# Package ‘OmicsQC’

January 20, 2025

**Type** Package

**Title** Nominating Quality Control Outliers in Genomic Profiling Studies

**Version** 1.1.0

**Description** A method that analyzes quality control metrics from multi-sample genomic sequencing studies and nominates poor quality samples for exclusion. Per sample quality control data are transformed into z-scores and aggregated. The distribution of aggregated z-scores are modelled using parametric distributions. The parameters of the optimal model, selected either by goodness-of-fit statistics or user-designation, are used for outlier nomination. Two implementations of the Cosine Similarity Outlier Detection algorithm are provided with flexible parameters for dataset customization.

**Depends** R (>= 2.10)

**Imports** stats, utils, fitdistrplus, lsa, BoutrosLab.plotting.general

**Suggests** knitr, rmarkdown, kableExtra, dplyr, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**VignetteBuilder** knitr

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Author** Anders Hugo Frelin [aut],
Helen Zhu [aut],
Paul C. Boutros [aut, cre] (<https://orcid.org/0000-0003-0553-7520>)

**Maintainer** Paul C. Boutros <PBoutros@mednet.ucla.edu>

**Repository** CRAN

**Date/Publication** 2024-03-01 15:02:43 UTC

## Contents

---

`accumulate.zscores` *Sum across sign corrected z-scores for total sample quality score*

---

### Description

This function takes a dataframe of all the sign corrected scores, thus all negative, and aggregates to get a total sample quality score.

### Usage

```
accumulate.zscores(zscores.corrected, filename = NULL)
```

### Arguments

| | |
|---|---|
| `zscores.corrected` | A dataframe whose rows are samples and each column a QC metric |
| `filename` | A filename where to save data. If NULL data will not be saved to file |

### Value

A dataframe of aggregated z-scores for each sample

**Sample** Sample IDs defined by the rownames of `zscores.corrected`

**Sum** Sum of z-scores

## correct.zscore.signs *Corrects the z-scores signs according to the metrics*

### Description

For some metrics a high z-score is good, while for others a low one is good. This function corrects for that so that a negative z-score is a poor score for every metric. It then sets all positive scores to zero.

### Usage

```
correct.zscore.signs(
  zscores,
  signs.data,
  metric.col.name = "Metric",
  signs.col.name = "Sign",
  filename = NULL
)
```

### Arguments

| | |
|---|---|
| `zscores` | A dataframe whose rows are samples and each column a QC metric, entries are z-scores |
| `signs.data` | A dataframe of two columns, the metric names and the sign of the metric |
| `metric.col.name` | The name of the column in signs.data that stores the metric name |
| `signs.col.name` | The name of the column in signs.data that stores sign as 'neg' or 'pos' |
| `filename` | A filename where to save data. If NULL data will not be saved to file |

### Value

A dataframe whose rows are the QC metrics, and columns are samples with the z-scores if they are negative

## cosine.similarity.cutoff *Calculate an outlier cutoff using cosine similarity*

## Description

This function takes quality.scores, trims it and fits it to the distribution given. It then simulates as many datasets as stated by no.simulations, and computes the cosine similarity of each dataset against theoretical distribution. It uses what would correspond to a significant value to then calculate what observed value this would correspond to. The function supports the following distributions:

- 'weibull'
- 'norm'
- 'gamma'
- 'exp'
- 'lnorm'
- 'cauchy'
- 'logis'

## Usage

```
cosine.similarity.cutoff(
  quality.scores,
  no.simulations,
 distribution = c("lnorm", "weibull", "norm", "gamma", "exp", "cauchy", "logis"),
  trim.factor = 0.05,
  alpha.significant = 0.05
)
```

## Arguments

| | |
|---|---|
| `quality.scores` | A dataframe with columns 'Sum' (of scores) and 'Sample', i.e. the output of accumulate.zscores |
| `no.simulations` | The number of datasets to simulate |
| `distribution` | A distribution to test, will default to 'lnorm' |
| `trim.factor` | What fraction of values of each to trim to get parameters without using extremes |
| `alpha.significant` | Alpha value for significance |

