

# Package ‘GeomxTools’

May 5, 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** 3.17.0

**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

**git\_url** <https://git.bioconductor.org/packages/GeomxTools>

**git\_branch** devel

**git\_last\_commit** 37ea812

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-04

**Author** Maddy Griswold [cre, aut],  
Nicole Ortogero [aut],  
Zhi Yang [aut],  
Ronaly Vitancol [aut],  
David Henderson [aut]

**Maintainer** Maddy Griswold <mgriswold@nanosttring.com>

## Contents

aggregateCounts . . . . .	3
as.Seurat . . . . .	3
as.SpatialExperiment . . . . .	4
checkQCFlags . . . . .	5
checkQCFlags,NanoStringGeoMxSet-method . . . . .	6
compareToConfig . . . . .	7
computeNormalizationFactors . . . . .	7
countsShiftedByOne . . . . .	8
hkNames . . . . .	8
iggNames . . . . .	9
logtBase . . . . .	9
mixedModelIDE . . . . .	10
NanoStringGeoMxSet-class . . . . .	11
ngeoMean . . . . .	13
ngeoSD . . . . .	14
normalize,NanoStringGeoMxSet-method . . . . .	14
plotConcordance . . . . .	15
plotNormFactorConcordance . . . . .	16
qcProteinSignal . . . . .	16
qcProteinSignalNames . . . . .	17
readDccFile . . . . .	18
readNanoStringGeoMxSet . . . . .	18
readPKCFile . . . . .	20
setBackgroundQCFlags . . . . .	21
setBioProbeQCFlags . . . . .	22
setGeoMxQCFlags . . . . .	23
setQCFlags,NanoStringGeoMxSet-method . . . . .	23
setSegmentQCFlags . . . . .	24
setSeqQCFlags . . . . .	25
shiftCountsOne . . . . .	25
summarizeNegatives . . . . .	26
updateGeoMxSet . . . . .	27
writeNanoStringGeoMxSet . . . . .	27

**Index**

**29**

---

aggregateCounts	<i>Aggregate probe counts to target level for feature data</i>
-----------------	--

---

**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	<i>Convert GeoMxSet Object to SeuratObject</i>
-----------	--

---

**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	<i>Check QC Flags in the GeoMxSet and removes the probe or sample from the object</i>
--------------	---

---

**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	<i>Compare given PKC probes to probes in config file</i>
-----------------	--

---

**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	<i>Generate normalization factors</i>
-----------------------------	---------------------------------------

---

**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



**Value**

names of HKs

---

iggNames	<i>Return the IgG negative controls for protein</i>
----------	---

---

**Description**

Return the IgG negative controls for protein

**Usage**

iggNames(object)

**Arguments**

object            name of the NanoStringGeoMxSet object

**Value**

names of IgGs

---

logtBase	<i>Get take the log of a numeric vector</i>
----------	---

---

**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 <a href="#">AnnotatedDataFrame</a> containing the phenotypic data of areas of interest.
featureData	An <a href="#">AnnotatedDataFrame</a> containing target information; target name, accession number, functional groups, etc.
experimentData	An optional <a href="#">MIAME</a> instance with meta-data about the experiment.
annotation	A character string for the PKC file(s).
protocolData	An <a href="#">AnnotatedDataFrame</a> 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 <a href="#">SignatureSet</a> object containing signature definitions.
design	An optional one-sided formula representing the experimental design based on columns from <a href="#">phenoData</a>
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 <a href="#">ExpressionSet</a> .

