Package 'NanoStringNCTools'

May 4, 2026

Title NanoString nCounter Tools

Description Tools for NanoString Technologies nCounter Technology. Provides support for reading RCC files into an ExpressionSet
derived object. Also includes methods for QC and normalizaztion of NanoString data.

Version 1.21.0

Encoding UTF-8

Depends R (>= 3.6), Biobase, S4Vectors, ggplot2

Imports BiocGenerics, Biostrings, ggbeeswarm, ggiraph, ggthemes, grDevices, IRanges, methods, pheatmap, RColorBrewer, stats, utils

Suggests biovizBase, ggbio, RUnit, rmarkdown, knitr, qpdf

License MIT

Collate SignatureSet-class.R RccMetadata.R NanoStringRccSet-class.R NanoStringRccSet-validity.R NanoStringRccSet-accessors.R NanoStringRccSet-signatures.R NanoStringRccSet-subset.R NanoStringRccSet-utils.R NanoStringRccSet-summary.R NanoStringRccSet-qc.R NanoStringRccSet-normalize.R NanoStringRccSet-munge.R NanoStringRccSet-ggplot.R NanoStringRccSet-autoplot.R readRccFile.R readRlfFile.R readNanoStringRccSet.R writeNanoStringRccSet.R utils.R

biocViews GeneExpression, Transcription, CellBasedAssays, DataImport, Transcriptomics, Proteomics, mRNAMicroarray, ProprietaryPlatforms, RNASeq

VignetteEngine knitr

VignetteBuilder knitr

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

git_branch devel

git_last_commit 902fffc

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-03

Author Patrick Aboyoun [aut],
Nicole Ortogero [aut],
Maddy Griswold [cre],
Zhi Yang [ctb]

Maintainer Maddy Griswold <mgriswold@nanostring.com>

Contents

`geom_beeswarm_interactive`

Geometry for Interactive Bee Swarm Points

Description

The interactive version of `geom_beeswarm` from **ggbeeswarm**.

Usage

```
geom_beeswarm_interactive(mapping = NULL, data = NULL,
                          priority = c("ascending", "descending", "density",
                                       "random", "none"),
                          cex = 1, groupOnX = NULL, dodge.width = 0,
                          stat = "identity", na.rm = FALSE, show.legend = NA,
                          inherit.aes = TRUE, ...)
```

Arguments

`mapping`	The aesthetic mapping. See `geom_beeswarm`.
`data`	The data to be displayed at this layer. See `geom_beeswarm`.
`priority`	Method used to perform point layout. See `geom_beeswarm`.
`cex`	Scaling for adjusting point spacing. See `geom_beeswarm`.
`groupOnX`	Indicator for jittering on x-axis. See `geom_beeswarm`.
`dodge.width`	Dodge amount for points from different aesthetic groups. See `geom_beeswarm`.
`stat`	The statistical transformation to use on the data for this layer. See `geom_beeswarm`.
`na.rm`	Indicator for removing missing values with a warning. See `geom_beeswarm`.
`show.legend`	Indicator for including this layer in the legend. See `geom_beeswarm`.
`inherit.aes`	Indicator for inheriting the aesthetics. See `geom_beeswarm`.
`...`	Additional arguments. See `geom_beeswarm`.

Value

The interactive geometry based on geom_beeswarm.

Author(s)

Patrick Aboyoun

See Also

geom_beeswarm

Examples

```
# Create NanoStringRccSet from data files
datadir <- system.file("extdata", "3D_Bio_Example_Data",
                       package = "NanoStringNCTools")
rccs <- dir(datadir, pattern = "SKMEL.*\\.RCC$", full.names = TRUE)
rlf <- file.path(datadir, "3D_SolidTumor_Sig.rlf")
pheno <- file.path(datadir, "3D_SolidTumor_PhenoData.csv")
solidTumor <-
  readNanoStringRccSet(rccs, rlfFile = rlf, phenoDataFile = pheno)

eg_data <- as.data.frame(assayDataElement(solidTumor, "exprs")[1:5, 1])
eg_data[["tooltip"]] <- names(eg_data)
geom_beeswarm_interactive(aes_string(tooltip = "tooltip"), data=eg_data)
```

---

log2t *Logarithm With Thresholding*

Description

Safe log and log2 calculations where values within [0, thresh) are thresholded to thresh prior to the transformation.

Usage

```
logt(x, thresh = 0.5)
log2t(x, thresh = 0.5)
```

Arguments

x	a numeric or complex vector.
thresh	a positive number specifying the threshold.

