Package 'normScore'

September 10, 2026

Title Evaluation and Ranking of Normalization Methods for Proteomics Data

Version 0.99.1

Description Provides tools to evaluate and rank normalization methods for omics datasets using a composite score derived from multiple performance metrics. The package is designed to support systematic benchmarking and comparison of normalization strategies across datasets and experimental settings. It also includes utilities for summarizing results and visualizing normalization performance.

License GPL-2

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

Depends R (>= 4.6)

Imports stats, ggplot2, ggpubr, boot, MASS, rlang, withr

Suggests knitr, rmarkdown, testthat (>= 3.0.0), limma, NormalyzerDE, SummarizedExperiment, S4Vectors, methods, BiocStyle, BiocManager

VignetteBuilder knitr

BuildVignettes true

Config/testthat/edition 3

biocViews Software, Proteomics, Normalization, QualityControl, Preprocessing, MultipleComparison

URL https://github.com/juliagcurras/normScore, https://juliagcurras.github.io/normScore/

BugReports https://https://github.com/juliagcurras/normScore/issues

Config/roxygen2/version 8.1.0

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

git_branch devel

git_last_commit 38e8616

git_last_commit_date 2026-08-25

Repository Bioconductor 3.24

Date/Publication 2026-09-09

Author Julia García Currás [aut, cre] (ORCID:
<https://orcid.org/0009-0002-6354-5035>),
Axencia Galega de Innovación (GAIN), Xunta de Galicia [fnd] (Industrial
Doctorate Grant 2022-2026, Ref. 23_IN606D_2022_2707220)

Maintainer Julia García Currás <julia.gcurras@udc.es>

Contents

`normScore` *Rank Normalization Methods Using a Composite Scoring Framework*

Description

Computes a composite score to rank normalization methods based on multiple quality metrics derived from normalized data matrices.

The function evaluates each normalization method using six criteria: pooled coefficient of variation (PCV), within-group sample correlation, MA-plot trend deviation, mean-SD trend deviation, relative log expression (RLE) consistency, and total intensity consistency. These metrics are scaled, combined into a final score, and used to rank the normalization methods.

In addition, a correction factor derived from the coefficient of variation of raw sample intensities is applied to the `"Log"` normalization method. Optionally, bootstrap resampling is used to estimate mean composite scores and percentile-based confidence intervals.

Usage

```
normScore(
  normalizedDataList,
  groupData,
  rawData,
  refGroup = NULL,
  altGroup = NULL,
  returnDetails = TRUE,
  nBoot = 1000
)
```

Arguments

normalizedDataList	Named list of normalized numeric matrices or data frames. Each element represents one normalization method, with rows corresponding to features (e.g. proteins) and columns to samples.
groupData	data.frame or matrix describing the sample-group assignment. It must contain two columns, which are interpreted as sample names and group labels.
rawData	data.frame or matrix of raw numeric intensities, with rows representing features and columns representing samples.
refGroup	character or NULL. Reference group used for the MA-plot metric. If NULL, the last group in groupData is used.
altGroup	character or NULL. Alternative group used for the metric. If NULL, the first group in groupData is used.
returnDetails	logical. If TRUE, only the final ranking is MA-plot returned. If FALSE, detailed scores, bootstrap results, and a summaryplot are also returned. Default is FALSE.
nBoot	integer. Number of bootstrap resamples used to estimate mean normScore values and confidence intervals. Default is 1000.

Details

The function computes the following six item scores for each normalization method:

1. **Item1**: mean pooled coefficient of variation across groups (PVC).
2. **Item2**: within-group Spearman correlation summary, transformed so that lower values indicate better performance (Correlation).
3. **Item3**: shape-corrected area-based deviation from the expected MA trend (logFC = 0), based on MAplot.
4. **Item4**: area-based deviation of the mean-SD trend from horizontality, based on meanSD plot.
5. **Item5**: RLE-based mean absolute percentage error relative to 1.
6. **Item6**: quantile-based total intensity consistency metric.

