Package ‘qcmetrics’

April 15, 2017

Type Package

Title A Framework for Quality Control

Version 1.12.0

Author Laurent Gatto

Maintainer Laurent Gatto <`lg390@cam.ac.uk`>

Description The package provides a framework for generic quality control of data. It permits to create, manage and visualise individual or sets of quality control metrics and generate quality control reports in various formats.

Depends R (>= 2.10)

Imports Biobase, methods, knitr, tools, Nozzle.R1, xtable, pander, S4Vectors

Suggests affy, MSnbase, ggplot2, lattice, yaqcaffy, MAQCsubsetAFX, RforProteomics, AnnotationDbi, mzR, hgu133plus2cdf, BiocStyle

License GPL-2

URL `https://github.com/lgatto/qcmetrics`

biocViews Software, QualityControl, Proteomics, Microarray, MassSpectrometry, Visualization, ReportWriting

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

n15qc 2
psm 3
Qc2Tex 3
QcMetadata-class 4
QcMetric-class 5
QcMetrics-class 7
qcReport-methods 9
rnadeg 10

Index **12**

`n15qc` *N15 labelling QC report*

Description

A simple wrapper for the QC of 15N labelling. The respective QC items are the distribution of PSM incorporation rates, distribution of log2 fold-changes and number of identified features. See the vignette for details.

Usage

```
n15qc(object,
  fcol = c("Protein_Accession", "Peptide_Sequence", "Number_Of_Unique_Peptides", "Variable_Modif
   inctr = 97.5, lfctr = c(-0.5, 0.5), type, reportname)
```

Arguments

`object`	An `MSnSet` to be quality controlled.
`fcol`	The name of the feature variables for the protein identifiers (accession numbers for example), the peptide sequences, the number of unique peptides for each identified protein, the variable modifications identified on the peptides and the N15 incorporation rate. These must be provided in that order. Defaults are `Protein_Accession`, `Peptide_Sequence`, `Number_Of_Unique_Peptides`, `Variable_Modifications`, and `inc`.
`inctr`	The minimum level of median incorporation rate to set the QC item status to `TRUE`. Default is 97.5.
`lfctr`	The range of accepted median PSM log2 fold-change for the QC item status to be set to `TRUE`. Default is `c(-0.5, 0.5)`.
`type`	The type of report to be saved. If missing (default), no report is generated. See `qcReport` for details.
`reportname`	The name of the report, in case a `type` is defined. If missing (default), the report will be names `n15qcreport` followed by the generation data and time.

Value

Invisibly returns the resulting `QcMetrics` instance.

Author(s)

Laurent Gatto

`psm` *15N example data*

Description

An example data for 15N metabolic labelling, distributed as an `MSnSet` to illustrate 15N QC.

Usage

```
data(n15psm)
```

References

See `MSnSet` and the `MSnbase-demo` vignette in the `MSnbase` package.

Examples

```
library("MSnbase")
data(n15psm)
psm
```

`Qc2Tex` *'QcMetric' sectioning functions*

Description

These functions convert the `ith` `QcMetric` instance of the `QcMetrics` object into a section of the adequate format, i.e. TeX of R markdown.

Usage

```
Qc2Tex(object, i)
Qc2Tex2(object, i)
Qc2Tex3(object, i)
Qc2Rmd(object, i)
```

Arguments

`object`	An instance of class `QcMetrics` with at least one QC item.
`i`	A `numeric` of length 1 indicating the index of the item to be converted into text section.

Value

A `character` representing the QC item section.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

`qcReport` and the vignette.

`QcMetadata-class` *The* `"QcMetadata"` *class*

Description

The `QcMetadata` class is a simple interface to metadata. The objects can be displayed with `show` for a summary and `print` for the content.

Objects from the Class

Objects can be created by calls of the form `QcMetadata(...)`.

Slots

`metadata`**:** Object of class `"list"` that stores the metadata variables. The list must be named. `NA` and empty characters are not allowed.

Methods

[`signature(x = "QcMetadata")`: subsets `x` as a new `QcMetadata` instance.