**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(object)))`.

`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	<i>Get the geometric standard deviation of a vector</i>
--------	---

---

**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	<i>normalize</i>
--------------------------------------	------------------

---

**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	<i>Generate concordance figure of targets based on user provided factors</i>
-----------------	--

---

**Description**

Upper panels are the concordance plot. Lower panels are the standard deviation of the log<sub>2</sub>-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 <ol style="list-style-type: none"> <li>1. NanoStringGeoMxSet, use the NanoStringGeoMxSet class</li> </ol>
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	<i>Read DCC File</i>
-------------	----------------------

---

**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	<i>Read 'NanoStringGeoMxSet'</i>
------------------------	----------------------------------

---

**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 <ol style="list-style-type: none"> <li>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</li> <li>2. percentFailGrubbs, numeric to flag probes that fail Grubb's test in greater than or equal this percent of segments</li> </ol>
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	<i>Add GeoMx segment QC flags to NanoStringGeoMxSet object protocol data</i>
-----------------	--

---

**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 <ol style="list-style-type: none"> <li>1. minNuclei, numeric to flag segments with less than this number of nuclei</li> <li>2. minArea, numeric to flag segments with less than this <math>\mu\text{m}^2</math> area</li> </ol>

**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	<i>Add QC flags to feature and protocol data simultaneously</i>
--------------------------------------	---

---

**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,
                                maxNTCCCount=60,
                                minNuclei=16000,
                                minArea=20))
```

---

setSeqQCFlags	<i>Add sequencing QC flags to NanoStringGeoMxSet object protocol data</i>
---------------	---

---

**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 <ol style="list-style-type: none"> <li>1. minSegmentReads, numeric to flag segments with less than this number of reads</li> <li>2. percentAligned, numeric to flag segments with less than this percent of aligned reads</li> <li>3. percentSaturation, numeric to flag segments with less than this percent of sequencing saturation</li> </ol>

**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	<i>Add one to all counts in an expression matrix</i>
----------------	--

---

**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	<i>Update GeoMxSet object to current version</i>
----------------	--

---

**Description**

Update GeoMxSet object to current version

**Usage**