Details

For non-negative elements in x, calculates log(pmax(x, thresh)) or log2(pmax(x, thresh)).

Value

A vector of the same length as x containing the transformed values.

Author(s)

Patrick Aboyoun

See Also

log, log2

Examples

```
logt(0:8)
identical(logt(0:8), log(c(0.5, 1:8)))

log2t(0:8)
identical(log2t(0:8), log2(c(0.5, 1:8)))
```

NanoStringRccSet-autoplot
Plot NanoStringRccSet Data

Description

Generate common plots to visualize and QC NanoStringRccSet data.

Usage

```
        ## S3 method for class 'NanoStringRccSet'
autoplot(object,
           type = c("boxplot-feature",
                    "boxplot-signature",
                    "bindingDensity-mean",
                    "bindingDensity-sd",
                    "ercc-linearity",
                    "ercc-lod",
                    "heatmap-genes",
                    "heatmap-signatures",
                    "housekeep-geom",
                    "lane-bindingDensity",
                    "lane-fov",
                    "mean-sd-features",
                    "mean-sd-samples"),
           log2scale = TRUE,
           elt = "exprs",
           index = 1L,
           geomParams = list(),
           tooltipDigits = 4L,
           heatmapGroup = NULL,
           blacklist = NULL,
           tooltipID = NULL,
           qcCutoffs = list(
             Housekeeper = c("failingCutoff" = 32,"passingCutoff" = 100) ,
             Imaging = c("fovCutoff" = 0.75) ,
```

```
  BindingDensity = c("minimumBD" = 0.1, "maximumBD" = 2.25,
                     "maximumBDSprint" = 1.8) ,
  ERCCLinearity = c("correlationValue" = 0.95) ,
  ERCCLoD = c("standardDeviations" = 2) ),
scalingFactor=1L,
show_rownames_gene_limit=60L,
show_colnames_gene_limit=36L,
show_rownames_sig_limit=60L,
show_colnames_sig_limit=36L,
subSet = NULL ,
...)
```

Arguments

`object`	A NanoStringRccSet object
`type`	Character string referencing the type of plot to generate
`log2scale`	An optional boolean indicating expression data is on log2 scale
`elt`	An optional character string of the expression matrix name
`index`	An optional integer giving the feature of interest row location
`geomParams`	An option `list` of parameters for geometry
`tooltipDigits`	An optional integer for number of tooltip decimal places to display
`heatmapGroup`	An optional character string referencing `pData` column to color samples by in heatmap
`blacklist`	An optional character vector of features not to plot
`tooltipID`	An optional character string referencing `pData` column to use for sample ID in the tooltip
`qcCutoffs`	An optional `list` of QC cutoffs
`scalingFactor`	An optional numeric value indicating a scaling factor to apply to plot drawing
`show_rownames_gene_limit`	An optional integer limit on number of features to display row-wise
`show_colnames_gene_limit`	An optional integer limit on number of features to display column-wise
`show_rownames_sig_limit`	An optional integer limit on number of signatures to display row-wise
`show_colnames_sig_limit`	An optional integer limit on number of signatures to display column-wise
`subSet`	An optional subset to plot on
`...`	Additional arguments to pass on to autoplot function

Details

`"boxplot-feature"` Generate feature boxplots

`"boxplot-signature"` Generate signature boxplots

`"bindingDensity-mean"` Plot binding density displayed as average expression

`"bindingDensity-sd"` Plot binding density displayed as standard deviation of expression

`"ercc-linearity"` Assess linearity of ERCCs

`"ercc-lod"` Assess limit of detection based on ERCC expression

"heatmap-genes" Generate a heatmap from feature expression

"heatmap-signatures" Generate a heatmap from signature expression

"housekeep-geom" Plot geometric mean of housekeeper genes

"lane-bindingDensity" View binding density by lane

"lane-fov" Assess image quality by lane

"mean-sd-features" Plot mean versus standard deviation feature-wise

"mean-sd-samples" Plot mean versus standard deviation sample-wise

Value

A ggplot or pheatmap plot depending on the type of plot generated

Examples

```
# Create NanoStringRccSet from data files
datadir <- system.file("extdata", "3D_Bio_Example_Data",
                       package = "NanoStringNCTools")
rccs <- dir(datadir, pattern = "SKMEL.*\\.RCC$", full.names = TRUE)
rlf <- file.path(datadir, "3D_SolidTumor_Sig.rlf")
pheno <- file.path(datadir, "3D_SolidTumor_PhenoData.csv")
solidTumor <-
  readNanoStringRccSet(rccs, rlfFile = rlf, phenoDataFile = pheno)

# Assess experiment linearity
#autoplot(solidTumor, "ercc-linearity")

# Plot a feature's expression across all samples
#autoplot(solidTumor, "boxplot-feature", index=2)
```

NanoStringRccSet-class

Class to Contain NanoString Expression Level Assays

Description

The NanoStringRccSet class extends the ExpressionSet class for NanoString Reporter Code Count (RCC) data.