[[`signature(x = "QcMetadata")`: extracts a single element of `x`.

metadata `signature(x = "QcMetadata")`: return the `object`'s metadata list. Also available as `mdata`.

metadata<- `signature(x = "QcMetadata", value = "list")`: sets the objects metadata. Also available as `mdata`.

length `signature(x = "QcMetadata")`: returns the number of metadata variables.

names `signature(x = "QcMetadata")`: returns the names of the metadata variables.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
QcMetadata(metadata =
           list(name = "John Doe",
                lab = "Big Lab in Big Uni"))
## less typing
qmd <- QcMetadata(list(name = "John Doe",
                       lab = "Big Lab in Big Uni"))
mdata(qmd)
show(qmd)
print(qmd)
```

`QcMetric-class` *The* `"QcMetric"` *class for QC items*

Description

Data structure for individual QC metrics items.

Objects from the Class

Objects can be created using the constructor `QcMetric(...)`, where slots are assigned individually. See example below.

Slots

`name`**:** Object of class `"character"` of length 1 naming the `object`.

`description`**:** Object of class `"character"` of arbitrary length describing the qc metric in more details.

`qcdata`**:** Object of class `"environment"` that stores the actual data.

`plot`**:** Object of class `"function"` to graphically represent the data and infer quality status.

`show`**:** Object of class `"function"` to print a short textual representation of the `object`. A reasonable default value is provided.

`status`**:** Object of class `"logical"` that indicates weather the data passes (`TRUE`) or fails (`FALSE`) the metric or has not yet been evaluated.

Methods

name `signature(object = "QcMetric")`: retrieves the name of the `object`.

name<- `signature(object = "QcMetric", value = "character")`: set the name of the `object`.

description `signature(object = "QcMetric")`: retrieves the description of the `object`.

description<- `signature(object = "QcMetric", value = "character")`: set the description of the `object`.

status `signature(object = "QcMetric")`: retrieves the status of the `object`.

status<- `signature(object = "QcMetric", value = "logical")`: sets the status of the `objects`.

qcdata `signature(object = "QcMetric", x = "missing")`: lists all the data `objects` that are associated with the `objects`.

qcdata `signature(object = "QcMetric", x = "character")`: retrieves the variable `x` for the `object`.

qcdata<- `signature(object = "QcMetric", var)`: creates or overwrites (with a message) the data variable `var` by assigning the RHS value. If `var` is missing and the RHS expression is an environment, then `qcdata` is reset with all the variables in `value`.

qcenv `signature(object = "QcMetric")`: return the `environment` that stores the QC data.

qcenv<- `signature(object = "QcMetric")`: Set all variable in the RHS `environment` as `qcdata` variables. Equivalent to `qcdata(object) <- x` where `x` is an environment.

show `signature(object = "QcMetric")`: shows a textual summary of `object`. The default show implementation is available as the `qcshow{object, qcdata}` function. The second argument is a logical (default is `TRUE`) that specifies whether `qcdata(object)` should be displayed.

show<- `signature(object = "QcMetric", value = "function")`: sets a custom show method for `object`.

plot `signature(x = "QcMetric", y = "missing")`: plots the `object` using the provide show method.

plot<- `signature(object = "QcMetric", value = "function")`: sets a custom `plot` method for `object`.

qcReport `signature(x = "QcMetric", ...)`: to generate quality reports. See `qcReport` for details.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

The `QcMetrics` class to bundle a set of `QcMetric` instances.

Examples

```
(qc <- QcMetric())
qcdata(qc)
try(qcdata(qc, "x"))

x <- rnorm(10)
qcdata(qc, "qc1") <- x
qcdata(qc, "qc2") <- 1:10
qcdata(qc)
all.equal(qcdata(qc, "qc1"), x)
all.equal(qcdata(qc, "qc2"), 1:10)
name(qc) <- "My test QcMetric"
description(qc) <- "This qc metric describes bla bla bla, indicating possible issues in the third step of prot
status(qc) <- FALSE
qc

## or
e <- new.env()
e$qc1 <- rnorm(100)
e$qc2 <- 1:100
qcdata(qc) <- e
length(qcdata(qc, "qc1"))
head(qcdata(qc, "qc2"))

show(qc)
show(qc) <- function(object) cat("Updated show method\n")
show(qc)
show(qc) <- qcshow
qc


plot(qc)
plot(qc) <-
    function(object, ...)
```

```
        plot(qcdata(object, "qc2"),
             qcdata(object, "qc1"),
             xlab = "qc1",
             ylab = "qc2",
             ...)
plot(qc)
plot(qc, col = "red", pch = 19)

