Package: GeomxTools (via r-universe)

June 4, 2026

Title NanoString GeoMx Tools

Description Tools for NanoString Technologies GeoMx Technology. Package provides functions for reading in DCC and PKC files based on an ExpressionSet derived object. Normalization and QC functions are also included.

Version 3.16.0

Encoding UTF-8

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

Imports BiocGenerics, rjson, readxl, EnvStats, reshape2, methods, utils, stats, data.table, lmerTest, dplyr, stringr, grDevices, graphics, GGally, rlang, ggplot2, SeuratObject

Suggests rmarkdown, knitr, testthat (>= 3.0.0), parallel, ggiraph, Seurat, SpatialExperiment (>= 1.4.0), SpatialDecon, patchwork

License MIT

Collate DccMetadata.R NanoStringGeoMxSet-class.R NanoStringGeoMxSet-validity.R NanoStringGeoMxSet-accessors.R NanoStringGeoMxSet-qc.R NanoStringGeoMxSet-autoplot.R NanoStringGeoMxSet-aggregate.R NanoStringGeoMxSet-signatures.R NanoStringGeoMxSet-normalize.R NanoStringGeoMxSet-de.R coercions.R readDccFile.R readPKCFile.R readNanoStringGeoMxSet.R writeNanoStringGeoMxSet.R utils.R outliersFunctions.R

biocViews GeneExpression, Transcription, CellBasedAssays, DataImport, Transcriptomics, Proteomics, mRNAMicroarray, ProprietaryPlatforms, RNASeq, Sequencing, ExperimentalDesign, Normalization, Spatial

VignetteEngine knitr::rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.1

Config/testthat/edition 3

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev make libicu-dev libpng-dev libuv1-dev libssl-dev zlib1g-dev

Repository https://bioc-release.r-universe.dev

Date/Publication 2026-04-28 12:54:51 UTC

RemoteUrl https://github.com/bioc/GeomxTools

RemoteRef RELEASE_3_23

RemoteSha 6033ab5ab6c128cd27face71a4eadd67444d1822

Contents

aggregateCounts *Aggregate probe counts to target level for feature data*

Description

Aggregate probe counts to target level for feature data

Usage

```
aggregateCounts(object, FUN = ngeoMean)
```

Arguments

object	name of the NanoStringGeoMxSet object to aggregate
FUN	function to use for count aggregation

Value

a NanoStringGeoMxSet object with targets as features

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
targetGeoMxSet <- aggregateCounts(demoData[,1:10])
```

as.Seurat *Convert GeoMxSet Object to SeuratObject*

Description

Convert GeoMxSet Object to SeuratObject

Usage

```
## S3 method for class 'NanoStringGeoMxSet'
as.Seurat(
  x,
  ident = NULL,
  normData = NULL,
  coordinates = NULL,
  forceRaw = FALSE,
  ...
)
```

Arguments

x	An object to convert to class Seurat
ident	column in GeoMxSet segmentProperties to set as Seurat object's identity class
normData	assay containing normalized data
coordinates	X and Y coordinates of each ROI, format: c(X,Y)
forceRaw	should raw data be forced into SeuratObject, not recommended
...	Arguments passed to other methods

Value

SeuratObject containing GeoMx data

See Also

SeuratObject::as.Seurat

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data", package = "GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))

target_demoData <- aggregateCounts(demoData[1:1000,1:10])

target_demoData <- normalize(target_demoData, "quant")

seurat_demoData <- as.Seurat(target_demoData, ident = "cell_line",
                             normData = "exprs_norm", forceRaw = FALSE)
```

---

as.SpatialExperiment *Convert Object to SpatialExperiment*

Description

Convert Object to SpatialExperiment

Convert GeoMxSet Object to SpatialExperiment

Usage

```
as.SpatialExperiment(x, ...)

## S3 method for class 'NanoStringGeoMxSet'
as.SpatialExperiment(
  x,
  normData = NULL,
  coordinates = NULL,
```

```
  forceRaw = FALSE,
  ...
)
```