## Value

Results in the form of a named list

**cutoff** Computed cutoff for aggregated z-scores used as a threshold for nominating outliers

**no.outliers** Number of nominated outliers

**outlier.labels** Outlier IDs, corresponding to `Sample` column of `quality.scores`

## cosine.similarity.iterative

*Tests the accumulated quality scores for outliers using cosine similarity*

### Description

This function takes quality.scores, trims it and fits it to the distribution given. It then iteratively tests the largest datapoint compared a null distribution of size no.simulations. If the largest datapoint has a significant p-value it tests the 2nd largest one and so on. The function supports the following distributions:

- 'weibull'
- 'norm'
- 'gamma'
- 'exp'
- 'lnorm'
- 'cauchy'
- 'logis'

### Usage

```
cosine.similarity.iterative(
  quality.scores,
  no.simulations,
 distribution = c("lnorm", "weibull", "norm", "gamma", "exp", "cauchy", "logis"),
  trim.factor = 0.05,
  alpha.significant = 0.05
)
```

### Arguments

| | |
|---|---|
| `quality.scores` | A dataframe with columns 'Sum' (of scores) and 'Sample', i.e. the output of accumulate.zscores |
| `no.simulations` | The number of datasets to simulate |
| `distribution` | A distribution to test, will default to 'lnorm' |
| `trim.factor` | What fraction of values of each to trim to get parameters without using extremes |
| `alpha.significant` | Alpha value for significance |

### Value

Results in the form of a named list

**no.outliers** Number of nominated outliers

**outlier.labels** Outlier IDs, corresponding to `Sample` column of `quality.scores`

## example.qc.dataframe *QC metrics across 100 samples*

### Description

QC metrics across 100 samples

### Usage

```
data(example.qc.dataframe)
```

### Format

A data frame containing QC data; columns represent QC metrics and rows represent samples

### Examples

```
data(example.qc.dataframe)
zscores.from.metrics(
  qc.data = example.qc.dataframe
  );
```

## fit.and.evaluate *Fits the QC data to distributions and returns the KS test result and BIC score*

### Description

This function takes the accumulated QC scores, a vector of distributions and a trimming factor. It then returns the results for each distribution in a dataframe. This function supports the following distributions:

- 'weibull'
- 'norm'
- 'gamma'
- 'exp'
- 'lnorm'
- 'cauchy'
- 'logis'

### Usage

```
fit.and.evaluate(
  quality.scores,
  distributions = c("weibull", "norm", "gamma", "exp", "lnorm", "cauchy", "logis"),
  trim.factor = 0.05
)
```

### Arguments

| | |
|---|---|
| `quality.scores` | The accumulated QC scores, the output of accumulate.zscores |
| `distributions` | A vector of distributions to fit and test |
| `trim.factor` | The fraction of extremes on each end to trim before fitting |

### Value

A dataframe of the results with the following columns

**distribution** Name of the fitted distribution

**KS.rejected** Whether the Kolmogorov-Smirnov test rejects the fit; see `fitdistrplus::gofstat` - kstest

**BIC.value** Bayesian Information Criterion

---

## `get.qc.barplot` *Generates the standard barplot of scores for each sample*

---

### Description

This function takes the quality score data generated by accumulate.zscores() and returns the barplot if no filename is specified. If filename is is specified it saves the plot as file and returns NULL. It will also draw a cut-off for which samples to exclude. get.qc.barplot offers a standard template for generating a QC barplot, but can also take any parameter that BoutrosLab.plotting.general::create.barplot takes for more customizability.