Each item is scaled to the range 0 to 1 using min-max scaling and then summed into a total score. Lower scores indicate better normalization performance.

A correction factor based on the coefficient of variation of total raw sample intensities is applied only to the normalization method named "Log".

When returnDetails = FALSE, bootstrap resampling of the six item scores is performed.The results can be visualized using plotBootstrapNormScore.

Value

If returnDetails = TRUE, returns a list with:

finalRanking Named numeric vector with the final normalization ranking.

If returnDetails = FALSE, returns a list with:

finalRanking Named numeric vector with the final normalization ranking.

detailRanking data.frame with scaled item-wise scores, total score, and corrected total score for each normalization method.

bootstrapScore data.frame with bootstrap mean scores and 95\ percentile confidence intervals for each normalization method.

graphic A ggplot2 object showing the bootstrap mean scores and confidence intervals.

See Also

plotNormScoreDiagnostics, plotBootstrapNormScore, simulateData

Examples

```
# Simulate proteomic data
simData <- simulateData(nProteins = 1000)

# Artificial normalization (just adding shifts to raw data)
normalizedDataList <- list(
    Norm1 = simData$logData + 0.1,
    Norm2 = simData$logData + 1,
    Norm3 = simData$logData - 1,
    Norm4 = simData$logData * 1.1,
    Norm5 = simData$logData * 0.9,
    Norm6 = simData$logData * runif(ncol(simData$logData), 0.8, 1.2)
)

# Compute ranking
result <- normScore(
    normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    returnDetails = TRUE,
    nBoot = 100
)
result
```

---

plotBootstrapNormScore

Plot bootstrap normScore results

Description

Generates a point-range plot showing the mean bootstrap normScore and its 95% confidence interval for each normalization method.

Usage

```
plotBootstrapNormScore(x)
```

Arguments

x	A normScore result object containing a bootstrapScore element. This element should be a data frame with normalization names, mean normScore values, and lower and upper 95% confidence limits. It can beobtained by setting returnDetails = TRUE at normScore main function.

Value

A ggplot object showing bootstrap mean normScore values and 95% confidence intervals for each normalization method.

Examples

```
bootstrapScore <- data.frame(
    normalization = c("Norm1", "Norm2", "Norm3"),
    meanNormScore = c(0.8, 1.2, 1.5),
    ll95 = c(0.6, 1.0, 1.2),
    ul95 = c(1.0, 1.4, 1.8)
)

result <- list(bootstrapScore = bootstrapScore)

plotBootstrapNormScore(result)
```

---

plotDiagnosticMA *Plot MA diagnostic plots (Item 3)*

Description

Generates MA plots for each normalization method using the selected reference and alternative groups. These plots are used to assess intensity-dependent differences between groups.

Important! First run `validateNormScoreInput()` function and them use the correspondent outputs to plot individual diagnostic graphs as it is shown in the example below.

Usage

```
plotDiagnosticMA(normalizedDataList, groupData, refGroup, altGroup)
```

Arguments

`normalizedDataList`	A named list of normalized data matrices.
`groupData`	A data frame with sample names and group labels.
`refGroup`	Character string indicating the reference group.
`altGroup`	Character string indicating the alternative group.

Value

An arranged ggplot object containing one MA plot per normalization method.

Examples

```
# Simulate proteomic data
simData <- simulateData(nProteins = 1000, propDE = 0.5)

# Adding variations to log raw data to simulate different normalization
# situations
normalizedDataList <- list(
    Log = simData$logData,
    Norm1 = cbind(simData$logData[, 1:20] +
        -0.8 * (rowMeans(simData$logData, na.rm = TRUE) -
            mean(rowMeans(simData$logData, na.rm = TRUE))),
        simData$logData[, 21:40]),
    Norm2 = cbind(simData$logData[, 1:20] +
        0.3 * (rowMeans(simData$logData, na.rm = TRUE) -
            mean(rowMeans(simData$logData, na.rm = TRUE))),
        simData$logData[, 21:40])
)

# Validating inputs for diagnostic graphs
inputs <- validateNormScoreInput(
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    fromMainFunction = FALSE
)

# Plot MA diagnostic plots
plotDiagnosticMA(
    normalizedDataList = inputs$normalizedDataList,
    groupData = inputs$groupData,
    refGroup = inputs$refGroup,
    altGroup = inputs$altGroup
) # Best to Worst: Log > Norm2 > Norm1
```