Arguments

x	GeoMxSet object to convert
...	Arguments passed to other methods
normData	assay containing normalized data
coordinates	X and Y coordinates of each ROI, format: c(X,Y)
forceRaw	should raw data be forced into SpatialExperiment, not recommended

Value

SpatialExperiment containing GeoMx data

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package = "GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))

target_demoData <- aggregateCounts(demoData[1:1000,1:10])

target_demoData <- normalize(target_demoData, "quant")

seurat_demoData <- as.SpatialExperiment(target_demoData,
                                        normData = "exprs_norm",
                                        forceRaw = FALSE)
```

---

checkQCFlags *Check QC Flags in the GeoMxSet and removes the probe or sample from the object*

Description

Check QC Flags in the GeoMxSet and removes the probe or sample from the object

Usage

```
checkQCFlags(object, ...)
```

Arguments

object	name of the NanoStringGeoMxSet object to check the QC Flags
...	for other arguments

Value

a NanoStringGeoMxSet object probes and samples failing QC removed

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
  package = "GeomxTools"
)
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
QCobject <- checkQCFlags(demoData)
```

---

checkQCFlags,NanoStringGeoMxSet-method
checkQCFlags

Description

checkQCFlags

Usage

```
## S4 method for signature 'NanoStringGeoMxSet'
checkQCFlags(object, removeLowLocalOutliers = FALSE, ...)
```

Arguments

`object`	name of the NanoStringGeoMxSet object to check the QC Flags
`removeLowLocalOutliers`	logical, if TRUE it sets outlier counts to zero, default is FALSE,
`...`	optional arguments

Value

NanoStringGeoMxSet

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
  package = "GeomxTools"
)
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
QCobject <- checkQCFlags(demoData)
```

compareToConfig *Compare given PKC probes to probes in config file*

Description

Check if extra PKCs are given based on probes in config file

Usage

```
compareToConfig(config, pkcProbes, pkcHeader)
```

Arguments

config	file path to config file
pkcProbes	probe information from readPKCFile
pkcHeader	pkc metadata from readPKCFile

computeNormalizationFactors *Generate normalization factors*

Description

For use with protein data ONLY.

Generate normalization factors for protein data to determine the best normalization method

Usage

```
computeNormalizationFactors(
  object,
  igg.names = NULL,
  hk.names = NULL,
  area = NULL,
  nuclei = NULL
)
```

Arguments

object	name of the object class to subset 1. NanoStringGeoMxSet, use the NanoStringGeoMxSet class
igg.names	names of IgGs, if NULL IgGs will be detected automatically
hk.names	names of HK, if NULL HK will be detected automatically
area	name of area column in annotation sheet, optional
nuclei	name of nuclei column in annotation sheet, optional

Examples

```
proteinData <- readRDS(file= system.file("extdata","DSP_Proteogenomics_Example_Data",
"proteinData.rds", package = "GeomxTools"))

normfactors <- computeNormalizationFactors(object = proteinData)

normfactors_withAreaNuclei <- computeNormalizationFactors(object = proteinData,
area = "AOI.Size.um2", nuclei = "Nuclei.Counts")
```

---

countsShiftedByOne *Accessor to check if "exprs"* assDataElement *was shifted by one*

Description

Accessor to check if "exprs" assDataElement was shifted by one

Usage

```
countsShiftedByOne(object)
```

Arguments

object	name of the NanoStringGeoMxSet object

Value

boolean indicating if counts in default matrix were shifted by one

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
countsShiftedByOne(demoData)
```

hkNames *Return the House Keeper positive controls for protein*

Description

Return the House Keeper positive controls for protein

Usage

```
hkNames(object)
```

Arguments

object	name of the NanoStringGeoMxSet object

Value

names of HKs

iggNames *Return the IgG negative controls for protein*

Description

Return the IgG negative controls for protein

Usage

```
iggNames(object)
```

Arguments

object	name of the NanoStringGeoMxSet object

Value

names of IgGs

logtBase *Get take the log of a numeric vector*

Description

Get take the log of a numeric vector

Usage

```
logtBase(x, thresh = 0.5, base = 2)
```

Arguments

x	numeric vector
thresh	minimum numeric value greater than 0 to have in vector
base	numeric value indicating base to log with

