observed, so a signal replicated in three contrasts is interpreted as a target for which the evidence crosses the stringent threshold in one or more of the contrasts and passes the lax contrast in the others.

Note that multiple important parameters are passed directly to 'ct.compareContrasts()' if not specified in the command. Users are advised to study the corresponding manual page to better understand their options regarding contrast thresholding, orientation, etc.

Usage

```
ct.upSet(dflist, add.stats = TRUE, nperm = 10000, ...)
```

Arguments

dflist	a named list of (possibly simplified) 'resultsDf's.
add.stats	Logical indicating whether the significance of set overlaps should be included in the visualization.
nperm	Number of permutations for P-value generation. Ignored if 'add.stats' is 'FALSE'.
...	Other named arguments to 'ComplexHeatmap::UpSet()', 'ct.compareContrasts', or 'ct.simpleResult()'.

Value

An UpSet plot on the current device. Silently, a combination matrix appropriate for plotting that plot, containing useful information about the observed intersections.

Author(s)

Russell Bainer

Examples

```
data('resultsDF')
sets <- ct.upSet(list('first' = resultsDF, 'second' = resultsDF[1:5000,]))
```

---

ct.viewControls   *View nontargeting guides within an experiment*

Description

This function tries to identify, and then plot the abundance of, the full set of non-targeting controls from an ExpressionSet object. Ideally, the user will supply a geneSymbol present in the appropriate annotation file that uniquely identifies the nontargeting gRNAs. Absent this, the the function will search for common identifier used by nontargeting controls (geneID 'no_gid', or geneSymbol NA).

Usage

```
ct.viewControls(
  eset,
  annotation,
  sampleKey,
  geneSymb = NULL,
  normalize = TRUE,
  lib.size = NULL
)
```

Arguments

eset   An ExpressionSet object containing, at minimum, a matrix of gRNA abundances
                extractable with the exprs function.

annotation   An annotation data.frame for the experiment. gRNAs are annotated by row, and
                must minimally contain columns geneSymbol and geneID.

sampleKey   A sample key, supplied as an ordered factor linking the samples to experimental
                variables. The names attribute should exactly match those present in eset, and
                the control condition is assumed to be the first level.

geneSymb   The geneSymbol identifier in annotation that corresponds to nontargeting gR-
                NAs. If absent, ct.ViewControls will attempt to infer nontargeting guides by
                searching for 'no_gid' or NA in the appropriate columns.

normalize   Logical indicating whether to attempt to normalize the data in the eset by DE-
                Seq size factors present in the metadata. If TRUE, then the metadata must contain
                a column containing these factors, named sizeFactor.crispr-gRNA.

lib.size   An optional vector of voom-appropriate library size adjustment factors, usually
                calculated with [calcNormFactors](#) and transformed to reflect the appropriate
                library size. These adjustment factors are interpreted as the total library sizes
                for each sample, and if absent will be extrapolated from the columnwise count
                sums of the exprs slot of the eset.

Value

An image of nontargeting control gRNA abundances on the default device.

Author(s)

Russell Bainer

Examples

```
data('es')
data('ann')

#Build the sample key
library(Biobase)
sk <- ordered(relevel(as.factor(pData(es)$TREATMENT_NAME), 'ControlReference'))
names(sk) <- row.names(pData(es))
```

```
ct.viewControls(es, ann, sk, geneSymb = NULL, normalize = FALSE)
ct.viewControls(es, ann, sk, geneSymb = NULL, normalize = TRUE)
```

---

ct.viewGuides *Generate a Plot of individual gRNA Pair Data in a Crispr Screen*

Description

This function generates a visualization of the effect estimates from a MArrayLM model result for all of the individual guides targeting a particular element, specified somewhere in the library annotation file. The estimated effect size and variance is plotted relative to zero for the specified contrast, with the color of the dot indicating the relative scale of the of the guide intercept within the model framework, with warmer colors indicating lowly expressed guides. For comparison, the density of gRNA fold change estimates is privided in a pane on the right, with white lines indicating the exact levels of the individual guides.

Usage

```
ct.viewGuides(
  gene,
  fit,
  ann,
  type = "geneSymbol",
  contrast.term = NULL,
  ylims = NULL
)
```

Arguments

gene	the name of the target element of interest, contained within the 'type' column of the annotation file.
fit	An object of class MArrayLM containing, at minimum, an 'Amean' slot containing the guide level abundances, a 'coefficients' slot containing the effect estimates for each guide, and an 'stdev.unscaled' slot giving the coefficient standard Deviations.
ann	A data.frame object containing the gRNA annotations. At mimimum, it should have a column with the name specified by the type argument, containing the element targeted by each guide.
type	A character string indicating the column in ann containing the target of interest.
contrast.term	If a fit object with multiple coefficients is passed in, a string indiating the coefficient of interest.
ylims	An optional numeric vector of length 2 indicating the extremes of the y-axis scale.

Value

An image summarizing gRNA behavior within the specifed gene on the default device.

Author(s)

Russell Bainer

Examples

```
data('fit')
data('ann')
ct.viewGuides('Target1633', fit, ann)
```

---

es   *ExpressionSet of count data from a Crispr screen with strong selection*

Description

Expressionset of raw counts from a screen in mouse cells performed at Genentech, Inc. All sample, gRNA, and Gene information has been anonymized and randomized.

Source

Genentech, Inc.

See Also

Please see 'vignettes/Crispr_example_workflow.R' for details.

Examples

```
data('es')
print(es)
```

---

essential.genes   *Artificial list of 'essential' genes in the example Crispr screen included for plotting purposes*

Description

Example gene list, designed to demonstrate functions using gene lists. All sample, gRNA, and Gene information has been anonymized and randomized.

Source

Russell Bainer

See Also

Please see 'vignettes/Crispr_example_workflow.R' for details.

Examples

```
data('essential.genes')
essential.genes
```

---

fit   *Precalculated contrast fit from a Crispr screen*

Description

A precalculated fit object (class MArrayLM) comparing the death and control expansion arms of a crispr screen performed at Genentech, Inc. All sample, gRNA, and Gene information has been anonymized and randomized.

Source

Genentech, Inc.

See Also

Please see 'vignettes/Crispr_example_workflow.R' for model details.

Examples

```
data('fit')
show(fit)
```

---

resultsDF   *Precalculated gene-level summary of a crispr screen*

Description

A precalculated summary Dataframe comparing the death and control expansion arms of the provided example Crispr screen (using 8 cores, seed = 2). All sample, gRNA, and Gene information has been anonymized and randomized.

Source

Genentech, Inc.

See Also

Please see 'vignettes/Crispr_example_workflow.R' for model details.

Examples

```
data('resultsDF')
head(resultsDF)
```

Index