```
updateGeoMxSet(object)
```

**Arguments**

object	GeoMxSet object to update
--------	---------------------------

**Value**

updated GeoMxSet object

---

writeNanoStringGeoMxSet	<i>write 'NanoStringGeoMxSet'</i>
-------------------------	-----------------------------------

---

**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

- \* **NanoStringGeoMxSet**
  - readNanoStringGeoMxSet, [18](#)
  - writeNanoStringGeoMxSet, [27](#)
- \* **classes**
  - NanoStringGeoMxSet-class, [11](#)
- \* **file**
  - readDccFile, [18](#)
  - readNanoStringGeoMxSet, [18](#)
  - readPKCFile, [20](#)
  - writeNanoStringGeoMxSet, [27](#)
- \* **manip**
  - readDccFile, [18](#)
  - readNanoStringGeoMxSet, [18](#)
  - readPKCFile, [20](#)
  - writeNanoStringGeoMxSet, [27](#)
- \* **methods**
  - NanoStringGeoMxSet-class, [11](#)
- \* **objects**
  - as.Seurat, [3](#)
- aggregateCounts, [3](#)
- analyte (NanoStringGeoMxSet-class), [11](#)
- analyte, NanoStringGeoMxSet-method (NanoStringGeoMxSet-class), [11](#)
- AnnotatedDataFrame, [11](#)
- as.Seurat, [3](#)
- as.SpatialExperiment, [4](#)
- checkQCFlags, [5](#)
- checkQCFlags, NanoStringGeoMxSet-method, [6](#)
- class:NanoStringGeoMxSet (NanoStringGeoMxSet-class), [11](#)
- coerce, ExpressionSet, NanoStringGeoMxSet-method (NanoStringGeoMxSet-class), [11](#)
- compareToConfig, [7](#)
- computeNormalizationFactors, [7](#)
- countsShiftedByOne, [8](#)
- DataFrame, [21](#)
- design, NanoStringGeoMxSet-method (NanoStringGeoMxSet-class), [11](#)
- design<- , NanoStringGeoMxSet, ANY-method (NanoStringGeoMxSet-class), [11](#)
- design<- , NanoStringGeoMxSet, formula-method (NanoStringGeoMxSet-class), [11](#)
- design<- , NanoStringGeoMxSet, NULL-method (NanoStringGeoMxSet-class), [11](#)
- dimLabels, NanoStringGeoMxSet-method (NanoStringGeoMxSet-class), [11](#)
- dimLabels<- , NanoStringGeoMxSet, character-method (NanoStringGeoMxSet-class), [11](#)
- ExpressionSet, [11](#), [12](#)
- featureType (NanoStringGeoMxSet-class), [11](#)
- featureType, NanoStringGeoMxSet-method (NanoStringGeoMxSet-class), [11](#)
- featureType<- (NanoStringGeoMxSet-class), [11](#)
- featureType<- , NanoStringGeoMxSet, character-method (NanoStringGeoMxSet-class), [11](#)
- hkNames, [8](#)
- iggNames, [9](#)
- logtBase, [9](#)
- MIAME, [11](#)
- mixedModelDE, [10](#)
- NanoStringGeoMxSet, [18–20](#), [27](#)
- NanoStringGeoMxSet (NanoStringGeoMxSet-class), [11](#)
- NanoStringGeoMxSet, environment-method (NanoStringGeoMxSet-class), [11](#)
- NanoStringGeoMxSet, ExpressionSet-method (NanoStringGeoMxSet-class), [11](#)
- NanoStringGeoMxSet, matrix-method (NanoStringGeoMxSet-class), [11](#)
- NanoStringGeoMxSet, missing-method (NanoStringGeoMxSet-class), [11](#)
- NanoStringGeoMxSet, NanoStringGeoMxSet-method (NanoStringGeoMxSet-class), [11](#)
- NanoStringGeoMxSet-class, [11](#)
- ngeoMean, [13](#)
- ngeoSD, [14](#)

- normalize, NanoStringGeoMxSet-method, [14](#)
- phenoData, [11](#), [12](#)
- plotConcordance, [15](#)
- plotNormFactorConcordance, [16](#)
- qcProteinSignal, [16](#)
- qcProteinSignalNames, [17](#)
- readDccFile, [18](#)
- readNanoStringGeoMxSet, [12](#), [18](#), [18](#), [21](#)
- readPKCFile, [20](#)
- sData (NanoStringGeoMxSet-class), [11](#)
- sData, NanoStringGeoMxSet-method  
(NanoStringGeoMxSet-class), [11](#)
- setBackgroundQCFlags, [21](#)
- setBioProbeQCFlags, [22](#)
- setGeoMxQCFlags, [23](#)
- setQCFlags, NanoStringGeoMxSet-method, [23](#)
- setSegmentQCFlags, [24](#)
- setSeqQCFlags, [25](#)
- SeuratObject::as.Seurat, [4](#)
- shiftCountsOne, [25](#)
- show, NanoStringGeoMxSet-method  
(NanoStringGeoMxSet-class), [11](#)
- signatureGroups  
(NanoStringGeoMxSet-class), [11](#)
- signatureGroups, NanoStringGeoMxSet-method  
(NanoStringGeoMxSet-class), [11](#)
- signatures (NanoStringGeoMxSet-class), [11](#)
- signatures, NanoStringGeoMxSet-method  
(NanoStringGeoMxSet-class), [11](#)
- signatures<-  
(NanoStringGeoMxSet-class), [11](#)
- signatures<- , NanoStringGeoMxSet, SignatureSet-method  
(NanoStringGeoMxSet-class), [11](#)
- signatureScores  
(NanoStringGeoMxSet-class), [11](#)
- signatureScores, NanoStringGeoMxSet-method  
(NanoStringGeoMxSet-class), [11](#)
- SignatureSet, [11](#), [12](#)
- summarizeNegatives, [26](#)
- svarLabels (NanoStringGeoMxSet-class), [11](#)
- svarLabels, NanoStringGeoMxSet-method  
(NanoStringGeoMxSet-class), [11](#)
- updateGeoMxSet, [27](#)
- updateObject, NanoStringGeoMxSet-method  
(NanoStringGeoMxSet-class), [11](#)
- writeNanoStringGeoMxSet, [27](#)