Value

numeric vector with logged values

Examples

```
logtBase(c(0, 1, 2, 2), thresh=0.1, base=10)
```

mixedModelDE *Run a mixed model on GeoMxSet*

Description

Run a mixed model on GeoMxSet

Usage

```
mixedModelDE(
  object,
  elt = "exprs",
  modelFormula = NULL,
  groupVar = "group",
  nCores = 1,
  multiCore = TRUE,
  pAdjust = "BY",
  pairwise = TRUE
)
```

Arguments

object	name of the object class to perform QC on 1. NanoStringGeoMxSet, use the NanoStringGeoMxSet class
elt	assayDataElement of the geoMxSet object to run the DE on
modelFormula	formula used in DE, if null, the design(object) is used
groupVar	= "group", sample annotation to group the data for comparing means
nCores	= 1, number of cores to use, set to 1 if running in serial mode
multiCore	= TRUE, set to TRUE to use multiCore, FALSE to run in cluster mode
pAdjust	= "BY" method for p-value adjustment
pairwise	boolean to calculate least-square means pairwise differences

Value

mixed model output list

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data", package = "GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
target_demoData <- aggregateCounts(demoData)
target_demoData <- normalize(target_demoData, norm_method="quant")
pData(target_demoData)[["slide"]] <-
    factor(pData(target_demoData)[["slide name"]])
protocolData(target_demoData)[["pool_rep"]] <-
    factor(protocolData(target_demoData)[["pool_rep"]])
mixedOutmc <- mixedModelDE(target_demoData,
                           elt = "exprs_norm",
                           modelFormula = ~ pool_rep +  (1 | slide),
                           groupVar = "pool_rep",
                           nCores = 2,
                           multiCore = TRUE,
                           pAdjust = NULL
)
```

NanoStringGeoMxSet-class

Class to Contain NanoString Spatial Expression Level Assays

Description

The NanoStringGeoMxSet class extends the ExpressionSet class for NanoString GeoMx Digital Count Conversion (DCC) data.

Usage

```
NanoStringGeoMxSet(assayData,
              phenoData=Biobase::annotatedDataFrameFrom(assayData, byrow=FALSE),
             featureData=Biobase::annotatedDataFrameFrom(assayData, byrow=TRUE),
                  experimentData=Biobase::MIAME(),
                  annotation=character(),
            protocolData=Biobase::annotatedDataFrameFrom(assayData, byrow=FALSE),
                  dimLabels=c("TargetName", "SampleID"),
                  signatures=SignatureSet(),
                  design=NULL,
                  featureType="Probe",
                  analyte="RNA",
                  ...)
```

Arguments

assayData	A matrix or environment containing the DCCs.
phenoData	An AnnotatedDataFrame containing the phenotypic data of areas of interest.
featureData	An AnnotatedDataFrame containing target information; target name, accession number, functional groups, etc.
experimentData	An optional MIAME instance with meta-data about the experiment.
annotation	A character string for the PKC file(s).
protocolData	An AnnotatedDataFrame containing meta-data about the protocol and sequencing; columns could include "FileVersion", "SoftwareVersion", "Date", "Plate_ID", "Well", "SeqSetId", "trimGaloreOpts", "flash2Opts", "umiExtractOpts", "boxtie2Opts", "Raw", "Trimmed", "Stitched", "Aligned", "umiQ30", "rtsQ30".
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
featureType	A character string indicating if features are on "Probe" or "Target" level
analyte	A character string indicating if features are "RNA" or "Protein"
...	Additional arguments for ExpressionSet.

Value

An S4 class containing data from a NanoString GeoMx experiment

Updating

Objects can be updated to current version using updateGeoMxSet(object)

Accessing

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

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

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

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

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

`featureType(object)` extracts the `featureType` of the `object`.

`featureType(object) <- value` replaces the `featureType` 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.

`signatureGroups(object)` extract the groups of SignatureSet.

`signatureGroups(object) <- value` replaces the groups of SignatureSet.

`analyte(object)` extracts the `analyte` of the `object`.

Author(s)

Zhi Yang & Nicole Ortogero

See Also

readNanoStringGeoMxSet, ExpressionSet

Examples

