

Package: rjd3qr (via r-universe)

October 13, 2026

Type Package

Title 'JDemetra+' Quality Report Generator

Version 0.4.2.9000

Description Tool for generating quality reports from cruncher outputs (and calculating series scores). The latest version of the cruncher can be downloaded here:
<<https://github.com/jdemetra/jwsacruncher/releases>>.

License EUPL

URL <https://github.com/InseeFr/rjd3qr>,
<https://insee.fr.github.io/rjd3qr/>

BugReports <https://github.com/InseeFr/rjd3qr/issues>

Depends R (>= 4.1)

Imports openxlsx, stats, tools, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev

Repository <https://insee.fr.r-universe.dev>

Date/Publication 2026-08-10 09:25:42 UTC

RemoteUrl <https://github.com/insee/rjd3qr>

RemoteRef HEAD

RemoteSha 109c7d0f6640be17517383ff5105769998487d03

Contents

compute_score	2
deprecated-rjd3qr	4
extract_JVS	5
extract_QR	7
extract_score	9
get_thresholds	10
JVS_matrix	11
print.QR_matrix	13
QR_matrix	15
QR_var_manipulation	17
rbind.QR_matrix	19
recode_indicator_num	20
set_thresholds	22
sort	23
weighted_score	24
write	25
Index	28

compute_score	<i>Score calculation</i>
---------------	--------------------------

Description

To calculate a score for each series from a quality report

Usage

```
## S3 method for class 'QR_matrix'
compute_score(
  x,
  score_pond = c(qs_residual_s_on_sa = 30L, f_residual_s_on_sa = 30L, qs_residual_sa_on_i
    = 20L, f_residual_sa_on_i = 20L, f_residual_td_on_sa = 30L, f_residual_td_on_i = 20L,
    oos_mean = 15L, oos_mse = 10L, residuals_independency = 15L,
    residuals_homoskedasticity = 5L, residuals_skewness = 5L, m7 = 5L, q_m2 = 5L),
  modalities = c("Good", "Uncertain", "", "Bad", "Severe"),
  normalize_score_value = NULL,
  na.rm = TRUE,
  n_contrib_score = NULL,
  conditional_indicator = NULL,
  thresholds = getOption("rjd3qr.thresholds"),
  ...
)

## S3 method for class 'mQR_matrix'
compute_score(x, ...)
```

Arguments

<code>x</code>	a QR_matrix or mQR_matrix object.
<code>score_pond</code>	the formula used to calculate the series score.
<code>modalities</code>	modalities ordered by importance in the score calculation (cf. details).
<code>normalize_score_value</code>	integer indicating the reference value for weights normalisation. If missing, weights will not be normalised.
<code>na.rm</code>	logical indicating whether missing values must be ignored when calculating the score.
<code>n_contrib_score</code>	integer indicating the number of variables to create in the quality report's values matrix to store the <code>n_contrib_score</code> greatest contributions to the score (cf. details). If not specified, no variable is created.
<code>conditional_indicator</code>	a list containing 3-elements sub-lists: "indicator", "conditions" and "condition_modalities". To reduce down to 1 the weight of chosen indicators depending on other variables' values (cf. details).
<code>thresholds</code>	list of numerical vectors. Thresholds applied to the various tests in order to classify into modalities Good, Uncertain, Bad and Severe. By default, the value of the "jdc_threshold" option is used. You can call the get_thresholds function to see what the thresholds object should look like.
<code>...</code>	other unused parameters.

Details

The function `compute_score` calculates a score from the modalities of a quality report: to each modality corresponds a weight that depends on the parameter `modalities`. The default parameter is `c("Good", "Uncertain", "Bad", "Severe")`, and the associated weights are respectively 0, 1, 2 and 3.

The score calculation is based on the `score_pond` parameter, which is a named integer vector containing the weights to apply to the (modalities matrix) variables. For example, with `score_pond = c(qs_residual_s_on_sa = 10, f_residual_td_on_sa = 5)`, the score will be based on the variables `qs_residual_s_on_sa` and `f_residual_td_on_sa`. The `qs_residual_s_on_sa` grades will be multiplied by 10 and the `f_residual_td_on_sa` grades, by 5. To ignore the missing values when calculating a score, use the parameter `na.rm = TRUE`.

The parameter `normalize_score_value` can be used to normalise the scores. For example, to have all scores between 0 and 20, specify `normalize_score_value = 20`.

When using parameter `n_contrib_score`, `n_contrib_score` new variables are added to the quality report's values matrix. These new variables store the names of the variables that contribute the most to the series score. For example, `n_contrib_score = 3` will add to the values matrix the three variables that contribute the most to the score. The new variables' names are `i_highest_score`, with `i` being the rank in terms of contribution to the score (`1_highest_score` contains the name of the greatest contributor, `2_highest_score` the second greatest, etc). Only the variables that have a non-zero contribution to the score are taken into account: if a series score is 0, all `i_highest_score` variables will be empty. And if a series score is positive only because of the `m7` statistic, `1_highest_score` will have a value of "m7" for this series and the other `i_highest_score` will be empty.