---

plotDiagnosticMeanSD *Plot mean-SD diagnostic plots (Item 4)*

Description

Generates diagnostic plots showing the relationship between sample order by mean intensity and standard deviation for each normalization method.

Usage

```
plotDiagnosticMeanSD(normalizedDataList)
```

Arguments

normalizedDataList	A named list of normalized data matrices.

Details

Important! First run `validateNormScoreInput()` function and them use the correspondent outputs to plot individual diagnostic graphs as it is shown in the example below.

Value

An arranged ggplot object containing one plot per normalization method.

Examples

```
# Simulate proteomic data
simData <- simulateData(nProteins = 1000, sampleShiftSd = 0.5,
    sampleShiftCap = 1, sampleSdStrength = -0.5, sampleSdRho = 1.5)

# Artificial normalization (just adding shifts to raw data)
normalizedDataList <- list(
    Log = simData$logData,
    Norm1 = simData$logData * runif(ncol(simData$logData), 0.7, 1.2),
    Norm2 = simData$logData * runif(ncol(simData$logData), 0.8, 1.1)
)

# Validating inputs for diagnostic graphs
inputs <- validateNormScoreInput(
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    fromMainFunction = FALSE
)

# Plot mean-SD diagnostic plots
plotDiagnosticMeanSD(
    normalizedDataList = inputs$normalizedDataList
) # Best to Worst: Norm1 > Norm2 > Log
```

---

plotDiagnosticPCV *Plot pooled coefficient of variation (Item 1)*

Description

Generates the diagnostic plot associated with the PCV item. PCV values are computed for each normalization method and group, and displayed as point-range or boxplot-based summaries depending on the number of groups.

Usage

```
plotDiagnosticPCV(normalizedDataList, groupData)
```

Arguments

`normalizedDataList`	A named list of normalized data matrices.
`groupData`	A data frame with sample names and group labels.

Details

Important! First run `validateNormScoreInput()` function and them use the correspondent outputs to plot individual diagnostic graphs as it is shown in the example below.

Value

A `ggplot` or arranged ggplot object.

Examples

```
# Simulate proteomic data with considerable systematic bias
simData <- simulateData(nProteins = 1000, sampleShiftSd = 1,
    sampleShiftCap = 0.9)

# Adding variations to log raw data to simulate different normalization
# situations
normalizedDataList <- list(
    Log = simData$logData,
    Norm1 = simData$logData +
        rnorm(n = ncol(simData$logData), mean = 5, sd = 0.5),
    Norm2 = sweep(simData$logData, 2,
        apply(simData$logData, 2, median, na.rm = TRUE) -
        median(as.matrix(simData$logData), na.rm = TRUE), "-")
)

# Validating inputs for diagnostic graphs
inputs <- validateNormScoreInput(
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    fromMainFunction = FALSE
)

# Plot pooled coefficient of variation
plotDiagnosticPCV(
    normalizedDataList = inputs$normalizedDataList,
    groupData = inputs$groupData
) # Best to Worst: Norm2 > Norm1 > Log
```

`plotDiagnosticRLE` *Plot RLE distributions (Item 5)*

Description

Generates RLE distribution plots for each normalization method. These plots are used to inspect whether sample-wise RLE distributions are centered and comparable after normalization.

Usage

```
plotDiagnosticRLE(normalizedDataList)
```

Arguments

normalizedDataList
: A named list of normalized data matrices.