## Not run:
    ## generate a report
    qcReport(qcm)

## End(Not run)
```

`QcMetrics-class` *The* `"QcMetrics"` *class for collections of QC items*

Description

Data structure for storing lists of `QcMetric` items.

Objects from the Class

Objects can be created using the constructor `QcMetrics(...)`, where slots are assigned individually. See example below.

In a standardised quality control pipeline, the `QcMetrics` and `QcMetric` object are not generated manually. Their creation is delegated to a wrapper function that reads a specific type of files, parses the data, produces the individual `QcMetric` instances and, eventually, the `QcMetric` object. See the package vignette for details and examples.

Slots

`metadata`**:** Object of class `QcMetadata` storing the metadata of the `object`. This list would typically contain the input file the data was read from, the date the object was generated, ... or fully fledged *minimum information* descriptions (see `MIAxE`), when available.

`qcdata`**:** Object of class `"list"` storing all the individual `QcMetric` instances.

Methods

[`signature(x = "QcMetrics")`: subsets `x` as a new `QcMetrics` instance.

[[`signature(x = "QcMetrics")`: extracts a single `QcMetric` instance.

length `signature(x = "QcMetrics")`: returns the number of `QcMetric` instances populate `x`.

metadata `signature(x = "QcMetrics")`: return the object's metadata list. Also available as `mdata`.

metadata<- `signature(x = "QcMetrics", value = "list")`: sets the objects metadata. Also available as `mdata`.

metadata<- `signature(x = "QcMetric", value = "QcMetadata")`: sets the objects metadata. Also available as `mdata`.

name `signature(object = "QcMetrics")`: returns a `character` vector of length `length(object)` with the names of the `QcMetric` instances.

qcdata `signature(object = "QcMetrics", x = "missing")`: returns a list of all `QcMetric` instances.

qcdata<- `signature(object = "QcMetrics", value = "list")`: sets the `qcdata` of object.

show `signature(object = "QcMetrics")`: prints a short textual description of `object`.

status `signature(object = "QcMetrics")`: returns a vector of quality statuses (`logicals`).

status<- `signature(object = "QcMetrics", value = "logical")`: sets the quality statuses. Length of `value` and `object` must be identical.

as `signature(object = "QcMetrics", "data.frame")`: coerces `object` as a `length(object)` by 2 data frame with the respective `QcMetric` instances names and statuses.

qcReport `signature(object = "QcMetrics")`: ...

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

The `QcMetric` class for individual QC items.

Examples

```
example(QcMetric)
show(qc)

qc2 <- QcMetric(name = "My other metric", status = TRUE)
qcdata(qc2, "x") <- rnorm(100)
qcdata(qc2, "k") <- rep(LETTERS[1:2], 50)

plot(qc2) <- function(object, ...) {
    require("lattice")
    d <- data.frame(x = qcdata(object, "x"),
                    k = qcdata(object, "k"))
    bwplot(x ~ k, data = d)
}

qcm <- QcMetrics(qcdata = list(qc, qc2))
qcm

qcm[1] ## a QcMetrics instance
qcm[[1]] ## a single QcMetric

metadata(qcm)
metadata(qcm) <- QcMetadata(list(name = "Prof. Who",
                                 lab = "Cabin University"))
## or, shorter but equivalent
metadata(qcm) <- list(name = "Prof. Who",
                      lab = "Cabin University")
metadata(qcm) ## or mdata(qcm)
## update metadata
metadata(qcm) <- list(lab = "Big lab", ## updated
                      uni = "Cabin University") ## added
mdata(qcm)
```

`qcReport-methods` *Automatic report generation*

Description

The `qcReport` method generates report in various formats taking a `QcMetrics` instance as input. Each individual quality control item produces a section with description of the item and a assessment figure.

Details

The reporting functions take a `QcMetrics` instance as input, generate the source of the report and compile it into the final format that are currently available are `reporting_pdf`, `reporting_tex`, `reporting_rmd`, `reporting_html` and `reporting_nozzle`. See `qcto` for details about the sectioning functions, that convert individual `QcMetric` objects into adequate report sections.

The package vignette documents the report generation in more details and describes possibilities for customisation.

Methods

`signature( object = "QcMetrics", reportname = "character", type = "character", author = "charact`
generates a report for the `QcMetrics` `object`. The report will be named according the `reportname` (default is `qcreprt`)and `type`, the latter defining the output format and the extension. Possible types are `pdf` (default), "tex", "Rmd", "html" (all generated using the package `knitr`) and "nozzle" (generated using the package `Nozzle.R1` package.) A custom `title` can be provided; default is "Quality control report generated with qcmetrics". If no `author` is provided, the default value (`Sys.getenv("USER")`) is used. The addition of a table of contents (default is `FALSE`), a metadata section, a summary section and the session information can be controlled with the `toc`, `metadata`, `summary` and `sessioninformation` arguments. The metadata section is added to the report when present and the other have `TRUE` as default.

It is possible to supply custom templates using the `template` arguments. Intermediate files are deleted, unless `clean` is set to `FALSE` and verbose output can be turned on by setting `quiet` to `FALSE`.

The `reporter` and `qcto` arguments are used to convert `QcMetric` and `QcMetrics` objects into report source. See Details and the package vignette for details.

Addition parameters can be passed to inner functions. For the `pdf` report, passed to `texi2pdf`; for `html`, passed to `markdown::markdownToHTML`.

The method invisibly returns the name of the report that was generated.

Examples

