

# Package ‘JDCruncher’

May 7, 2026

**Type** Package

**Title** Interface Between the 'JDemetra+' Cruncher and R, and Quality Report Generator

**Version** 0.3.6

**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/JDCruncher>,  
<https://insee.fr.github.io/JDCruncher/>

**BugReports** <https://github.com/InseeFr/JDCruncher/issues>

**Depends** R (>= 4.1)

**Imports** openxlsx, stats, tools, utils

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Tanguy Barthelemy [aut, cre, art],  
Alain Quartier-la-Tente [aut] (ORCID:  
<<https://orcid.org/0000-0001-7890-3857>>),  
Institut national de la statistique et des études économiques [cph]  
(<https://www.insee.fr/>),  
Anna Smyk [aut]

**Maintainer** Tanguy Barthelemy <tanguy.barthelemy@insee.fr>

**Repository** CRAN

**Date/Publication** 2025-07-24 14:20:14 UTC

## Contents

add_indicator . . . . .	2
compute_score . . . . .	3
export_xlsx . . . . .	5
export_xlsx.mQR_matrix . . . . .	6
export_xlsx.QR_matrix . . . . .	7
extract_QR . . . . .	8
extract_score . . . . .	9
get_thresholds . . . . .	11
print.QR_matrix . . . . .	11
QR_matrix . . . . .	12
QR_var_manipulation . . . . .	13
rbind.QR_matrix . . . . .	14
recode_indicator_num . . . . .	15
set_thresholds . . . . .	16
sort . . . . .	17
weighted_score . . . . .	18
<b>Index</b>	<b>20</b>

---

add_indicator	<i>Adding an indicator in QR_matrix objects</i>
---------------	---

---

### Description

Function to add indicators in [QR\\_matrix](#) objects.

### Usage

```
add_indicator(x, indicator, variable_name, ...)
```

### Arguments

x	a <a href="#">QR_matrix</a> or <a href="#">mQR_matrix</a> object
indicator	a vector or a <code>data.frame</code> (cf. details).
variable_name	a string containing the name of the variables to add.
...	other parameters of the function <a href="#">merge</a> .

### 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

This function 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](#), [recode\\_indicator\\_num\(\)](#)

---

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_sa_on_sa = 30L, f_residual_sa_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("jdc_thresholds"),
  ...
)

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

### Arguments

<code>x</code>	a <code>QR_matrix</code> or <code>mQR_matrix</code> 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 <code>get_thresholds</code> 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_sa_on_sa = 10, f_residual_td_on_sa = 5)`, the score will be based on the variables `qs_residual_sa_on_sa` and `f_residual_td_on_sa`. The `qs_residual_sa_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 = "JDCruncherR"),
  "WS/ws_ipi/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

Exporting `QR_matrix` or `mQR_matrix` objects in an Excel file

**Usage**

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

**Arguments**

x                    a `QR_matrix` or `mQR_matrix` object.  
 ...                  other parameters of the function `export_xlsx.QR_matrix`.

**Value**

If x is a `mQR_matrix`, the function returns invisibly (via `invisible(x)`) the same `mQR_matrix` object as x. Else if x is a `QR_matrix`, the function returns invisibly (via `invisible(x)`) a workbook object created by `XLConnect::loadWorkbook()` for further manipulation.

**See Also**

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

---

```
export_xlsx.mQR_matrix
```

*Exporting mQR\_matrix objects in Excel files*

---

**Description**

To export several quality reports in Excel files

**Usage**

```
## S3 method for class 'mQR_matrix'
export_xlsx(
  x,
  export_dir,
  layout_file = c("ByComponent", "ByQRMatrix", "AllTogether"),
  auto_format = TRUE,
  overwrite = TRUE,
  ...
)
```

**Arguments**

x                    a `mQR_matrix` object to export.  
 export\_dir          export directory.  
 layout\_file        export parameter. By default, (`layout_file = "ByComponent"`) and an Excel file is exported for each part of the quality report matrix (modalities and values matrices). To group both modalities and values reports/sheets into a single Excel file, use the option `layout_file = "ByQRMatrix"`.

auto_format	logical indicating whether to format the output (auto_format = TRUE by default).
overwrite	logical indicating whether to create an Excel file if it doesn't exist yet (create = TRUE by default)
...	other unused arguments

**Value**

Returns invisibly (via `invisible(x)`) the same `mQR_matrix` object as `x`.