### Usage

```
get.qc.barplot(
  quality.scores,
  filename = NULL,
  abline.h = -20,
  yaxis.cex = 0.8,
  xaxis.cex = 0,
  yaxis.tck = 1,
  xaxis.tck = 0,
  xlab.label = "",
  ylab.label = "Sum of Z (Z < 0)",
  ylab.cex = 1,
  abline.col = "darkgrey",
  axes.lwd = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| `quality.scores` | A dataframe with columns 'Sum' (of scores) and 'Sample', i.e. the output of accumulate.zscores |
| `filename` | Filename for tiff output, or if NULL returns the trellis object itself |
| `abline.h` | Adds a horizontal line to the plot; useful for depicting the threshold for what is deemed a poor sample quality score |
| `yaxis.cex` | Size of y-axis tick labels, defaults to 0.8 |
| `xaxis.cex` | Size of x-axis tick labels, defaults to 0 |
| `yaxis.tck` | Specifies the length of the tick marks for y-axis, defaults to 1 |
| `xaxis.tck` | Specifies the length of the tick marks for x-axis, defaults to 0 |
| `xlab.label` | The label for the x-axis, defaults to '' |
| `ylab.label` | label for the y-axis, defaults to 'Sum of Z (Z < 0)' |
| `ylab.cex` | Size of y-axis label, defaults to 1 |
| `abline.col` | Colour of the horizontal line on the plot, defaults to 'darkgrey' |
| `axes.lwd` | Specify line width of the axes; set to 0 to turn off axes |
| `...` | The function can also take any parameter that BoutrosLab.plotting.general::create.barplot takes |

## Value

The barplot or NULL depending if filename is specified

---

`get.qc.heatmap` *Generates the standard heatmap of scores for each sample.*

---

## Description

This function takes the the scores for each sample and each metric, after being sign-corrected, and returns the standard heatmap, if filename is NULL. If filename is not NULL it saves the heatmap to file and returns NULL. The function also takes quality.scores to make sure the samples are ordered correctly, as well as the y labels for the quality metrics. get.qc.heatmap offers a standard template for generating a QC heatmap, but can also take any parameter that BoutrosLab.plotting.general::create.barplot takes for customisability.

## Usage

```
get.qc.heatmap(
  zscores,
  quality.scores,
  yaxis.lab = colnames(zscores),
  xaxis.lab = quality.scores[, "Sample"],
  filename = NULL,
  yaxis.cex = 0.8,
  xaxis.cex = 0,
  xlab.cex = 1,
  xlab.label = "Samples",
  clustering.method = "none",
  colour.scheme = c("red", "white"),
  colour.centering.value = 0,
  colourkey.labels.at = c(-10:0),
  colourkey.cex = 1,
  at = seq(0, -10, -2),
  same.as.matrix = TRUE,
  row.lines = seq(1, ncol(zscores), 1) + 0.5,
  grid.row = TRUE,
  row.colour = "black",
  row.lwd = 1,
  axes.lwd = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| `zscores` | A dataframe of (sign-corrected) z-scores for each sample and test metric, i.e. the output of correct.zscore.signs |
| `quality.scores` | A dataframe with columns 'Sum' (of scores) and 'Sample', i.e. the output of accumulate.zscores |
| `yaxis.lab` | A vector of metric labels for the y-axis; defaults to column names of z-scores |
| `xaxis.lab` | A vector of sample labels for the x-axis; defaults to ordered Sample column elements in quality.scores |
| `filename` | Filename for tiff output, or if NULL returns the trellis object itself |
| `yaxis.cex` | Size of y-axis tick labels, defaults to 0.8 |
| `xaxis.cex` | Size of x-axis tick labels, defaults to 0 |
| `xlab.cex` | Size of x-axis label, defaults to 1 |
| `xlab.label` | The label for the x-axis, defaults so 'Sample' |
| `clustering.method` | Method used to cluster the records – "none" gives unclustered data. Accepts all agglomerative clustering methods available in hclust, plus "diana" (which is divisive). |
| `colour.scheme` | Heatmap colouring. Accepts old-style themes, or a vector of either two or three colours that are gradiated to create the final palette. |

| | |
|---|---|
| `colour.centering.value` | What should be the center of the colour-map |
| `colourkey.labels.at` | A vector specifying the tick-positions on the colourkey |
| `colourkey.cex` | Size of colourkey label text |
| `at` | A vector specifying the breakpoints along the range of x; each interval specified by these breakpoints are assigned to a colour from the palette. Defaults to seq(0, -10, -2), to give a clear discrete display of colours. If x has values outside of the range specified by "at" those values are shown with the colours corresponding to the extreme ends of the colour spectrum and a warning is given. |
| `same.as.matrix` | Prevents the flipping of the matrix that the function normally does |
| `row.lines` | Vector specifying location of lines, default is seq(1, ncol(x), 1) + 0.5. Note: Add 0.5 to customized vector |
| `grid.row` | Allow turning off of the interior grid-lines. Default is TRUE. |
| `row.colour` | Interior grid-line colour, defaults to "black". Can be a vector |
| `row.lwd` | Interior grid-line width, defaults to 1. Setting to zero is equivalent to grid.row = FALSE and grid.col = FALSE. Can be a vector. |
| `axes.lwd` | Width of heatmap border. Note it also changes the colourkey border and ticks |
| `...` | The function can also take any parameter that BoutrosLab.plotting.general::create.heatmap takes |