```
# Create NanoStringGeoMxSet from data files
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
dccFiles <- dir(datadir, pattern=".dcc$", full.names=TRUE)
pkc <- unzip(zipfile = file.path(datadir,  "/pkcs.zip"))
sampleAnnotationFile <- file.path(datadir, "annotations.xlsx")

dccFileColumn <- "Sample_ID"

dccSet <- readNanoStringGeoMxSet(dccFiles=dccFiles,
                                 pkcFiles=pkc,
                                 phenoDataFile=sampleAnnotationFile,
                                 phenoDataSheet="CW005",
                                 phenoDataDccColName=dccFileColumn,
```

```
                                protocolDataColNames=c("aoi", "cell_line",
                                                       "roi_rep", "pool_rep",
                                                       "slide_rep"),
                                experimentDataColNames="panel",
                                phenoDataColPrefix="")

# Accessing sample data and column names
head(sData(dccSet))
svarLabels(dccSet)
featureType(dccSet)
analyte(dccSet)

# Accessing number of samples and features
dim(dccSet)
```

---

ngeoMean *Get the geometric mean of a vector*

Description

Get the geometric mean of a vector

Usage

```
ngeoMean(x, thresh = 0.5)
```

Arguments

x	numeric vector
thresh	minimum numeric value greater than 0 to have in vector

Value

numeric geometric mean of vector

Examples

```
ngeoMean(c(0, 1, 2, 2), thresh=0.1)
```

ngeoSD *Get the geometric standard deviation of a vector*

Description

Get the geometric standard deviation of a vector

Usage

```
ngeoSD(x, thresh = 0.5)
```

Arguments

x	numeric vector
thresh	minimum numeric value greater than 0 to have in vector

Value

numeric geometric standard deviation of vector

Examples

```
ngeoSD(c(0, 1, 2, 2), thresh=0.1)
```

normalize,NanoStringGeoMxSet-method *normalize*

Description

normalize GeoMxSet using different normalization methods

Usage

```
## S4 method for signature 'NanoStringGeoMxSet'
normalize(
  object,
  norm_method = c("quant", "neg", "hk", "subtractBackground"),
  fromElt = "exprs",
  toElt = "exprs_norm",
  housekeepers = HOUSEKEEPERS,
  ...
)
```

Arguments

object	name of the object class to perform normalization on
norm_method	the normalization method to be applied on the object
fromElt	name of the assayDataElement to normalize
toElt	name of the assayDataElement to store normalized values
housekeepers	optional vector of housekeeper target names
...	optional arguments

Value

a NanoStringGeoMxSet object with normalized counts and normalized factors

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
  package = "GeomxTools"
)
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
norm_object <- normalize(demoData[1:1000,1:10])
```

---

plotConcordance *Generate concordance figure of targets based on user provided factors*

Description

Upper panels are the concordance plot. Lower panels are the standard deviation of the log2-ratios between the targets

Usage

```
plotConcordance(targetList, object, plotFactor)
```

Arguments

targetList	names of targets to plot concordance, normally IgGs.
object	name of the object class to subset 1. NanoStringGeoMxSet, use the NanoStringGeoMxSet class
plotFactor	segment factor to color the plot by

Examples

```
proteinData <- readRDS(file= system.file("extdata","DSP_Proteogenomics_Example_Data",
"proteinData.rds", package = "GeomxTools"))

igg.names <- iggNames(proteinData)

protSegTypeFig <- plotConcordance(targetList = igg.names, object = proteinData,
                                  plotFactor = "Segment_Type")
protSegTypeFig
```

`plotNormFactorConcordance`

Generate concordance figure of normalization factors based on user provided factors

---

Description

For use with protein data ONLY.