```
example(QcMetrics)
show(qcm)

destdir <- tempdir()
(report <- file.path(destdir, "testQCReport"))

## pdf report
qcReport(qcm, reportname = report)

## Not run:
```

```
    ## use pdflatex to generate the pdf file
    qcReport(qcm, reportname = report, texi2dvi = "pdflatex")

## End(Not run)

## default html report
html <- qcReport(qcm, reportname = report, type = "html")
html
if (interactive())
    browseURL(html)

## using a custom css templates
writeLines("
body {
    font-size: 14pt;
    width: 650px;
    background: #789855;
    margin-left: auto;
    margin-right: auto;
    margin-top: 20px;
    margin-bottom: 20;
    text-align:justify;
}", con = "style.css")

html2 <- qcReport(qcm, reportname = "customreport", template = "style.css", type = "html")
if (interactive())
    browseURL(html2)
```

`rnadeg` *A simple RNA degradation QC for Affymetrix arrays*

Description

A simple wrapper function that uses `affy`'s RNA degradation curves and `yaqcaffy`'s actin and GAPDH 3'/5' ratios to generate a simple RNA degradation `QcMetrics` results. Optionally generates a QC report. See the `qcmetrics` vignette for an explanation of the function and an example.

Usage

```
rnadeg(input, status, type,
  reportname = "rnadegradation")
```

Arguments

`input`	A `character` of CEL file names or an instance of class `affybatch`.
`status`	A `logical` of length 2 to set the respective `QcMetric`'s statuses.
`type`	The `type` of the report to be generated. Is missing, no report is generated.
`reportname`	The name of the report.

Value

Invisibly return the `QcMetrics` for the `input`.

Author(s)

Laurent Gatto

See Also

`QcMetric` and `QcMetrics` for details about the QC infrastructure and `qcReport` for information about the report generation.

Index

∗Topic **classes**
  QcMetadata-class, 4
  QcMetric-class, 5
  QcMetrics-class, 7
∗Topic **datasets**
  psm, 3
∗Topic **methods**
  qcReport-methods, 9
∗Topic **report**
  qcReport-methods, 9
[,QcMetadata-method *(QcMetadata-class)*, 4
[,QcMetrics-method *(QcMetrics-class)*, 7
[[,QcMetadata-method *(QcMetadata-class)*, 4
[[,QcMetrics-method *(QcMetrics-class)*, 7

as.data.frame.QcMetrics *(QcMetrics-class)*, 7

class:QcMetadata *(QcMetadata-class)*, 4
class:QcMetric *(QcMetric-class)*, 5
class:QcMetrics *(QcMetrics-class)*, 7

description,QcMetric-method *(QcMetric-class)*, 5
description<-,QcMetric,character-method *(QcMetric-class)*, 5

length,QcMetadata-method *(QcMetadata-class)*, 4
length,QcMetrics-method *(QcMetrics-class)*, 7

mdata *(QcMetrics-class)*, 7
mdata,QcMetadata-method *(QcMetadata-class)*, 4
mdata,QcMetrics-method *(QcMetrics-class)*, 7
mdata<- *(QcMetrics-class)*, 7
mdata<-,QcMetadata,list-method *(QcMetadata-class)*, 4
mdata<-,QcMetrics,list-method *(QcMetrics-class)*, 7