**See Also**

[Traduction française](#)

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

---

`export_xlsx.QR_matrix` *Exporting QR\_matrix objects in an Excel file*

---

**Description**

To export a quality report in an Excel file.

**Usage**

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

**Arguments**

<code>x</code>	a <code>QR_matrix</code> object.
<code>file</code>	a character object with the path to the file to export que l'on veut créer
<code>auto_format</code>	logical indicating whether to format the output (auto_format = TRUE by default).
<code>overwrite</code>	logical indicating whether to create an Excel file if it doesn't exist yet (create = TRUE by default)
...	other unused arguments

**Value**

Returns invisibly (via `invisible(x)`) a workbook object created by `XLConnect::loadWorkbook()` for further manipulation.

**See Also**

[Traduction française](#)

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

---

extract_QR	<i>Extraction of a quality report</i>
------------	---------------------------------------

---

### Description

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

### Usage

```
extract_QR(
  file,
  x,
  matrix_output_file,
  sep = ";",
  dec = ",",
  thresholds = getOption("jdc_thresholds")
)
```

### Arguments

file	the csv file containing the diagnostics matrix. This argument supersedes the argument <code>matrix_output_file</code> .
x	data.frame containing the diagnostics matrix.
matrix_output_file	the csv file containing the diagnostics matrix.
sep	the separator used in the csv file (by default, <code>sep = ";"</code> )
dec	the decimal separator used in the csv file (by default, <code>dec = ","</code> )
thresholds	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 <a href="#">get_thresholds</a> function to see what the thresholds object should look like.

### 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: [export\\_xlsx\(\)](#), [export\\_xlsx.QR\\_matrix\(\)](#), [export\\_xlsx.mQR\\_matrix\(\)](#), [rbind.QR\\_matrix\(\)](#), [sort\(\)](#), [weighted\\_score\(\)](#)

## Examples

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "JDCruncherR"),
  "WS/ws_ipi/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

*Score extraction*

---

## 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 = "JDCruncheR"),
  "WS/ws_ipi/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_sa_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.

**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)
```

---

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 <code>mQR_matrix</code> or <code>QR_matrix</code> 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 <code>mQR_matrix</code> 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](#)

---

QR\_matrix

*Quality report objects*

---

**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(), ...)
```

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

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

**Arguments**

modalities	a <code>data.frame</code> containing the output variables' modalities (Good, Bad, etc.)
values	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.

score\_formula the formula used to calculate the series score (if defined).  
 x a [QR\\_matrix](#) object, a [mQR\\_matrix](#) object or a list of [QR\\_matrix](#) objects.  
 ... objects of the same type as x.

### 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](#)

---

QR\_var\_manipulation    *Editing the indicators list*

---

### Description

Functions to remove indicators (`remove_indicators()`) or retrain some indicators only (`retain_indicators()`) from [QR\\_matrix](#) or [mQR\\_matrix](#) objects. The series names (column "series") cannot be removed.

### Usage

```
remove_indicators(x, ...)
```

```
retain_indicators(x, ...)
```

### Arguments

x a [QR\\_matrix](#) or [mQR\\_matrix](#) object.  
 ... names of the variable to remove (or keep)

**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.

**See Also**

[Traduction française](#)

Other var `QR_matrix` manipulation: [add\\_indicator\(\)](#), [recode\\_indicator\\_num\(\)](#)

**Examples**

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "JDCruncher"),
  "WS/ws_ipi/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)

# Retain indicators
retain_indicators(QR, "score", "m7") # retaining "score" and "m7"
retain_indicators(QR, c("score", "m7")) # Same