Upper panels are the concordance plot. Lower panels are the standard deviation of the log2-ratios between the normalization factors

Usage

```
plotNormFactorConcordance(object, plotFactor, normfactors = NULL)
```

Arguments

`object`	name of the object class to subset 1. NanoStringGeoMxSet, use the NanoStringGeoMxSet class
`plotFactor`	segment factor to color the plot by
`normfactors`	normalization factors from computeNormalizationFactors(). If NULL these are calculated automatically.

Examples

```
proteinData <- readRDS(file= system.file("extdata","DSP_Proteogenomics_Example_Data",
"proteinData.rds", package = "GeomxTools"))

normConcord <- plotNormFactorConcordance(object = proteinData, plotFactor = "Segment_Type")
normConcord
```

qcProteinSignal *Generate Protein QC signal boxplot figure*

Description

For use with protein data ONLY.

Usage

```
qcProteinSignal(object, neg.names = NULL)
```

Arguments

object	name of the object class to subset 1. NanoStringGeoMxSet, use the NanoStringGeoMxSet class
neg.names	names of IgGs, if NULL IgGs will be detected automatically

Value

figure function

Examples

```
proteinData <- readRDS(file= system.file("extdata","DSP_Proteogenomics_Example_Data",
"proteinData.rds", package = "GeomxTools"))

igg.names <- iggNames(proteinData)

qcFig <- qcProteinSignal(object = proteinData, neg.names = igg.names)

qcFig()
```

qcProteinSignalNames *Generate list of proteins ordered by SNR*

Description

For use with protein data ONLY.

Usage

```
qcProteinSignalNames(object, neg.names)
```

Arguments

object	name of the object class to subset 1. NanoStringGeoMxSet, use the NanoStringGeoMxSet class
neg.names	names of IgGs, if NULL IgGs will be detected automatically

Value

protein names in increasing SNR order

Examples

```
proteinData <- readRDS(file= system.file("extdata","DSP_Proteogenomics_Example_Data",
"proteinData.rds", package = "GeomxTools"))

igg.names <- iggNames(proteinData)

proteinOrder <- qcProteinSignalNames(object = proteinData, neg.names = igg.names)
```

---

readDccFile *Read DCC File*

Description

Read a NanoString GeoMx Digital Count Conversion (DCC) file.

Usage

```
readDccFile(file)
```

Arguments

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

Value

A list object with two elements:

"Header"	a data.frame object containing the protocol and sequencing information.
"Code_Summary"	a data.frame object containing the target probe counts.

Author(s)

Zhi Yang & Nicole Ortogero

See Also

readNanoStringGeoMxSet

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
dccFiles <- dir(datadir, pattern=".dcc$", full.names=TRUE)
dccData <- sapply(dccFiles[1:10], readDccFile, simplify = FALSE)
```

`readNanoStringGeoMxSet` *Read 'NanoStringGeoMxSet'*

Description

Create an instance of class `NanoStringGeoMxSet` by reading data from NanoString GeoMx Digital Count Conversion (DCC) data.

Usage