Some indicators are only relevant under certain conditions. For example, the homoscedasticity test is only valid when the residuals are independant, and the normality tests, only when the residuals are both independant and homoscedastic. In these cases, the parameter `conditional_indicator` can be of use since it reduces the weight of some variables down to 1 when some conditions are met. `conditional_indicator` is a list of 3-elements sub-lists:

- "indicator": the variable whose weight will be conditionally changed
- "conditions": the variables used to define the conditions
- "conditions_modalities": modalities that must be verified to induce the weight change For example, `conditional_indicator = list(list(indicator = "residuals_skewness", conditions = c("residuals_independency", "residuals_homoskedasticity"), conditions_modalities = c("Bad", "Severe")))`, reduces down to 1 the weight of the variable "residuals_skewness" when the modalities of the independancy test ("residuals_independency") or the homoscedasticity test ("residuals_homoskedasticity") are "Bad" or "Severe".

Value

a `QR_matrix` or `mQR_matrix` object.

See Also

[Traduction française](#)

Examples

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

# Extract the quality report from the demetra_m file
QR <- extract_QR(demetra_path)

# Calculer le score
QR <- compute_score(QR, n_contrib_score = 2)
print(QR)

# Extract the modalities matrix:
QR[["modalities"]][["score"]]
```

Description

Use `write()` instead of `export_xslx()`.

Usage

```
export_xlsx(x, ...)
```

Arguments

`x` An object of class `JVS_matrix`, `QR_matrix`, or `mQR_matrix` to export.
`...` Other unused arguments.

Value

"QR_matrix", "mQR_matrix" or "JVS_matrix" object invisibly.

Examples

```
# Path leading to a demetra_m matrix
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

# Extract the quality report from the demetra_m file
QR <- extract_QR(demetra_path)

# Compute the scores
QR1 <- compute_score(x = QR, n_contrib_score = 5)
QR2 <- compute_score(
  x = QR,
  score_pond = c(qs_residual_s_on_sa = 5, qs_residual_sa_on_i = 30,
    f_residual_td_on_sa = 10, f_residual_td_on_i = 40,
    oos_mean = 30, residuals_skewness = 15, m7 = 25)
)
mQR <- mQR_matrix(list(a = QR1, b = QR2))

# Export the Multiple Quality Report to an Excel file
# `export_xlsx` is deprecated.
# Use `write` instead:
write(x = QR, file = tempfile(fileext = ".xlsx"))
write(x = mQR, export_dir = tempdir())
```

extract_JVS

Extraction of a JVS report

Description

Extract a JVS report from the output files of JDemetra+. The output files can be generated with the GUI or with the cruncher and are CSV files containing the diagnostics matrix and the output series.

Usage

```
extract_JVS(
  dir = NULL,
  demetra_m = NULL,
  y = NULL,
  sa = NULL,
  s = NULL,
  t = NULL,
  ...
)
```

Arguments

<code>dir</code>	path to the directory containing the <code>demetra_m.csv</code> , <code>series_y.csv</code> , <code>series_sa.csv</code> , <code>series_s.csv</code> and <code>series_t.csv</code> files.
<code>demetra_m</code>	data.frame containing the diagnostics matrix. If missing or <code>NULL</code> , the file is searched for in <code>dir</code> .
<code>y</code>	data.frame containing the initial series. If missing or <code>NULL</code> , the file is searched for in <code>dir</code> .
<code>sa</code>	data.frame containing the seasonally adjusted series. If missing or <code>NULL</code> , the file is searched for in <code>dir</code> .
<code>s</code>	data.frame containing the seasonal component of the series. If missing or <code>NULL</code> , the file is searched for in <code>dir</code> .
<code>t</code>	data.frame containing the trend component series. If missing or <code>NULL</code> , the file is searched for in <code>dir</code> .
<code>...</code>	Other parameters to pass to <code>read_demetra_m()</code> such as <code>sep</code> (the separator used in the csv file. By default, <code>sep = ";"</code>) and <code>dec</code> (the decimal separator used in the csv file. By default, <code>dec = ","</code>)

Details

This function generates a JVS report from the output of JDemetra+. The output needed are the `demetra_m` diagnostics matrix (usually from the file *demetra_m.csv*) and the series `y`, `sa`, `s` and `t` (usually read from the series CSV files).

All this files can be generated by launching the cruncher (functions [cruncher_and_param](#)).

For more information about the generation of the output, see the vignette: `browseVignettes(package = "rjd3qr")`

If the series are provided, they have to be data.frame with the dates in the first column and the values of the series in the other columns.

This function returns a [JVS_matrix](#) object, which is a data.frame containing several indications on the seasonal adjustment of the series.

If all data frames (`demetra_m`, `y`, `sa`, `s` and `i`) are supplied, the `dir` argument is ignored. Otherwise, `dir` should point to the directory containing the corresponding CSV files.

Value

a `JVS_matrix` object.

See Also

[Traduction française](#)

Examples