### Value

The heatmap or NULL depending if filename is specified

## `get.qc.multipanelplot` *Generates the multipanel plot of heatmap and barplot*

### Description

This function takes the barplot and heatmap and returns the multipanel plot of the two.

### Usage

```
get.qc.multipanelplot(
  barplot,
  heatmap,
  filename = NULL,
  width = 10,
  height = 8,
  layout.height = 2,
  layout.width = 1,
  plot.objects.heights = c(1, 3),
  y.spacing = -1,
  ylab.axis.padding = -19,
  left.padding = 8,
  main = "QC Summary",
  main.cex = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| `barplot` | A barplot of the samples aggregated score |
| `heatmap` | A heatmap of the sign-corrected scores for each sample |
| `filename` | Filename to output to |
| `width` | Width of resulting file |
| `height` | Height of resulting file |
| `layout.height` | how many plots per column |
| `layout.width` | how many plots per row. |
| `plot.objects.heights` | Heights of each row of the plot. Must be vector of same size as layout.height |
| `y.spacing` | vertical spacing between each plot. Can be single value or vector of length layout.height - 1 |
| `ylab.axis.padding` | padding between axis and y label of plots. Can be single value or vector of length layout.width |
| `left.padding` | padding from the left side of the frame |
| `main` | main label text |
| `main.cex` | main label cex |
| `...` | The function can also take any parameter that BoutrosLab.plotting.general::create.multipanelplot takes |

## Value

The multipanelplot or NULL depending if filename is specified

---

`sign.correction` *Directionality of QC metrics*

---

## Description

Directionality of QC metrics

## Usage

```
data(sign.correction)
```

## Format

A data frame containing the following columns: Metric, Sign

**Metric** Quality control metrics, corresponding to the metrics in `example.qc.dataframe`

**Sign** Directionality of each metric; positive (pos) means a higher metric is better, negative (neg) means a lower metric is better

## Examples

```
data(sign.correction)
data(example.qc.dataframe)
correct.zscore.signs(
  zscores = example.qc.dataframe,
  signs.data = sign.correction,
  metric.col.name = 'Metric',
  signs.col.name = 'Sign',
  );
```

---

`ylabels` *Formatted QC metrics labels*

---

## Description

Formatted QC metrics labels

## Usage

```
data(ylabels)
```

## Format

A character vector of formatted QC metric labels

## Examples

```
data(ylabels)
data(example.qc.dataframe)
data(sign.correction)
zscores <- zscores.from.metrics(qc.data = example.qc.dataframe);
zscores.corrected <- correct.zscore.signs(
  zscores = zscores,
  signs.data = sign.correction,
  metric.col.name = 'Metric',
  signs.col.name = 'Sign'
);
quality.scores <- accumulate.zscores(zscores.corrected = zscores.corrected);
qc.heatmap <- get.qc.heatmap(
  zscores = zscores.corrected,
  quality.scores = quality.scores,
  yaxis.lab = ylabels
);
```

---

`zscores.from.metrics` *Calculate z-scores for each metric across each sample*

---

## Description

This function takes a dataframe of QC metrics, and calculates the the z-scores. If filename is specified, the results will be saved to file.

## Usage

```
zscores.from.metrics(qc.data, filename = NULL)
```

## Arguments

| | |
|---|---|
| `qc.data` | A dataframe whose rows are samples and each column a QC metric |
| `filename` | A filename where to save data. If NULL data will not be saved to file |

## Value

A dataframe of z-scores; rows correspond to samples and columns correspond to metrics from `qc.data`

# Index