```
readNanoStringGeoMxSet(dccFiles, pkcFiles, phenoData = NULL, phenoDataFile = NULL,
                       phenoDataSheet= NULL, phenoDataDccColName = "Sample_ID",
                       phenoDataColPrefix = "", protocolDataColNames = NULL,
                       experimentDataColNames = NULL,
                       configFile = NULL, analyte = "RNA",
                       defaultPKCVersions = NULL, ...)
```

Arguments

`dccFiles`	A character vector containing the paths to the DCC files.
`pkcFiles`	A character vector representing the path to the corresponding PKC file.
`phenoData`	A data.frame containing the phenotypic data. If both `phenoData` and `phenoDataFile` are available, `phenoData` will be used.
`phenoDataFile`	Character string representing the path to the corresponding phenotypic data file. Data file can be Lab Worksheet, excel, or other delimited file. It is recommended to use the Lab Worksheet in the exact order samples are provided in.
`phenoDataSheet`	Character string representing the excel sheet name containing the phenotypic data, if needed.
`phenoDataDccColName`	Character string identifying unique sample identifier column in `phenoData` or `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.
`protocolDataColNames`	Character list of column names from `phenoDataFile` containing data about the experimental protocol or sequencing data.

`experimentDataColNames`
: Character list of column names from `phenoDataFile` containing data about the experiment's meta-data.

`configFile`
: An optional character string representing the path to the corresponding config file. This is used to ensure the only the correct PKC files are added

`analyte`
: GeoMxSet objects can only hold one analyte at a time. For studies with multiple analytes, which one should be read in? Options: RNA (default) and Protein

`defaultPKCVersions`
: Optional list of pkc file names to use as default if more than one pkc version of each module is provided.

`...`
: Optional parameters to pass to `readxl::read_xlsx` function for annotation read in

Value

An instance of the `NanoStringGeoMxSet` class.

Author(s)

Zhi Yang & Nicole Ortogero

See Also

`NanoStringGeoMxSet`

Examples

```
# Data file paths
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
dccFiles <- dir(datadir, pattern=".dcc$", full.names=TRUE)
pkc <- unzip(zipfile = file.path(datadir,  "/pkcs.zip"))
sampleAnnotationFile <- file.path(datadir, "annotations.xlsx")

dccFileColumn <- "Sample_ID"

dccSet <- readNanoStringGeoMxSet(dccFiles=dccFiles[1:10],
                                 pkcFiles=pkc,
                                 phenoDataFile=sampleAnnotationFile,
                                 phenoDataSheet="CW005",
                                 phenoDataDccColName=dccFileColumn,
                                 protocolDataColNames=c("aoi", "cell_line",
                                                        "roi_rep", "pool_rep",
                                                        "slide_rep"),
                                 experimentDataColNames="panel",
                                 phenoDataColPrefix="")

# All data
dccSet <- readNanoStringGeoMxSet(dccFiles, pkcFile = pkc,
                                   phenoDataFile = sampleAnnotationFile,
```

```
                                    phenoDataSheet="CW005")
varLabels(dccSet)

# All data with phenoData prefix
dccSetPhenoPrefix <- readNanoStringGeoMxSet(dccFiles,
                                pkcFile = pkc,
                                phenoDataFile = sampleAnnotationFile,
                                phenoDataSheet="CW005",
                                phenoDataColPrefix = "PHENO_")
varLabels(dccSetPhenoPrefix)
```

`readPKCFile` *Read PKC File*

Description

Read a NanoString Probe Kit Configuration (PKC) file.

Usage

```
readPKCFile(file, default_pkc_vers=NULL)
```

Arguments

`file`	A character string containing the path to the PKC file.
`default_pkc_vers`	Optional list of pkc file names to use as default if more than one pkc version of each module is provided.

Value

An instance of the `DataFrame` class containing columns:

`"RTS_ID"`	unique probe ID
`"TargetName"`	target or gene name
`"Module"`	PKC name
`"Negative"`	negative probe
`...`	additional columns

Author(s)

Zhi Yang & Nicole Ortogero

See Also

`readNanoStringGeoMxSet`

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
pkc <- unzip(zipfile = file.path(datadir,  "/pkcs.zip"))
PKCData <- readPKCFile(pkc)
```

`setBackgroundQCFlags` *Add background QC flags to NanoStringGeoMxSet object protocol data*