```
# Path leading to the directory containing the needed files
dir_path <- system.file(
  "extdata",
  "WS/WS_world/Output/SAProcessing-1",
  package = "rjd3qr"
)

# Extract the JVS report from the directory
JVS <- extract_JVS(dir = dir_path)
```

extract_QR

Extraction of a quality report

Description

To extract a quality report from the csv file containing the diagnostics matrix.

Usage

```
extract_QR(file, x, thresholds = getOption("rjd3qr.thresholds"), ...)
```

Arguments

<code>file</code>	the csv file containing the diagnostics matrix. This argument supersedes the argument <code>matrix_output_file</code> .
<code>x</code>	data.frame containing the diagnostics matrix.
<code>thresholds</code>	list of numerical vectors. Thresholds applied to the various tests in order to classify into modalities Good, Uncertain, Bad and Severe. By default, the value of the <code>"jdc_threshold"</code> option is used. You can call the get_thresholds function to see what the thresholds object should look like.
<code>...</code>	Other parameter to pass to <code>read_demetra_m</code> such as <code>sep</code> (the separator used in the csv file. By default, <code>sep = ";"</code>) and <code>dec</code> (the decimal separator used in the csv file. By default, <code>dec = ","</code>)

Details

This function generates a quality report from a csv file containing diagnostics (usually from the file *demetra_m.csv*). The *demetra_m.csv* file can be generated by launching the cruncher (functions [cruncher](#) or [cruncher_and_param](#)) with the default export parameters, having used the default option `csv_layout = "vtable"` to format the output tables of the functions [cruncher_and_param](#) and [create_param_file](#) when creating the parameters file.

This function returns a [QR_matrix](#) object, which is a list of 3 objects:

- `modalities`, a `data.frame` containing several indicators and their categorical quality (Good, Uncertain, Bad, Severe).
- `values`, a `data.frame` containing the same indicators and the values that lead to their quality category (i.e.: p-values, statistics, etc.) as well as additional variables that don't have a modality/quality (series frequency and arima model).
- `score_formula` that will store the formula used to calculate the score (when relevant). Its initial value is `NULL`.

If `x` is supplied, the `file` and `matrix_output_file` arguments are ignored. The `file` argument also designates the path to the file containing the diagnostic matrix (which can be imported into R in parallel and used with the `x` argument).

Value

a [QR_matrix](#) object.

See Also

[Traduction française](#)

Other [QR_matrix](#) functions: [QR_matrix\(\)](#), [print.QR_matrix\(\)](#), [rbind.QR_matrix\(\)](#), [sort_weighted_score\(\)](#)

Examples

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

# Extract the quality report from the demetra_m file
QR <- extract_QR(file = demetra_path)

print(QR)

# Extract the modalities matrix:
QR[["modalities"]]
# Or:
QR[["modalities"]]
```

extract_score	<i>Score extraction</i>
---------------	-------------------------

Description

To extract score variables from [QR_matrix](#) or [mQR_matrix](#) objects.

Usage

```
extract_score(
  x,
  format_output = c("data.frame", "vector"),
  weighted_score = FALSE
)
```

Arguments

x a [QR_matrix](#) or [mQR_matrix](#).

format_output string of characters indicating the output format: either a `data.frame` or a vector.

weighted_score logical indicating whether to extract the weighted score (if previously calculated) or the unweighted one. By default, the unweighted score is extracted.

Details

For [QR_matrix](#) objects, the output is a vector or the object `NULL` if no score was previously calculated. For [mQR_matrix](#) objects, it is a list of scores (`NULL` elements or vectors).

Value

`extract_score()` returns a `data.frame` with two column: the series name and their score.

See Also

[Traduction française](#)

Examples

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

# Extract the quality report from the demetra_m file
QR <- extract_QR(demetra_path)
```

```
# Compute the score
QR1 <- compute_score(x = QR, n_contrib_score = 5)
QR2 <- compute_score(
  x = QR,
  score_pond = c(qs_residual_s_on_sa = 5, qs_residual_sa_on_i = 30,
                 f_residual_td_on_sa = 10, f_residual_td_on_i = 40,
                 oos_mean = 30, residuals_skewness = 15, m7 = 25)
)
mQR <- mQR_matrix(list(a = QR1, b = QR2))

# Extract score
extract_score(QR1)
extract_score(mQR)
```

get_thresholds	<i>Get all (default) thresholds</i>
----------------	-------------------------------------

Description

Get all (default) thresholds

Usage

```
get_thresholds(test_name, default = TRUE)
```

Arguments

test_name	String. The name of the test to get.
default	Boolean. (default is TRUE) If TRUE, the default threshold will be returned. If FALSE the current used thresholds.

Details

If test_name is missing, all threshold will be returned.

The thresholds are read from the option `rjd3qr.thresholds`.

Value

Returns invisibly the thresholds of the chosen item. If test_name is missing, it returns all the current thresholds.

See Also

[Traduction française](#)

Examples