Usage

```
NanoStringRccSet(assayData,
                 phenoData = annotatedDataFrameFrom(assayData, byrow = FALSE),
                 featureData = annotatedDataFrameFrom(assayData, byrow = TRUE),
                 experimentData = MIAME(),
                 annotation = character(),
                protocolData = annotatedDataFrameFrom(assayData, byrow = FALSE),
                 dimLabels = c("GeneName", "SampleID"),
                 signatures = SignatureSet(),
                 design = NULL,
                 ...)
```

Arguments

assayData	A matrix or environment containing the RCCs.
phenoData	An AnnotatedDataFrame containing the phenotypic data.
featureData	An AnnotatedDataFrame containing columns "CodeClass", "GeneName", "Accession", "IsControl", and "ControlConc".
experimentData	An optional MIAME instance with meta-data about the experiment.
annotation	A character string for the "GeneRLF".
protocolData	An AnnotatedDataFrame containing columns "FileVersion", "SoftwareVersion", "SystemType", "SampleID", "SampleOwner", "SampleComments", "SampleDate", "SystemAPF", "AssayType", "LaneID", "FovCount", "FovCounted", "ScannerID", "StagePosition", "BindingDensity", "CartridgeID", and "CartridgeBarcode".
dimLabels	A character vector of length 2 that provides the column names to use as labels for the features and samples respectively in the autoplot method.
signatures	An optional SignatureSet object containing signature definitions.
design	An optional one-sided formula representing the experimental design based on columns from phenoData
...	Additional arguments for ExpressionSet.

Value

An S4 class containing NanoString Expression Level Assays

Accessing

In addition to the standard ExpressionSet accessor methods, NanoStringRccSet objects have the following:

sData(object) extracts the data.frame containing the sample data, cbind(pData(object), pData(protocolData(ob

svarLabels(object) extracts the sample data column names, c(varLabels(object), varLabels(protocolData(obje

dimLabels(object) extracts the column names to use as labels for the features and samples in the autoplot method.

dimLabels(object) <- value replaces the dimLabels of the object.

signatures(object) extracts the SignatureSet of the object.

signatures(object) <- value replaces the SignatureSet of the object.

signatureScores(object, elt = "exprs") extracts the matrix of computed signature scores.

design(object) extracts the one-sided formula representing the experimental design based on columns from phenoData.

design(object) <- value replaces the one-sided formula representing the experimental design based on columns from phenoData.

setSignatureFuncs(object) returns the signature functions.

setSignatureFuncs(object) <- value replaces the signature functions.

setSignatureGroups(object) <- value returns the signature groups.

setSignatureGroups(object) <- value replaces the signature groups.

Summarizing

summary(object, MARGIN = 2L, GROUP = NULL, log2scale = TRUE, elt = "exprs", signatureScores = FALSE)
When `signatureScores = FALSE`, the marginal summaries of the `elt` `assayData` matrix along either the feature (`MARGIN = 1`) or sample (`MARGIN = 2`) dimension.

When `signatureScores = TRUE`, the marginal summaries of the `elt` `signatureScores` matrix along either the signature (`MARGIN = 1`) or sample (`MARGIN = 2`) dimension.

When `log2scale = FALSE`, the summary statistics are Mean, Standard Deviation, Skewness, Excess Kurtosis, Minimum, First Quartile, Median, Third Quartile, and Maximum.

When `log2scale = TRUE`, the summary statistics are Geometric Mean with thresholding at 0.5, Size Factor (`2^(`MeanLog2` - mean(`MeanLog2`))`), Mean of Log2 with thresholding at 0.5, Standard Deviation of Log2 with thresholding at 0.5, Minimum, First Quartile, Median, Third Quartile, and Maximum.