Details

Important! First run `validateNormScoreInput()` function and them use the correspondent outputs to plot individual diagnostic graphs as it is shown in the example below.

Value

An arranged ggplot object containing one RLE plot per normalization method.

Examples

```
# Simulate proteomic data with considerable systematic bias
simData <- simulateData(nProteins = 1000, sampleShiftSd = 1,
    sampleShiftCap = 0.9)

# Adding variations to log raw data to simulate different normalization
# situations
normalizedDataList <- list(
    Log = simData$logData,
    Norm1 = simData$logData +
        rnorm(n = ncol(simData$logData), mean = 5, sd = 0.5),
    Norm2 = sweep(simData$logData, 2,
        apply(simData$logData, 2, median, na.rm = TRUE) -
        median(as.matrix(simData$logData), na.rm = TRUE), "-")
)

# Validating inputs for diagnostic graphs
inputs <- validateNormScoreInput(
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    fromMainFunction = FALSE
)

# Plot RLE distributions
plotDiagnosticRLE(
    normalizedDataList = inputs$normalizedDataList
) # Best to Worst: Norm2 > Log = Norm1
```

---

plotDiagnosticSystematicBias

Plot functions for original criteria

Description

This file contains plot functions used as outputs for interpreting the detailed ranking results. Plot raw total intensity across samples (ITEM 0)

Usage

```
plotDiagnosticSystematicBias(data)
```

Arguments

data	A raw intensity matrix or data frame with proteins in rows and samples in columns.

Details

Generates the diagnostic plot associated with the systematic bias penalty item (item 0). Raw total intensity is computed for each sample and displayed as a bar plot.

Important! First run `validateNormScoreInput()` function and them use the correspondent outputs to plot individual diagnostic graphs as it is shown in the example below.

Value

A `ggplot` object.

Examples

```
# Simulate proteomic data with considerable systematic bias
simData <- simulateData(nProteins = 1000, sampleShiftSd = 1,
    sampleShiftCap = 0.9)

# Adding variations to log raw data to simulate different normalization
# situations
normalizedDataList <- list(
    Log = simData$logData,
    Norm1 = simData$logData +
        rnorm(n = ncol(simData$logData), mean = 5, sd = 0.5),
    Norm2 = sweep(simData$logData, 2,
        apply(simData$logData, 2, median, na.rm = TRUE) -
        median(as.matrix(simData$logData), na.rm = TRUE), "-")
)

# Validating inputs for diagnostic graphs
inputs <- validateNormScoreInput(
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    fromMainFunction = FALSE
)

# Plot Systematic Bias Penalty
plotDiagnosticSystematicBias(data = inputs$rawData)
```

`plotDiagnosticTotalIntensity` *Plot log-intensity distributions (Item 6)*

Description

Generates sample-wise intensity distribution plots for each normalization method. These plots are used to visually inspect global distributional consistency across samples.

Usage

```
plotDiagnosticTotalIntensity(normalizedDataList)
```

Arguments

`normalizedDataList` A named list of normalized data matrices.

Details

Important! First run `validateNormScoreInput()` function and them use the correspondent outputs to plot individual diagnostic graphs as it is shown in the example below.

Value

An arranged ggplot object containing one intensity distribution plot per normalization method.

Examples

```
# Simulate proteomic data with considerable systematic bias
simData <- simulateData(nProteins = 1000, sampleShiftSd = 1,
    sampleShiftCap = 0.9)

# Adding variations to log raw data to simulate different normalization
# situations
normalizedDataList <- list(
    Log = simData$logData,
    Norm1 = simData$logData +
        rnorm(n = ncol(simData$logData), mean = 5, sd = 0.5),
    Norm2 = sweep(simData$logData, 2,
        apply(simData$logData, 2, median, na.rm = TRUE) -
        median(as.matrix(simData$logData), na.rm = TRUE), "-")
)

# Validating inputs for diagnostic graphs
inputs <- validateNormScoreInput(
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    fromMainFunction = FALSE
)
```

```
# Plot log-intensity distributions
plotDiagnosticTotalIntensity(
    normalizedDataList = inputs$normalizedDataList
) # Best to Worst: Norm2 > Norm1 = Log
```

`plotDiagnosticWithinGroupCor` *Plot within-group correlations (Item 2)*

Description

Generates the diagnostic plot associated with the correlation item. The plot displays the distribution of within-group sample correlations for each normalization method.