```
# Get all default thresholds
get_thresholds(default = TRUE)

# Get all current thresholds
get_thresholds(default = FALSE)

# Get all current thresholds
get_thresholds(test_name = "oos_mean", default = FALSE)
```

JVS_matrix

*JVS matrix object***Description**

JVS_matrix() are creating a quality report based on the Eurostat JVS Plug-In.

Usage

```
JVS_matrix(x = list())

## S3 method for class 'data.frame'
JVS_matrix(x)

## S3 method for class 'JVS_matrix'
JVS_matrix(x)

## Default S3 method:
JVS_matrix(x)
```

Arguments

x a data.frame containing the output variables' values (test p-values, test statistics, etc.) and modalities (Yes/No).

Details

A [JVS_matrix](#) object is a data.frame with 30 items:

- Series
- Method
- Period
- Nobs
- Start
- End
- Adjustment

- Presence of Seasonality in the Raw Series
- Presence of TD effects
- Log-Transformation
- ARIMA Model
- LeapYear
- MovingHoliday
- NbTD
- Noutliers
- Outlier1
- Outlier2
- Outlier3
- Residual Seasonality in SA Series (F-test)
- Residual TD Effect
- Q-Stat (for X13)
- Final Henderson Filter
- Stage 2 Henderson Filter
- Seasonal Filter
- Quality
- Autocorrelation of order 1 of the SA series
- Ljung-Box Test (P-value)
- Autocorrelation negative and significant
- Irregular Standard-Deviation
- Max-Adj

Value

JVS_matrix() creates and returns a [JVS_matrix](#) object.

See Also

[Traduction française](#)

Examples

```
JVS_data <- data.frame(  
  Series = "Series 1",  
  Method = "X13",  
  Period = 12L,  
  Nobs = 300L,  
  Start = "2000-01-01",  
  End = "2024-12-01",  
  Adjustment = "SA",  
  `Presence of Seasonality in the Raw Series` = "Yes",
```

```

`Presence of TD effects` = "No",
`Log-Transformation` = "No",
`ARIMA Model` = "(0,1,1)(0,1,1)",
LeapYear = "Yes",
MovingHoliday = "No",
NbTD = 0L,
Noutliers = 1L,
Outlier1 = "A0 (2020-01)",
Outlier2 = "A0 (2018-11)",
Outlier3 = NA_character_,
`Residual Seasonality in SA Series (F-test)` = "No",
`Residual TD Effect` = "No",
`Q-Stat (for X13)` = "Good",
`Final Henderson Filter` = "H13",
`Stage 2 Henderson Filter` = "H13",
`Seasonal Filter` = "S3X5",
Quality = "Good",
`Autocorrelation of order 1 of the SA series` = 0.2,
`Ljung-Box Test (P-value)` = 0.8,
`Autocorrelation negative and significant` = "",
`Irregular Standard-Deviation` = 0.8,
`Max-Adj` = 2.5
)

# Create a JVS_matrix object
JVS <- JVS_matrix(JVS_data)

# Check the class of the object
class(JVS)

```

print.QR_matrix	<i>Printing QR_matrix and mQR_matrix objects</i>
-----------------	--

Description

To print information on a QR_matrix or mQR_matrix object.

Usage

```

## S3 method for class 'QR_matrix'
print(x, print_variables = TRUE, print_score_formula = TRUE, ...)

## S3 method for class 'mQR_matrix'
print(x, score_statistics = TRUE, ...)

```

Arguments

x a `mQR_matrix` or `QR_matrix` object.

`print_variables`
 logical indicating whether to print the indicators' name (including additional variables).

`print_score_formula`
 logical indicating whether to print the formula with which the score was calculated (when calculated).

`...`
 other unused arguments.

`score_statistics`
 logical indicating whether to print the statistics in the `mQR_matrix` scores (when calculated).

Value

the `print` method prints a `mQR_matrix` or `QR_matrix` object and returns it invisibly (via `invisible(x)`).

See Also

[Traduction française](#)

Other `QR_matrix` functions: `QR_matrix()`, `extract_QR()`, `rbind.QR_matrix()`, `sort`, `weighted_score()`

Examples

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

# Extract the quality report from the demetra_m file
QR <- extract_QR(file = demetra_path)

print(QR)

# Prepare 2 quality reports
QR1 <- compute_score(x = QR, n_contrib_score = 5)
QR2 <- compute_score(
  x = QR,
  score_pond = c(qs_residual_s_on_sa = 5, qs_residual_sa_on_i = 30,
    f_residual_td_on_sa = 10, f_residual_td_on_i = 40,
    oos_mean = 30, residuals_skewness = 15, m7 = 25)
)
mQR <- mQR_matrix(list(a = QR1, b = QR2))

print(mQR)
```

QR_matrix	<i>Quality report objects</i>
-----------	-------------------------------

Description

`mQR_matrix()` and `QR_matrix()` are creating one (or several) quality report. The function `is.QR_matrix()` and `is.mQR_matrix()` are functions to test whether an object is a quality report or a list of quality reports.