Description

Add background QC flags to NanoStringGeoMxSet object protocol data

Usage

```
setBackgroundQCFlags(object, qcCutoffs = DEFAULTS)
```

Arguments

`object`	name of the NanoStringGeoMxSet object to perform QC on
`qcCutoffs`	a list of qc cutoffs to use 1. minNegativeCount, numeric to flag segments with less than this number of counts 2. maxNTCCount, numeric to flag segments with more than this number of NTC counts

Value

`NanoStringGeoMxSet` object with `QCFlags` data frame appended to `protocolData`

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
setBackgroundQCFlags(demoData[,1:10],
                qcCutoffs=list(minNegativeCount=10,
                               maxNTCCount=60))
```

`setBioProbeQCFlags` *Add probe QC flags to NanoStringGeoMxSet object feature data*

Description

Add probe QC flags to NanoStringGeoMxSet object feature data

Usage

```
setBioProbeQCFlags(object, qcCutoffs = DEFAULTS, removeLocalOutliers = TRUE)
```

Arguments

`object`	name of the NanoStringGeoMxSet object to perform QC on
`qcCutoffs`	a list of qc cutoffs to use 1. minProbeRatio, numeric between 0 and 1 to flag probes that have (geomean probe in all segments) / (geomean probes within target) less than or equal to this ratio 2. percentFailGrubbs, numeric to flag probes that fail Grubb's test in greater than or equal this percent of segments
`removeLocalOutliers`	boolean to determine if local outliers should be excluded from `exprs` matrix

Value

`NanoStringGeoMxSet` object with `QCFlags` data frame appended to `protocolData`

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
demoData <- shiftCountsOne(demoData, elt="exprs", useDALogic=TRUE)
setBioProbeQCFlags(demoData[,1:10],
                   qcCutoffs=list(minProbeRatio=0.1,
                                  percentFailGrubbs=20),
                   removeLocalOutliers=TRUE)
```

setGeoMxQCFlags *Add GeoMx segment QC flags to NanoStringGeoMxSet object protocol data*

Description

Add GeoMx segment QC flags to NanoStringGeoMxSet object protocol data

Usage

```
setGeoMxQCFlags(object, qcCutoffs = DEFAULTS)
```

Arguments

object	name of the NanoStringGeoMxSet object to perform QC on
qcCutoffs	a list of qc cutoffs to use 1. minNuclei, numeric to flag segments with less than this number of nuclei 2. minArea, numeric to flag segments with less than this um^2 area

Value

NanoStringGeoMxSet object with QCFlags data frame appended to protocolData

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
setGeoMxQCFlags(demoData[,1:10],
                qcCutoffs=list(minNuclei=16000,
                               minArea=20))
```

setQCFlags,NanoStringGeoMxSet-method *Add QC flags to feature and protocol data simultaneously*

Description

Add QC flags to feature and protocol data simultaneously

Usage

```
## S4 method for signature 'NanoStringGeoMxSet'
setQCFlags(object, qcCutoffs = DEFAULTS, ...)
```

Arguments

object	name of the object class to perform QC on 1. NanoStringGeoMxSet, use the NanoStringGeoMxSet class
qcCutoffs	list of cutoffs and thresholds to use for QC
...	optional parameters to pass

Value

the object that QC was performed on

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
setQCFlags(demoData[,1:10])
```

---

setSegmentQCFlags *Add segment QC flags to protocol data*

Description

Add segment QC flags to protocol data

Usage

```
setSegmentQCFlags(object, qcCutoffs = DEFAULTS)
```

Arguments

object	name of the object class to perform QC on 1. NanoStringGeoMxSet, use the NanoStringGeoMxSet class
qcCutoffs	list of cutoffs and thresholds to use for QC