Subsetting

In addition to the standard `ExpressionSet` subsetting methods, `NanoStringRccSet` objects have the following:

subset(x, subset, select, ...) Subset the feature and sample dimensions using the `subset` and `select` arguments respectively. The `subset` argument will be evaluated with respect to the `featureData`, while the `select` argument will be evaluated with respect to the `phenoData` and `protocolData`.

endogenousSubset(x, subset, select) Extracts the endogenous barcode class feature subset of `x` with optional additional subsetting using `subset` and `select`.

housekeepingSubset(x, subset, select) Extracts the housekeeping barcode class feature subset of `x` with optional additional subsetting using `subset` and `select`.

negativeControlSubset(x, subset, select) Extracts the negative control barcode class feature subset of `x` with optional additional subsetting using `subset` and `select`.

positiveControlSubset(x, subset, select) Extracts the positive control barcode class feature subset of `x` with optional additional subsetting using `subset` and `select`.

controlSubset(x, subset, select) Extracts the feature subset representing the controls of `x` with optional additional subsetting using `subset` and `select`.

nonControlSubset(x, subset, select) Extracts the feature subset representing the non-controls of `x` with optional additional subsetting using `subset` and `select`.

signatureSubset(x, subset, select) Extracts the feature subset representing the genes in the `signatures` of `x` with optional additional subsetting using `subset` and `select`.

Looping

assayDataApply(X, MARGIN, FUN, ..., elt = "exprs") Loop over the feature (`MARGIN = 1`) or sample (`MARGIN = 2`) dimension of `assayDataElement(X, elt)`.

signatureScoresApply(X, MARGIN, FUN, ..., elt = "exprs") Loop over the signature (`MARGIN = 1`) or sample (`MARGIN = 2`) dimension of `signatureScores(X, elt)`.

esBy(X, GROUP, FUN, ..., simplify = TRUE) Split `X` by `GROUP` column within `featureData`, `phenoData`, or `protocolData` and apply `FUN` to each partition.

Transforming

munge(data, mapping = update(design(data), exprs ~ .), extradata = NULL, elt = "exprs", ...) munge argument `data` into a data.frame object for modeling and visualization using the `mapping` argument. Supplemental data can be specified using the `extradata` argument.

transform('_data', ...) Similar to the `transform` generic in the **base** package, creates or modifies one or more `assayData` matrices based upon `name = value` pairs in `...`. The expressions in `...` are appended to the preprocessing list in `experimentData`, which can be extracted using the `preproc` method.

Evaluating

with(data, expr, ...) Evaluate expression `expr` with respect to `assayData`, `featureData`, `phenoData`, and `protocolData`; `c(as.list(assayData(data)), fData(data), sData(data))`.

Normalizing

normalize(object, type, fromElt = "exprs", toElt = "exprs_norm", ...)

Plotting

ggplot(data, mapping = aes(), ..., extradata = NULL, tooltip_digits = 4L, environment = parent.frame()) the `NanoStringRccSet` method for `ggplot`.

autoplot(object, type, log2scale = TRUE, elt = "exprs", index = 1L, geomParams = list(), tooltipDigits = 4L, heatm

Author(s)

Patrick Aboyoun

See Also

`readNanoStringRccSet`, `writeNanoStringRccSet`, `ExpressionSet`

Examples

```
# Create NanoStringRccSet from data files
datadir <- system.file("extdata", "3D_Bio_Example_Data",
                       package = "NanoStringNCTools")
rccs <- dir(datadir, pattern = "SKMEL.*\\.RCC$", full.names = TRUE)
rlf <- file.path(datadir, "3D_SolidTumor_Sig.rlf")
pheno <- file.path(datadir, "3D_SolidTumor_PhenoData.csv")
solidTumor <-
  readNanoStringRccSet(rccs, rlfFile = rlf, phenoDataFile = pheno)


# Create a deep copy of a NanoStringRccSet object
deepCopy <- NanoStringRccSet(solidTumor)
all.equal(solidTumor, deepCopy)
identical(solidTumor, deepCopy)


# Accessing sample data and column names
head(sData(solidTumor))
svarLabels(solidTumor)


# Set experimental design
design(solidTumor) <- ~ BRAFGenotype + Treatment
design(solidTumor)
munge(solidTumor)
```

```
# Marginal summarizing of NanoStringRccSet assayData matrices
head(summary(solidTumor, 1)) # Marginal summaries along features
head(summary(solidTumor, 2)) # Marginal summaries along samples


# Subsetting NanoStringRccSet objects
# Extract the positive controls for wildtype BRAF
dim(solidTumor)
dim(subset(solidTumor, CodeClass == "Positive", BRAFGenotype == "wt/wt"))

# Extract by barcode class
with(solidTumor, table(CodeClass))
with(endogenousSubset(solidTumor), table(CodeClass))
with(housekeepingSubset(solidTumor), table(CodeClass))
with(negativeControlSubset(solidTumor), table(CodeClass))
with(positiveControlSubset(solidTumor), table(CodeClass))
with(controlSubset(solidTumor), table(CodeClass))
with(nonControlSubset(solidTumor), table(CodeClass))


# Looping over NanoStringRccSet assayData matrices
log1pCoefVar <- function(x){
  x <- log1p(x)
  sd(x) / mean(x)
}

# Log1p Coefficient of Variation along Features
head(assayDataApply(solidTumor, 1, log1pCoefVar))

# Log1p Coefficient of Variation along Samples
head(assayDataApply(solidTumor, 2, log1pCoefVar))


# Transforming NanoSetRccSet assayData matrices
# Subtract max count from each sample
# Create log1p transformation of adjusted counts
thresh <- assayDataApply(negativeControlSubset(solidTumor), 2, max)
solidTumor2 <-
  transform(solidTumor,
            negCtrlZeroed = sweep(exprs, 2, thresh),
            log1p_negCtrlZeroed = log1p(pmax(negCtrlZeroed, 0)))
assayDataElementNames(solidTumor2)


# Evaluating expression using NanoStringRccSet data
meanLog1pExprs <-
  with(solidTumor,
       {
         means <- split(apply(exprs, 1, function(x) mean(log1p(x))), CodeClass)
         means <- means[order(sapply(means, median))]
         boxplot(means, horizontal = TRUE)
         means
       })
```