Usage

```
QR_matrix(modalities = NULL, values = NULL, score_formula = NULL)
```

```
mQR_matrix(x = list(), ...)
```

```
## S3 method for class 'QR_matrix'
```

```
mQR_matrix(x = QR_matrix(), ...)
```

```
## S3 method for class 'mQR_matrix'
```

```
mQR_matrix(x = mQR_matrix.default(), ...)
```

```
## Default S3 method:
```

```
mQR_matrix(x = list(), ...)
```

```
is.QR_matrix(x)
```

```
is.mQR_matrix(x)
```

Arguments

<code>modalities</code>	a <code>data.frame</code> containing the output variables' modalities (Good, Bad, etc.)
<code>values</code>	a <code>data.frame</code> containing the output variables' values (test p-values, test statistics, etc.) Therefore, the values data frame can contain more variables than the data frame modalities.
<code>score_formula</code>	the formula used to calculate the series score (if defined).
<code>x</code>	a <code>QR_matrix</code> object, a <code>mQR_matrix</code> object or a list of <code>QR_matrix</code> objects.
<code>...</code>	objects of the same type as <code>x</code> .

Details

A `QR_matrix` object is a list of three items:

- `modalities`, a `data.frame` containing a set of categorical variables (by default: Good, Uncertain, Bad, Severe).
- `values`, a `data.frame` containing the values corresponding to the modalities indicators (i.e. p-values, statistics, etc.), as well as variables for which a modality cannot be defined (e.g. the series frequency, the ARIMA model, etc).

- `score_formula` contains the formula used to calculate the series score (once the calculus is done).

Value

`QR_matrix()` creates and returns a `QR_matrix` object. `mQR_matrix()` creates and returns a `mQR_matrix` object (ie. a list of `QR_matrix` objects). `is.QR_matrix()` and `is.mQR_matrix()` return Boolean values (TRUE or FALSE).

See Also

[Traduction française](#)

Other `QR_matrix` functions: [extract_QR\(\)](#), [print.QR_matrix\(\)](#), [rbind.QR_matrix\(\)](#), [sort_weighted_score\(\)](#)

Examples

```
modalities <- data.frame(
  Quality = c("Good", "Uncertain", "Bad"),
  Seasonality = c("Good", "Good", "Bad")
)

values <- data.frame(
  Quality = c(0.95, 0.75, 0.02),
  Seasonality = c(0.80, 0.60, 0.01),
  Period = c(12L, 12L, 12L)
)

# Create two quality report objects
QR1 <- QR_matrix(
  modalities = modalities,
  values = values
)

QR2 <- QR_matrix(
  modalities = modalities,
  values = values
)

# Test whether an object is a quality report
is.QR_matrix(QR1)

# Create a list of quality reports
mQR <- mQR_matrix(QR1, QR2)

# Test whether an object is a list of quality reports
is.mQR_matrix(mQR)
```

Description

Functions to add indicator (`add_indicator()`), remove indicators (`remove_indicators()`) or re-train some indicators only (`retain_indicators()`) to and from `QR_matrix` or `mQR_matrix` objects. The series names (column "series") cannot be removed.

Usage

```
remove_indicators(x, ...)  
  
## Default S3 method:  
remove_indicators(x, ...)  
  
## S3 method for class 'QR_matrix'  
remove_indicators(x, ...)  
  
## S3 method for class 'mQR_matrix'  
remove_indicators(x, ...)  
  
retain_indicators(x, ...)  
  
## Default S3 method:  
retain_indicators(x, ...)  
  
## S3 method for class 'QR_matrix'  
retain_indicators(x, ...)  
  
## S3 method for class 'mQR_matrix'  
retain_indicators(x, ...)  
  
add_indicator(x, indicator, variable_name, ...)  
  
## Default S3 method:  
add_indicator(x, indicator, variable_name, ...)  
  
## S3 method for class 'QR_matrix'  
add_indicator(x, indicator, variable_name, ...)  
  
## S3 method for class 'mQR_matrix'  
add_indicator(x, indicator, variable_name, ...)
```

Arguments

`x` a `QR_matrix` or `mQR_matrix` object.

... other parameters of the function `merge` (for `add_indicator`) or names of the variable to remove or keep (for `retain_indicators` and `remove_indicators`)

`indicator` a vector or a `data.frame` (cf. details).

`variable_name` a string containing the name of the variables to add.

Details

The function `add_indicator()` adds the chosen indicator to the values matrix of a quality report. Therefore, because said indicator isn't added in the modalities matrix, it cannot be used to calculate a score (except for weighting). Before using the added variable for score calculation, it will have to be coded with the function `recode_indicator_num`.

The new indicator can be a vector or a `data.frame`. In both cases, its format must allow for pairing:

- a vector's elements must be named and these names must match those of the quality report (variable "series");
- a `data.frame` must contain a "series" column that matches with the quality report's series.