Value

`NanoStringGeoMxSet` object with `QCFlags` data frame appended to `protocolData`

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
setSegmentQCFlags(demoData[,1:10],
                  qcCutoffs=list(minSegmentReads=1000,
                                 percentAligned=80,
                                 percentSaturation=50,
                                 minNegativeCount=10,
                                 maxNTCCount=60,
                                 minNuclei=16000,
                                 minArea=20))
```

---

setSeqQCFlags *Add sequencing QC flags to NanoStringGeoMxSet object protocol data*

Description

Add sequencing QC flags to NanoStringGeoMxSet object protocol data

Usage

```
setSeqQCFlags(object, qcCutoffs = DEFAULTS)
```

Arguments

object	name of the NanoStringGeoMxSet object to perform QC on
qcCutoffs	a list of qc cutoffs to use 1. minSegmentReads, numeric to flag segments with less than this number of reads 2. percentAligned, numeric to flag segments with less than this percent of aligned reads 3. percentSaturation, numeric to flag segments with less than this percent of sequencing saturation

Value

NanoStringGeoMxSet object with QCFlags data frame appended to protocolData

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
setSeqQCFlags(demoData[,1:10],
                qcCutoffs=list(minSegmentReads=1000,
```

```
                                percentAligned=80,
                                percentSaturation=50))
```

`shiftCountsOne` *Add one to all counts in an expression matrix*

Description

Add one to all counts in an expression matrix

Usage

```
shiftCountsOne(object, elt = "exprs", useDALogic = FALSE)
```

Arguments

`object`	name of the NanoStringGeoMxSet object
`elt`	expression matrix element in `assayDataElement` to shift all counts by
`useDALogic`	boolean to use the same logic in DA (impute 0s to 1s) or set to FALSE to shift all counts by 1

Value

object of NanoStringGeoMxSet class

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
shiftCountsOne(demoData)
```

`summarizeNegatives` *Calculate negative probe summary stats*

Description

Calculate negative probe summary stats

Usage

```
summarizeNegatives(object, functionList = c())
```

Arguments

object	name of the NanoStringGeoMxSet object to summarize
functionList	optional list of additional functions to calculate negative probe stats, list element names should correspond to expected stat column header

Value

a NanoStringGeoMxSet object with negative probe summary stats appended to sample data

Examples

```
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
demoData <- readRDS(file.path(datadir, "/demoData.rds"))
demoData <-
    summarizeNegatives(demoData,
                       functionList=c(mean=mean, min=min, max=max))
```

---

updateGeoMxSet *Update GeoMxSet object to current version*

Description

Update GeoMxSet object to current version

Usage

```
updateGeoMxSet(object)
```

Arguments

object	GeoMxSet object to update

Value

updated GeoMxSet object

writeNanoStringGeoMxSet *write 'NanoStringGeoMxSet'*

Description

Take an instance of class NanoStringGeoMxSet and write NanoString GeoMx Digital Count Conversion (DCC) data.

Usage

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

Arguments

x	A NanoStringGeoMxSet object.
dir	A directory path to save all the DCC files.

Author(s)

Zhi Yang & Nicole Ortogero

See Also

NanoStringGeoMxSet

Examples

```
# Data file paths
datadir <- system.file("extdata", "DSP_NGS_Example_Data",
                       package="GeomxTools")
dccFiles <- dir(datadir, pattern=".dcc$", full.names=TRUE)
pkc <- unzip(zipfile = file.path(datadir,  "/pkcs.zip"))
sampleAnnotationFile <- file.path(datadir, "annotations.xlsx")

dccFileColumn <- "Sample_ID"

dccSet <- readNanoStringGeoMxSet(dccFiles=dccFiles,
                                 pkcFiles=pkc,
                                 phenoDataFile=sampleAnnotationFile,
                                 phenoDataSheet="CW005",
                                 phenoDataDccColName=dccFileColumn,
                                 protocolDataColNames=c("aoi", "cell_line",
                                                        "roi_rep", "pool_rep",
                                                        "slide_rep"),
                                 experimentDataColNames="panel",
                                 phenoDataColPrefix="")

# All data
```

```
writeNanoStringGeoMxSet(dccSet)
```

Index