normalize *Normalize RCCSet*

Description

This package performs normalization on NanoStringRccSet data using one of three methods.

Usage

```
normalize(object, ...)
```

Arguments

object	object NanoStringRccSet object
...	object additional arguments to pass on to normalize function

Details

Normalization is performed in one of three ways with data pulled from one slot of assayData and inserted into another. It is possible to overwrite the original slot of assayData if the fromElt and toElt are set to the same slot. nSolver normalization uses positive controls to scale and housekeepers to standardize the data and mimics the normalization performed by default in the nSolver software. The Housekeeping-Log2 normalization calculates the log2 sizeFactor of the housekeeping genes and then takes 2^ log2 expression data centered by the log transformed sizeFactor. PositiveControl-Log2Log2 regresses the log2 positive control probes greater than 0.5 concentration on their geometric mean and then uses the intercept and slope to predict normalized values from the log2 transformed expression values. The predictions are then rescaled by 2^. Additional parameters with NanoStringRccSet method include:

type normalization method to use. Options are nSolver, Housekeeping-Log2, and PositiveControl-Log2Log2

fromElt assayData slot from which to pull raw data

toElt assayData slot to which normalized data will be inserted

Value

The function returns a new NanoStringRccSet with either an additional assayData slot of normalized data, or overwrites the original assayData depending on whether fromElt and toElt are identical.

Author(s)

Patrick Aboyoun

References

NanoString nSolver User Manual https://www.nanostring.com/download_file/view/1168

Examples

```
datadir <- system.file("extdata", "3D_Bio_Example_Data",
                       package = "NanoStringNCTools")
rccs <- dir(datadir, pattern = "SKMEL.*\\.RCC$", full.names = TRUE)
rlf <- file.path(datadir, "3D_SolidTumor_Sig.rlf")
pheno <- file.path(datadir, "3D_SolidTumor_PhenoData.csv")

solidTumor <-
  readNanoStringRccSet(rccs, rlfFile = rlf, phenoDataFile = pheno)

solidTumor <- normalize(solidTumor, "nSolver" , fromElt = "exprs", toElt = "exprs_norm")
head( assayDataElement( solidTumor , elt = "exprs_norm" ) )
```

---

readNanoStringRccSet *Read 'NanoStringRccSet'*

Description

Create an instance of class NanoStringRccSet by reading data from NanoString Reporter Code Count (RCC) files.

Usage

```
readNanoStringRccSet(rccFiles, rlfFile = NULL,
                     phenoDataFile = NULL,
                     phenoDataRccColName = "^RCC",
                     phenoDataColPrefix = "")
```

Arguments

rccFiles	A character vector containing the paths to the RCC files.
rlfFile	An optional character string representing the path to the corresponding RLF file.
phenoDataFile	An optional character string representing the path to the corresponding phenotypic csv data file.
phenoDataRccColName	The regular expression that specifies the RCC column in the phenoDataFile.
phenoDataColPrefix	An optional prefix to add to the phenoData column names to distinguish them from the names of assayData matrices, featureData columns, and protocolData columns.

Value

An instance of the NanoStringRccSet class.