Usage

```
plotDiagnosticWithinGroupCor(normalizedDataList, groupData)
```

Arguments

`normalizedDataList`	A named list of normalized data matrices.
`groupData`	A data frame with sample names and group labels.

Details

Important! First run `validateNormScoreInput()` function and them use the correspondent outputs to plot individual diagnostic graphs as it is shown in the example below.

Value

A `ggplot` object.

Examples

```
# Simulate proteomic data with considerable systematic bias
simData <- simulateData(nProteins = 1000, sampleShiftSd = 1,
    sampleShiftCap = 0.9)

# Adding variations to log raw data to simulate different normalization
# situations
normalizedDataList <- list(
    Log = simData$logData,
    Norm1 = simData$logData +
        rnorm(n = ncol(simData$logData), mean = 5, sd = 0.5),
    Norm2 = sweep(simData$logData, 2,
        apply(simData$logData, 2, median, na.rm = TRUE) -
        median(as.matrix(simData$logData), na.rm = TRUE), "-")
)

# Validating inputs for diagnostic graphs
inputs <- validateNormScoreInput(
```

```
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    fromMainFunction = FALSE
)

# Plot within-group correlations
plotDiagnosticWithinGroupCor(
    normalizedDataList = inputs$normalizedDataList,
    groupData = inputs$groupData
) # Best to Worst: Log > Norm2 > Norm1
```

`plotNormScoreDiagnostics` *Plot normScore diagnostic items*

Description

Generates the set of diagnostic plots associated with the individual normScore evaluation items. These plots provide a visual interpretation of the criteria used to assess normalization performance, including raw total intensity, pooled coefficient of variation, within-group correlation, MA plots, mean-SD trends, RLE distributions, and intensity distributions.

Usage

```
plotNormScoreDiagnostics(
  normalizedDataList,
  groupData,
  rawData,
  refGroup = NULL,
  altGroup = NULL
)
```

Arguments

`normalizedDataList`	A named list of normalized data matrices or data frames. Each element should contain proteins in rows and samples in columns.
`groupData`	A data frame containing sample-group annotation. The first column is assumed to contain sample names and the second column group labels.
`rawData`	A matrix or data frame containing raw intensity values, with proteins in rows and samples in columns.
`refGroup`	Character string indicating the reference group used for the MA-plot diagnostic. If `NULL`, it is automatically selected.
`altGroup`	Character string indicating the alternative group used for the MA-plot diagnostic. If `NULL`, it is automatically selected.

Details

The function validates and aligns the input data before plotting. If only one group is provided, the MA-plot diagnostic is skipped because it requires a comparison between two groups.

Value

A named list of diagnostic plots. Each element corresponds to one normScore item. If the input contains a single group, the MA-plot element is returned as NULL.

Examples

```
# Simulate proteomic data
simData <- simulateData(nProteins = 1000)

# Normalyze ysing NormalyzerDE package
normalizedDataList <- list(
    Norm1 = simData$logData + 0.1,
    Norm2 = simData$logData + 1,
    Norm3 = simData$logData - 1,
    Norm4 = simData$logData * 1.1,
    Norm5 = simData$logData * 0.9,
    Norm6 = simData$logData * runif(ncol(simData$logData), 0.8, 1.2)
)

# Compute ranking
plotNormScoreDiagnostics(
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL
)
```

---

simulateData *Simulate label-free proteomics-like two-group data*

Description

Simulates two-group quantitative omics data with controlled signal structure, including abundance-dependent variance, differential expression, sample correlation, global shifts, mean-SD dependence, and optional MNAR missingness.