# Remove indicators
QR <- remove_indicators(QR, "score") # removing "score"

extract_score(QR) # is NULL because we removed the score indicator
```

---

rbind.QR\_matrix

*Combining QR\_matrix objects*

---

**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: [export\\_xlsx\(\)](#), [export\\_xlsx.QR\\_matrix\(\)](#), [export\\_xlsx.mQR\\_matrix\(\)](#), [extract\\_QR\(\)](#), [sort\(\)](#), [weighted\\_score\(\)](#)

## Examples

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "JDCruncher"),
  "WS/ws_ipi/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(QR, score_pond = c(m7 = 2, q = 3, qs_residual_sa_on_sa = 5))
QR2 <- compute_score(QR, score_pond = c(m7 = 2, qs_residual_sa_on_sa = 5))

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

---

recode\_indicator\_num *Converting "values variables" into "modalities variables"*

---

## 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"),
  ...
)

```

**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](#), [add\\_indicator\(\)](#)

---

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`.

## 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

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

**Value**

the input with sorted quality reports

**See Also**

[Traduction française](#)

Other QR\_matrix functions: [export\\_xlsx\(\)](#), [export\\_xlsx.QR\\_matrix\(\)](#), [export\\_xlsx.mQR\\_matrix\(\)](#), [extract\\_QR\(\)](#), [rbind.QR\\_matrix\(\)](#), [weighted\\_score\(\)](#)

**Examples**

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "JDCruncheR"),
  "WS/ws_ipi/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

*Weighted score calculation*

---

**Description**

Function to weight a pre-calculated score

**Usage**

```
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: [export\\_xlsx\(\)](#), [export\\_xlsx.QR\\_matrix\(\)](#), [export\\_xlsx.mQR\\_matrix\(\)](#), [extract\\_QR\(\)](#), [rbind.QR\\_matrix\(\)](#), [sort\(\)](#)

**Examples**

```
# Path of matrix demetra_m
demetra_path <- file.path(
  system.file("extdata", package = "JDCruncherR"),
  "WS/ws_ipi/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"]]
```

# Index

## \* QR\_matrix functions

export\_xlsx, 5  
export\_xlsx.mQR\_matrix, 6  
export\_xlsx.QR\_matrix, 7  
extract\_QR, 8  
rbind.QR\_matrix, 14  
sort, 17  
weighted\_score, 18

## \* var QR\_matrix manipulation

add\_indicator, 2  
QR\_var\_manipulation, 13  
recode\_indicator\_num, 15

add\_indicator, 2, 14, 16

compute\_score, 3  
create\_param\_file, 8  
cruncher, 8  
cruncher\_and\_param, 8  
cut, 16

export\_xlsx, 5, 7, 9, 15, 18, 19  
export\_xlsx.mQR\_matrix, 6, 6, 7, 9, 15, 18,  
19  
export\_xlsx.QR\_matrix, 6, 7, 7, 9, 15, 18, 19  
extract\_QR, 6, 7, 8, 15, 18, 19  
extract\_score, 9

get\_thresholds, 4, 8, 11

is.mQR\_matrix (QR\_matrix), 12  
is.QR\_matrix (QR\_matrix), 12

merge, 2  
mQR\_matrix, 2, 4–7, 9, 10, 12, 13, 16–18  
mQR\_matrix (QR\_matrix), 12

order, 17

print.mQR\_matrix (print.QR\_matrix), 11  
print.QR\_matrix, 11

QR\_matrix, 2, 4–10, 12, 13–18  
QR\_var\_manipulation, 3, 13, 16

rbind.QR\_matrix, 6, 7, 9, 14, 18, 19  
recode\_indicator\_num, 2, 3, 14, 15  
remove\_indicators  
(QR\_var\_manipulation), 13  
retain\_indicators  
(QR\_var\_manipulation), 13

set\_thresholds, 16  
sort, 6, 7, 9, 15, 17, 19

Traduction française, 3, 5, 7, 9, 10, 12–16,  
18, 19

weighted\_score, 6, 7, 9, 15, 18, 18