Author(s)

Patrick Aboyoun

See Also

NanoStringRccSet, writeNanoStringRccSet

Examples

```
# Data file paths
datadir <- system.file("extdata", "3D_Bio_Example_Data",
                       package = "NanoStringNCTools")
rccs <- dir(datadir, pattern = "SKMEL.*\\.RCC$", full.names = TRUE)
rlf <- file.path(datadir, "3D_SolidTumor_Sig.rlf")
pheno <- file.path(datadir, "3D_SolidTumor_PhenoData.csv")

# Just RCC data
solidTumorNoRlfPheno <- readNanoStringRccSet(rccs)
varLabels(solidTumorNoRlfPheno)
fvarLabels(solidTumorNoRlfPheno)

# RCC and RLF data
solidTumorNoPheno <- readNanoStringRccSet(rccs, rlfFile = rlf)
setdiff(fvarLabels(solidTumorNoPheno), fvarLabels(solidTumorNoRlfPheno))

# All data
solidTumor <-
  readNanoStringRccSet(rccs, rlfFile = rlf, phenoDataFile = pheno)
varLabels(solidTumor)
design(solidTumor) <- ~ BRAFGenotype + Treatment

# All data with phenoData prefix
solidTumorPhenoPrefix <-
  readNanoStringRccSet(rccs, rlfFile = rlf, phenoDataFile = pheno,
                       phenoDataColPrefix = "PHENO_")
varLabels(solidTumorPhenoPrefix)
design(solidTumorPhenoPrefix) <- ~ PHENO_BRAFGenotype + PHENO_Treatment
```

---

readRccFile *Read RCC File*

Description

Read a NanoString Reporter Code Count (RCC) file.

Usage

```
readRccFile(file)
```

Arguments

`file`	A character string containing the path to the RCC file.

Value

An list object with five elements:

`"Header"`	a `data.frame` object containing the header information.
`"Sample_Attributes"`	a `data.frame` object containing the attributes of the sample.
`"Lane_Attributes"`	a `data.frame` object containing the attributes of the lane.

"Code_Summary"	a data.frame object containing the reporter code counts.
"Messages"	A character vector containing messages, if any.

Author(s)

Patrick Aboyoun

See Also

readNanoStringRccSet

Examples

```
datadir <- system.file("extdata", "3D_Bio_Example_Data",
                       package = "NanoStringNCTools")
rccs <- dir(datadir, pattern = "SKMEL.*\\.RCC$", full.names = TRUE)
rccData <- lapply(rccs, readRccFile)
```

---

readRlfFile *Read RLF File*

Description

Read a NanoString Reporter Library File (RLF) file.

Usage

```
readRlfFile(file)
```

Arguments

file	A character string containing the path to the RLF file.

Value

An instance of the DataFrame class containing columns:

"CodeClass"	code class
"GeneName"	gene name
"Accession"	accession number
...	additional columns

Author(s)

Patrick Aboyoun

See Also

readNanoStringRccSet

Examples

```
datadir <- system.file("extdata", "3D_Bio_Example_Data",
                       package = "NanoStringNCTools")
rlf <- file.path(datadir, "3D_SolidTumor_Sig.rlf")
rlfData <- readRlfFile(rlf)
```

`setQCFlags` *Set flags for QC of the assayData in a NanoStringRccSet.*

Description

This function takes a list containing the quality control (QC) thresholds for data in a NanoStringRccSet and then returns a matrix of QC retults by sample to protocolData.

Usage

```
setQCFlags(object, ...)
```

Arguments

`object`	A valid NanoStringRccSet object with all housekeeping genes, positive control probes, and negative control probes present
`...`	Additional arguments to pass

Details

This function checks that the housekeeping genes, positive control, and negative control probes or genes are within acceptable boundaries. Additional parameters with NanoStringRccSet method include:

`qcCutoffs` An optional list with members named `Housekeeper`, `Imaging`, `BindingDensity`, `ERCCLinearity`, and `ERCCLoD`

`hkGenes` An optional vector of housekeeping gene names if alternative genes to those defined in the panel are to be used

`ReferenceSampleColumn` An optional character string indicating the `pData` column containing reference sample information

Borderline thresholds and fail thresholds are defined and each sample receives a row in a matrix that contains flags indicating either borderline or failing performance.

`Housekeeper` is a vector with names members. `failingCutoff` sets the lower bound of housekeeper gene expression such that samples with a value below this threshold are labeled as failures. `passingCutoff` sets a lower bound of housekeeper gene expression such that samples with a value below this threshold are labeled as borderline. Values greater than or equal to either threshold are labeled as either borderline or passing. The default values are `failingCutoff` = 32 and `passingCutoff` = 100.