Value

- `remove_indicators()` returns the same object `x` reduced by the flags and variables used as arguments ... So if the input `x` is a `QR_matrix`, an object of class `QR_matrix` is returned. If the input `x` is a `mQR_matrix`, an object of class `mQR_matrix` is returned.
- `retain_indicators()` returns the same object, with only the chosen indicators.
- `add_indicators()` returns the same object, enhanced with the chosen indicator. So if the input `x` is a `QR_matrix`, an object of class `QR_matrix` is returned. If the input `x` is a `mQR_matrix`, an object of class `mQR_matrix` is returned.

See Also

[Traduction française](#)

Other var `QR_matrix` manipulation: [recode_indicator_num\(\)](#)

Examples

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

# Extract the quality report from the demetra_m file
QR <- extract_QR(demetra_path)

# Add a new indicator
my_alea <- rnorm(nrow(QR$modalities))
names(my_alea) <- QR$modalities$series
QR <- add_indicator(QR, indicator = my_alea, variable_name = "alea")
```

```
# Retains indicators
retain_indicators(QR, "alea", "m7") # retaining "alea" and "m7"
retain_indicators(QR, c("alea", "m7")) # Same

# Remove indicators
QR <- remove_indicators(QR, "alea") # Remove "alea"

retain_indicators(QR, "alea")
# is empty because we removed the alea indicator
```

rbind.QR_matrix	<i>Combining QR_matrix objects</i>
-----------------	------------------------------------

Description

Function to combine multiple [QR_matrix](#) objects: line by line, both for the modalities and the values table.

Usage

```
## S3 method for class 'QR_matrix'
rbind(..., check_formula = TRUE)
```

Arguments

... [QR_matrix](#) objects to combine.

check_formula logical indicating whether to check the score formulas' coherency. By default, check_formula = TRUE: an error is returned if the scores were calculated with different formulas. If check_formula = FALSE, no check is performed and the score_formula of the output is NULL.

Value

rbind.QR_matrix() returns a [QR_matrix](#) object.

See Also

[Traduction française](#)

Other QR_matrix functions: [QR_matrix\(\)](#), [extract_QR\(\)](#), [print.QR_matrix\(\)](#), [sort](#), [weighted_score\(\)](#)

Examples

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

# Extract the quality report from the demetra_m file
QR <- extract_QR(demetra_path)

# Compute differents scores
QR1 <- compute_score(
  x = QR,
  score_pond = c(m7 = 2, q = 3, qs_residual_s_on_sa = 5)
)
QR2 <- compute_score(QR, score_pond = c(m7 = 2, qs_residual_s_on_sa = 5))

# Merge two quality report
try(rbind(QR1, QR2)) # Une erreur est renvoyée
rbind(QR1, QR2, check_formula = FALSE)
```

recode_indicator_num	<i>Converting "values variables" into "modalities variables"</i>
----------------------	--

Description

To transform variables from the values matrix into categorical variables that can be added into the modalities matrix.

Usage

```
recode_indicator_num(
  x,
  variable_name,
  breaks = c(0, 0.01, 0.05, 0.1, 1),
  labels = c("Good", "Uncertain", "Bad", "Severe"),
  ...
)

## S3 method for class 'QR_matrix'
recode_indicator_num(
  x,
  variable_name,
  breaks = c(0, 0.01, 0.05, 0.1, 1),
  labels = c("Good", "Uncertain", "Bad", "Severe"),
  ...
)
```

```

)

## S3 method for class 'mQR_matrix'
recode_indicator_num(
  x,
  variable_name,
  breaks = c(0, 0.01, 0.05, 0.1, 1),
  labels = c("Good", "Uncertain", "Bad", "Severe"),
  ...
)

```

Arguments

x	a QR_matrix or mQR_matrix object.
variable_name	a vector of strings containing the names of the variables to convert.
breaks	see function cut .
labels	see function cut .
...	other parameters of the cut function.

Value

The function `recode_indicator_num()` returns the same object, enhanced with the chosen indicator. So if the input `x` is a `QR_matrix`, an object of class `QR_matrix` is returned. If the input `x` is a `mQR_matrix`, an object of class `mQR_matrix` is returned.

See Also

[Traduction française](#)

Other var `QR_matrix` manipulation: [QR_var_manipulation](#)

Examples

```

# Path to the demetra_m.csv file
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

# Extract the quality report from the demetra_m file
QR <- extract_QR(demetra_path)

# Recode residuals_skewness
QR2 <- recode_indicator_num(QR, variable_name = "residuals_skewness",
  breaks = c(0.0, 0.01, 0.05, 0.1, 1.0),
  labels = c("Good", "Uncertain", "Bad", "Severe")
)

QR$modalities$residuals_skewness

```

```
QR2$modalities$residuals_skewness
```

set_thresholds	<i>Set values for thresholds</i>
----------------	----------------------------------

Description

Set values for thresholds

Usage

```
set_thresholds(test_name, thresholds)
```

Arguments

test_name	String. The name of the test to update.
thresholds	Named vector of numerics. The upper values of each break of a threshold.