Usage

```
simulateData(
  nProteins = 10000,
  nPerGroup = 20,
  muMean = 18,
  muSd = 1.2,
  muClip = c(15, 25),
  rhoWithin = 0.85,
  rhoBetween = 0.55,
```

```
  loadingSd = 0.25,
  sigmaHi = 0.05,
  sigmaLo = 0.4,
  gammaSigma = 2.5,
  propDE = 0.35,
  logFCSd = 1,
  logFCMean = 0,
  heteroLogFC = TRUE,
  fcHi = 0.55,
  fcLo = 2.5,
  gammaFC = 7,
  sampleShiftSd = 0,
  sampleShiftCap = 0.2,
  sampleSdStrength = 0,
  sampleSdRho = 0.8,
  sampleSdCap = 0.35,
  addMissing = TRUE,
  targetMissing = 0.001,
  kMnar = 1.2,
  missingBySampleSd = 0.05,
  seed = 9396
)
```

Arguments

nProteins	integer. Number of proteins or features to simulate. Default is 10000.
nPerGroup	integer. Number of samples per group. Default is 20.
muMean	numeric. Mean of the protein-wise abundance distribution on the log2 scale. Default is 18.
muSd	numeric. Standard deviation of the protein-wise abundance distribution on the log2 scale. Default is 1.2.
muClip	numeric vector of length 2. Lower and upper bounds used to truncate simulated protein means. Default is c(15, 25).
rhoWithin	numeric. Within-group correlation target for the latent sample factor. Default is 0.85.
rhoBetween	numeric. Between-group correlation target for the latent sample factor. Default is 0.55.
loadingSd	numeric. Standard deviation of protein-specific loadings for the correlated latent factor. Default is 0.25.
sigmaHi	numeric. Residual standard deviation at high abundance. Default is 0.05.
sigmaLo	numeric. Residual standard deviation at low abundance. Default is 0.4.
gammaSigma	numeric. Controls how strongly residual variance depends on abundance. Default is 2.5.
propDE	numeric. Proportion of differentially expressed proteins. Default is 0.35.
logFCSd	numeric. Standard deviation of differential expression log-fold changes. Default is 1.
logFCMean	numeric. Mean of differential expression log-fold changes. Default is 0.
heteroLogFC	logical. If TRUE, the variance of log-fold changes depends on abundance. Default is TRUE.

`fcHi` `numeric`. Lower multiplier for abundance-dependent logFC heterogeneity. Default is `0.55`.

`fcLo` `numeric`. Upper multiplier for abundance-dependent logFC heterogeneity. Default is `2.5`.

`gammaFC` `numeric`. Controls how strongly logFC variability depends on abundance. Default is `7`.

`sampleShiftSd` `numeric`. Standard deviation of global sample shifts. Default is `0`.

`sampleShiftCap` `numeric`. Maximum absolute value allowed for global sample shifts. Default is `0.2`.

`sampleSdStrength` `numeric`. Strength of sample-specific mean-SD dependence. A value of `0` implies independence. Default is `0`.

`sampleSdRho` `numeric`. Correlation between sample mean rank and sample-specific SD effect. Must lie between `0` and `1`. Default is `0.8`.

`sampleSdCap` `numeric`. Maximum absolute value allowed for the log-multiplier controlling sample-specific SD effects. Default is `0.35`.

`addMissing` `logical`. Should MNAR missing values be added? Default is `TRUE`.

`targetMissing` `numeric`. Target overall proportion of missing Default is `0.001`.

`kMnar` `numeric`. Strength of abundance dependence in the MNAR missing values. value mechanism. Default is `1.2`.

`missingBySampleSd` `numeric`. Standard deviation of sample-specific missingness shifts. Default is `0.05`.

`seed` `integer`. Random seed used for reproducibility. Default is `9396`.