`Imaging` is a vector with a single named member `fovCutoff`. This threshold determines the mimimum proportion of FOV to be counted. The default value is 0.75.

`BindingDensity` is a named vector with members `minimumBD`, `maximumBD`, and `maximumBDSprint`. `minimumBD` sets a minimum threshold for binding density across machine platforms. `maximumBD` sets a maxmimum binding density for non-Sprint machines while `maximumBDSprint` does the

same for Sprint machines. The default values are `minimumBD` = 0.1, `maximumBD` = 2.25, and `maximumBDSprint` = 1.8.

`ERCCLinearity` is a named vector with a single member `correlationValue`. This member sets a minimum threshold for the correlation between the observed counts of positive controls and their theoretical concentration. The default value is 0.95.

`ERCCLoD` is a named vector with a single member `standardDeviations`. This sets a minimum threshold for the 0.5uMol concentration to be above the geoMean of the negative controls in units of standard deviation of the negative controls. The default value is 2.

Value

This function returns a new `NanoStringRccSet` with matrices of QC pass and QC borderline criteria added to the `protocolData` slots called `QCFlags` and `QCBorderlineFlags`, respectively.

Examples

```
# Create NanoStringRccSet from data files
datadir <- system.file("extdata", "3D_Bio_Example_Data",
                       package = "NanoStringNCTools")
rccs <- dir(datadir, pattern = "SKMEL.*\\.RCC$", full.names = TRUE)
rlf <- file.path(datadir, "3D_SolidTumor_Sig.rlf")
pheno <- file.path(datadir, "3D_SolidTumor_PhenoData.csv")
solidTumor <-
  readNanoStringRccSet(rccs, rlfFile = rlf, phenoDataFile = pheno)

#Set QC flags with default cutoffs
solidTumorDefaultQC <- setQCFlags(solidTumor)
head( protocolData( solidTumorDefaultQC )[["QCFlags"]] )
head( protocolData( solidTumorDefaultQC )[["QCBorderlineFlags"]] )

#Update cutoffs
newQCCutoffs <- list(
  Housekeeper = c("failingCutoff" = 32,"passingCutoff" = 100) ,
  Imaging = c("fovCutoff" = 0.75) ,
  BindingDensity = c("minimumBD" = 0.1, "maximumBD" = 2.25, "maximumBDSprint" = 1.8) ,
  ERCCLinearity = c("correlationValue" = 0.98) ,
  ERCCLoD = c("standardDeviations" = 2)
  )

#Set QC flags with new cutoffs
solidTumorNewQC <- setQCFlags(solidTumor, qcCutoffs=newQCCutoffs)

#Compare QC results with default and new cutoffs
head( protocolData( solidTumorDefaultQC )[["QCFlags"]] )
head( protocolData( solidTumorNewQC )[["QCFlags"]] )
```

SignatureSet-class *Class to Contain Signature Definitions*

Description

The `SignatureSet` class defines gene-based signatures.

Usage

```
SignatureSet(weights = NumericList(), groups = factor(), func = character(),
             version = character(), ...)
```

Arguments

weights	A named NumericList defining signatures based on linear combinations of genes.
groups	A factor vector indicating groups in the SignatureSet
func	Character indicating function to use
version	Character indicating version to use
...	Additional arguments for future use.

Value

A SignatureSet object

Utilities

length(x) returns the number of signatures in x.

lengths(x, use.names = TRUE) returns a named integer vector containing the number of genes in each of the signatures in x.

names(x) returns a character vector containing the signature names in x.

weights(object) returns a named NumericList that defines the linear combination based signatures.

weights(object) <- value replaces the NumericList that defines the linear combination based signatures.

getSigFuncs(object) returns the signature functions of an object.

groups(object) returns a factor vector representing the signature groups.

groups(object) <- value replaces the factor vector representing the signature groups.

version(object) : returns the signature version.

Author(s)

Patrick Aboyoun

See Also

NanoStringRccSet

Examples

```
SignatureSet(weights=list(x = c(a = 1),
                          y = c(b = 1/3, d = 2/3),
                          z = c(a = 2, c = 4)),
             groups=factor("x", "y", "z"),
             func = c(x="default", y="default", z="default"))
```

sThresh *Convenience Functions for Assay Data Element Sweep Operations*

Description

Convenience functions for matrix thresholding, centering, and scaling based upon margin statistics.