mdata<-,QcMetrics,QcMetadata-method *(QcMetrics-class)*, 7
metadata *(QcMetrics-class)*, 7
metadata,QcMetadata-method *(QcMetadata-class)*, 4
metadata,QcMetrics-method *(QcMetrics-class)*, 7
metadata<- *(QcMetrics-class)*, 7
metadata<-,QcMetadata,list-method *(QcMetadata-class)*, 4
metadata<-,QcMetrics,list-method *(QcMetrics-class)*, 7
metadata<-,QcMetrics,QcMetadata-method *(QcMetrics-class)*, 7
MIAxE, *7*

n15psm *(psm)*, 3
n15qc, 2
name *(QcMetrics-class)*, 7
name,QcMetric-method *(QcMetric-class)*, 5
name,QcMetrics-method *(QcMetrics-class)*, 7
name<- *(QcMetrics-class)*, 7
name<-,QcMetric,character-method *(QcMetric-class)*, 5
name<-,QcMetrics,character-method *(QcMetrics-class)*, 7
names,QcMetadata-method *(QcMetadata-class)*, 4
names<-,QcMetadata,character-method *(QcMetadata-class)*, 4

plot,QcMetric,missing-method *(QcMetric-class)*, 5
plot<- *(QcMetric-class)*, 5
plot<-,QcMetric,function-method *(QcMetric-class)*, 5
print,QcMetadata-method *(QcMetadata-class)*, 4
psm, 3

Qc2Rmd *(Qc2Tex)*, 3
Qc2Tex, 3
Qc2Tex2 *(Qc2Tex)*, 3

Qc2Tex3 *(*Qc2Tex*)*, 3
qcdata *(*QcMetric-class*)*, 5
qcdata,QcMetric,character-method
*(*QcMetric-class*)*, 5
qcdata,QcMetric,missing-method
*(*QcMetric-class*)*, 5
qcdata,QcMetrics,ANY-method
*(*QcMetrics-class*)*, 7
qcdata<- *(*QcMetric-class*)*, 5
qcdata<-,QcMetric,ANY-method
*(*QcMetric-class*)*, 5
qcdata<-,QcMetrics,list-method
*(*QcMetrics-class*)*, 7
qcenv *(*QcMetric-class*)*, 5
qcenv,QcMetric-method *(*QcMetric-class*)*,
5
qcenv<- *(*QcMetric-class*)*, 5
qcenv<-,QcMetric,environment-method
*(*QcMetric-class*)*, 5
QcMetadata, *7*
QcMetadata *(*QcMetadata-class*)*, 4
QcMetadata-class, 4
QcMetric, *7*, *8*, *11*
QcMetric *(*QcMetric-class*)*, 5
QcMetric-class, 5
QcMetrics, *6*, *9*, *11*
QcMetrics *(*QcMetrics-class*)*, 7
QcMetrics-class, 7
qcReport, *2*, *4*, *6*, *11*
qcReport *(*qcReport-methods*)*, 9
qcReport,QcMetrics-method
*(*qcReport-methods*)*, 9
qcReport-methods, 9
qcshow *(*QcMetric-class*)*, 5
qcto, *9*
qcto *(*Qc2Tex*)*, 3

reporting *(*qcReport-methods*)*, 9
reporting_html *(*qcReport-methods*)*, 9
reporting_nozzle *(*qcReport-methods*)*, 9
reporting_pdf *(*qcReport-methods*)*, 9
reporting_rmd *(*qcReport-methods*)*, 9
reporting_tex *(*qcReport-methods*)*, 9
rnadeg, 10

show,QcMetadata-method
*(*QcMetadata-class*)*, 4
show,QcMetric-method *(*QcMetric-class*)*, 5
show,QcMetrics-method
*(*QcMetrics-class*)*, 7
show<- *(*QcMetric-class*)*, 5
show<-,QcMetric,function-method
*(*QcMetric-class*)*, 5
status *(*QcMetric-class*)*, 5
status,QcMetric-method
*(*QcMetric-class*)*, 5
status,QcMetrics-method
*(*QcMetrics-class*)*, 7
status<- *(*QcMetric-class*)*, 5
status<-,QcMetric,logical-method
*(*QcMetric-class*)*, 5
status<-,QcMetrics,logical-method
*(*QcMetrics-class*)*, 7