Details

If test_name is missing, the argument thresholds is not used and all thresholds will be updated to their default values.

If test_name is not missing, but if the argument thresholds is missing then only the thresholds of the test test_name will be updated to its default values.

Finally, if test_name and thresholds are not missing, then only the thresholds of the test test_name are updated with the value thresholds.

This function updates the option `rjd3qr.thresholds`.

Value

Returns invisibly all the current (updated) thresholds. It's a list of item with grade ("Good", "Uncertain", "Bad", "Severe", "Poor"...).

See Also

[Traduction française](#)

Examples

```
# Set "m7"
set_thresholds(
  test_name = "m7",
  thresholds = c(Good = 0.8, Bad = 1.4, Severe = Inf)
)

# Set "oos_mean" to default
```

```

set_thresholds(test_name = "oos_mean")

# Set all thresholds to default
set_thresholds()

```

sort

QR_matrix and mQR_matrix sorting

Description

To sort the quality reports on one or several variables

Usage

```

## S3 method for class 'QR_matrix'
sort(x, decreasing = FALSE, sort_variables = "score", ...)

## S3 method for class 'mQR_matrix'
sort(x, decreasing = FALSE, sort_variables = "score", ...)

```

Arguments

x	a QR_matrix or mQR_matrix object
decreasing	logical indicating whether the quality reports must be sorted in ascending or decreasing order. By default, the sorting is done in ascending order.
sort_variables	They must be present in the modalities table.
...	other parameters of the function order (unused for now)

Value

the input with sorted quality reports

See Also

[Traduction française](#)

Other [QR_matrix](#) functions: [QR_matrix\(\)](#), [extract_QR\(\)](#), [print.QR_matrix\(\)](#), [rbind.QR_matrix\(\)](#), [weighted_score\(\)](#)

Examples

```

# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

```

```
# Extract the quality report from the demetra_m file
QR <- extract_QR(demetra_path)

# Compute the score
QR <- compute_score(QR, n_contrib_score = 2)
print(QR[["modalities"]][["score"]])

# Sort the scores

# To sort by ascending scores
QR <- sort(QR, sort_variables = "score")
print(QR[["modalities"]][["score"]])
```

weighted_score	<i>Weighted score calculation</i>
----------------	-----------------------------------

Description

Function to weight a pre-calculated score

Usage

```
weighted_score(x, pond = 1L)

## Default S3 method:
weighted_score(x, pond = 1L)

## S3 method for class 'QR_matrix'
weighted_score(x, pond = 1L)

## S3 method for class 'mQR_matrix'
weighted_score(x, pond = 1L)
```

Arguments

x	a QR_matrix or mQR_matrix object
pond	the weights to use. Can be an integer, a vector of integers, the name of one of the quality report variables or a list of weights for the mQR_matrix objects.

Value

the input with an additionnal weighted score

See Also

[Traduction française](#)

Other QR_matrix functions: [QR_matrix\(\)](#), [extract_QR\(\)](#), [print.QR_matrix\(\)](#), [rbind.QR_matrix\(\)](#), [sort](#)

Examples

```

# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "rjd3qr"),
  "WS/WS_world/Output/SAProcessing-1",
  "demetra_m.csv"
)

# Extract the quality report from the demetra_m file
QR <- extract_QR(demetra_path)

# Compute the score
QR <- compute_score(QR, n_contrib_score = 2)

# Weighted score
QR <- weighted_score(QR, 2)
print(QR)

# Extract the weighted score
QR[["modalities"]][["score_pond"]]

```

write

Writing Quality Reports to Files

Description

Writing Quality Reports to Files

Usage

```

write(x, ...)

## S3 method for class 'QR_matrix'
write(x, file, auto_format = TRUE, overwrite = TRUE, ...)

## S3 method for class 'JVS_matrix'
write(
  x,
  file = file.path(tempdir(), "JobVacancySurveyQR.csv"),
  overwrite = TRUE,
  verbose = TRUE,
  ...
)

## S3 method for class 'mQR_matrix'
write(
  x,

```

```

export_dir,
layout_file = c("ByComponent", "ByQRMatrix", "AllTogether"),
auto_format = TRUE,
overwrite = TRUE,
...
)

```

Arguments

<code>x</code>	An object of class <code>JVS_matrix</code> , <code>QR_matrix</code> , or <code>mQR_matrix</code> to export.
<code>...</code>	Other unused arguments.
<code>file</code>	A character object containing the path to the file to be created.
<code>auto_format</code>	Boolean indicating whether to format the output (<code>auto_format = TRUE</code> by default).
<code>overwrite</code>	Boolean. Should an existing file be overwritten? By default, <code>overwrite = TRUE</code> .
<code>verbose</code>	Boolean indicating whether to print additional information. Default is <code>TRUE</code> .
<code>export_dir</code>	Path to the directory that will contain the exported files.
<code>layout_file</code>	Export parameter. By default, (<code>layout_file = "ByComponent"</code>) and an Excel file is exported for each component of the quality report matrix (modalities or values matrix), where each sheet corresponds to a quality report. To have one file per quality report, with each sheet corresponding to the exported component, use <code>layout_file = "ByQRMatrix"</code> . The modality <code>layout_file = "AllTogether"</code> corresponds to creating a file with 2 sheets per quality report (Values and Modalities).