Usage

```
# Loop over features
fThresh(x, STATS)
fCenter(x, STATS)
fScale(x, STATS)

## Round results to integers
fIntThresh(x, STATS)
fIntCenter(x, STATS)
fIntScale(x, STATS)

## Comparisons
fAbove(x, STATS)
fBelow(x, STATS)
fAtLeast(x, STATS)
fAtMost(x, STATS)


# Loop over samples
sThresh(x, STATS)
sCenter(x, STATS)
sScale(x, STATS)

# Round results to integers
sIntThresh(x, STATS)
sIntCenter(x, STATS)
sIntScale(x, STATS)

## Comparisons
sAbove(x, STATS)
sBelow(x, STATS)
sAtLeast(x, STATS)
sAtMost(x, STATS)
```

Arguments

x	a numeric array.
STATS	the summary statistic for thresholding, centering, or scaling.

Details

These functions are convenience wrappers for the following code:

```
fThresh: sweep(x, 1L, STATS, FUN = "pmax")
fCenter: sweep(x, 1L, STATS, FUN = "-")
fScale: sweep(x, 1L, STATS, FUN = "/")
fIntThresh: round(sweep(x, 1L, STATS, FUN = "pmax"))
fIntCenter: round(sweep(x, 1L, STATS, FUN = "-"))
fIntScale: round(sweep(x, 1L, STATS, FUN = "/"))
fAbove: sweep(x, 1L, STATS, FUN = ">")
fBelow: sweep(x, 1L, STATS, FUN = "<")
fAtLeast: sweep(x, 1L, STATS, FUN = ">=")
fAtMost: sweep(x, 1L, STATS, FUN = "<=")
sThresh: sweep(x, 2L, STATS, FUN = "pmax")
sCenter: sweep(x, 2L, STATS, FUN = "-")
sScale: sweep(x, 2L, STATS, FUN = "/")
sIntThresh: round(sweep(x, 2L, STATS, FUN = "pmax"))
sIntCenter: round(sweep(x, 2L, STATS, FUN = "-"))
sIntScale: round(sweep(x, 2L, STATS, FUN = "/"))
sAbove: sweep(x, 2L, STATS, FUN = ">")
sBelow: sweep(x, 2L, STATS, FUN = "<")
sAtLeast: sweep(x, 2L, STATS, FUN = ">=")
sAtMost: sweep(x, 2L, STATS, FUN = "<=")
```

Value

An array with the same shape as x that has been modified by thresholding, centering, or scaling.

Author(s)

Patrick Aboyoun

See Also

sweep

Examples

```
# Find reasonable column minimums
thresh <- apply(stack.x, 2L, quantile, 0.05)

# Threshold column values
identical(sThresh(stack.x, thresh),
          sweep(stack.x, 2L, thresh, FUN = "pmax"))

# Substract column values
identical(sCenter(stack.x, thresh),
          sweep(stack.x, 2L, thresh))

# Scale to common mean
identical(sScale(stack.x, colMeans(stack.x) / mean(colMeans(stack.x))),
          sweep(stack.x, 2L, colMeans(stack.x) / mean(colMeans(stack.x)),
```

```
                FUN = "/"))

# Scale to common mean, rounded to the nearest integer
sIntScale(stack.x, colMeans(stack.x) / mean(colMeans(stack.x)))
```

---

writeNanoStringRccSet *Write NanoString Reporter Code Count (RCC) files*

Description

Write NanoString Reporter Code Count (RCC) files from an instance of class NanoStringRccSet.

Usage

```
writeNanoStringRccSet(x, dir = getwd())
```

Arguments

x	an instance of class NanoStringRccSet.
dir	An optional character string representing the path to the directory for the RCC files.

Details

Writes a set of NanoString Reporter Code Count (RCC) files based upon x in dir.

Value

A character vector containing the paths for all the newly created RCC files.

Author(s)

Patrick Aboyoun

See Also

NanoStringRccSet, readNanoStringRccSet

Examples

```
datadir <- system.file("extdata", "3D_Bio_Example_Data",
                       package = "NanoStringNCTools")
rccs <- dir(datadir, pattern = "SKMEL.*\\.RCC$", full.names = TRUE)
solidTumorNoRlfPheno <- readNanoStringRccSet(rccs)
writeNanoStringRccSet(solidTumorNoRlfPheno, tempdir())
for (i in seq_along(rccs)) {
  stopifnot(identical(readLines(rccs[i]),
                      readLines(file.path(tempdir(), basename(rccs[i])))))
}
```

Index