Details

The simulated data include several structured components:

1. Protein-wise mean abundance on the log2 scale.
2. A latent correlation structure inducing stronger within-group than between-group sample correlation.
3. Abundance-dependent heteroscedastic residual noise to create an MA-like wedge pattern.
4. Symmetric differential expression between the two groups.
5. Optional global sample shifts affecting overall intensity.
6. Optional sample-level mean-SD dependence.
7. Optional MNAR missingness driven by abundance and sample-specific effects.

The output is intended for testing normalization methods and associated scoring procedures under controlled simulation settings.

Value

A list with the following elements:

logData Numeric matrix of simulated log2-scale data.

rawData Numeric matrix of simulated raw-scale data, obtained as `2^logData`.

metadata `data.frame` containing sample names and group labels.

Examples

```
simData <- simulateData(
    nProteins = 1000,
    nPerGroup = 5,
    propDE = 0.2,
    addMissing = TRUE,
    seed = 123
)

dim(simData$logData)
head(simData$metadata)
```

`validateNormScoreInput` *Validate and prepare inputs for normScore and diagnostic graphs*

Description

Checks the input objects used by `normScore()`, including group annotation, raw data, normalized datasets, sample names, protein names, and bootstrap settings. The function also aligns the order of rows and columns across normalized datasets, adds the log-transformed dataset when required, and identifies whether the analysis contains a single group.

Usage

```
validateNormScoreInput(
  normalizedDataList,
  groupData,
  rawData,
  refGroup,
  altGroup,
  returnDetails = NULL,
  nBoot = NULL,
  fromMainFunction = FALSE
)
```

Arguments

`normalizedDataList`	A named list of normalized data matrices or data frames. Each element should contain proteins in rows and samples in columns.
`groupData`	A data frame or object coercible to a data frame containing sample-group annotation. The first column is assumed to contain sample names and the second column group labels.
`rawData`	A matrix or data frame containing raw intensity values, with proteins in rows and samples in columns.
`refGroup`	Character string indicating the reference group used for group-comparison metrics. If `NULL` or invalid, it is automatically set.
`altGroup`	Character string indicating the alternative group used for group-comparison metrics. If `NULL` or invalid, it is automatically set.

`returnDetails` Logical value indicating whether detailed results should be returned by `normScore()`.

`nBoot` Numeric value indicating the number of bootstrap resamples.

`fromMainFunction` Logical value indicating wheter inputs will be used to `normScore()` main function (nBoot and returnDetails are required), or only for displaying diagnostic plots. Default: `FALSE`.

Details

It can also be used to validate data for diagnostic plotting, using either the main global funcion `plotNormScoreDiagnostic()` or any individual plotting function. In this case, some input parameters are not requiered for the validation function (nBoot and returnDetails).

Value

A list containing validated and prepared inputs: `normalizedDataList`, `groupData`, `rawData`, `refGroup`, `altGroup`, `returnDetails`, `nBoot`, and `singleGroup`.

Examples

```
# Simulate proteomic data
simData <- simulateData(nProteins = 1000)

# Artificial normalization (just adding shifts to raw data)
normalizedDataList <- list(
    Norm1 = simData$logData + 0.1,
    Norm2 = simData$logData + 1,
    Norm3 = simData$logData - 1,
    Norm4 = simData$logData * 1.1,
    Norm5 = simData$logData * 0.9,
    Norm6 = simData$logData * runif(ncol(simData$logData), 0.8, 1.2)
)

# Validating inputs for normScore() function
inputs <- validateNormScoreInput(
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    returnDetails = TRUE,
    nBoot = 100,
    fromMainFunction = TRUE
)

# Validating inputs for diagnostic graphs
inputs <- validateNormScoreInput(
    normalizedDataList = normalizedDataList,
    groupData = simData$metadata,
    rawData = simData$rawData,
    refGroup = NULL,
    altGroup = NULL,
    fromMainFunction = FALSE
)
```

Index