Details

Objects of class `JVS_matrix` can be exported to CSV or Excel files (depending on the format argument). Objects of class `QR_matrix` and `mQR_matrix` are only written to Excel files.

Value

If `x` is of class `JVS_matrix` or `mQR_matrix`, the function invisibly returns (with `invisible()`) the object `x`. If `x` is of class `QR_matrix`, the function invisibly returns (with `invisible()`) a workbook object created by `openxlsx::loadWorkbook()` for further manipulation.

See Also

[Traduction française](#)

Examples

```

# Path to the directory containing the demetra_m file and series
dir_path <- system.file(
  "extdata", "WS", "WS_world", "Output", "SAProcessing-1",
  package = "rjd3qr"
)

```

```
# Path to the demetra_m.csv file
demetra_path <- file.path(dir_path, "demetra_m.csv")

# Extract the quality report from the demetra_m.csv file
QR <- extract_QR(demetra_path)

# Export the QR to an Excel file
write(x = QR, file = tempfile(fileext = ".xlsx"))

# Prepare 2 quality reports
QR1 <- compute_score(x = QR, n_contrib_score = 5)
QR2 <- compute_score(
  x = QR,
  score_pond = c(qs_residual_s_on_sa = 5, qs_residual_sa_on_i = 30,
    f_residual_td_on_sa = 10, f_residual_td_on_i = 40,
    oos_mean = 30, residuals_skewness = 15, m7 = 25)
)
mQR <- mQR_matrix(list(a = QR1, b = QR2))

# Export the mQR to an Excel file
write(x = mQR, export_dir = tempdir())

# Extract the JVS report from CSV files
JVS <- extract_JVS(dir = dir_path)

# Export the JVS report to an Excel file
write(JVS, format = "xlsx", export_dir = tempdir(), overwrite = TRUE)

# Export the JVS report to a CSV file
write(JVS, format = "csv", export_dir = tempdir(), overwrite = TRUE)
```

Index

- * **JVS_matrix functions**
 - extract_JVS, [5](#)
- * **QR_matrix functions**
 - extract_QR, [7](#)
 - print.QR_matrix, [13](#)
 - QR_matrix, [15](#)
 - rbind.QR_matrix, [19](#)
 - sort, [23](#)
 - weighted_score, [24](#)
- * **var QR_matrix manipulation**
 - QR_var_manipulation, [17](#)
 - recode_indicator_num, [20](#)
- add_indicator (QR_var_manipulation), [17](#)
- compute_score, [2](#)
- create_param_file, [8](#)
- cruncher, [8](#)
- cruncher_and_param, [6](#), [8](#)
- cut, [21](#)
- deprecated-rjd3qr, [4](#)
- export_xlsx (deprecated-rjd3qr), [4](#)
- extract_JVS, [5](#)
- extract_QR, [7](#)
- extract_QR(), [14](#), [16](#), [19](#), [23](#), [24](#)
- extract_score, [9](#)
- get_thresholds, [3](#), [7](#), [10](#)
- is.mQR_matrix (QR_matrix), [15](#)
- is.QR_matrix (QR_matrix), [15](#)
- JVS_matrix, [5–7](#), [11](#), [11](#), [12](#), [26](#)
- merge, [18](#)
- mQR_matrix, [3–5](#), [9](#), [13–17](#), [21](#), [23](#), [24](#), [26](#)
- mQR_matrix (QR_matrix), [15](#)
- order, [23](#)
- print.mQR_matrix (print.QR_matrix), [13](#)
- print.QR_matrix, [13](#)
- print.QR_matrix(), [8](#), [16](#), [19](#), [23](#), [24](#)
- QR_matrix, [3–5](#), [8](#), [9](#), [15](#), [15–17](#), [19](#), [21](#), [23](#), [24](#), [26](#)
- QR_matrix(), [8](#), [14](#), [19](#), [23](#), [24](#)
- QR_var_manipulation, [17](#), [21](#)
- rbind.QR_matrix, [19](#)
- rbind.QR_matrix(), [8](#), [14](#), [16](#), [23](#), [24](#)
- recode_indicator_num, [18](#), [20](#)
- recode_indicator_num(), [18](#)
- remove_indicators (QR_var_manipulation), [17](#)
- retain_indicators (QR_var_manipulation), [17](#)
- set_thresholds, [22](#)
- sort, [8](#), [14](#), [16](#), [19](#), [23](#), [24](#)
- Traduction française, [4](#), [7–10](#), [12](#), [14](#), [16](#), [18](#), [19](#), [21–24](#), [26](#)
- weighted_score, [24](#)
- weighted_score(), [8](#), [14](#), [16](#), [19](#), [23](#)
- write, [25](#)
- write(), [4](#)