

Package ‘TSQCA’

February 18, 2026

Type Package

Title Threshold Sweep Extensions for Qualitative Comparative Analysis

Version 1.3.1

Description Provides threshold sweep methods for Qualitative Comparative Analysis (QCA). Implements Condition Threshold Sweep-Single (CTS-S), Condition Threshold Sweep-Multiple (CTS-M), Outcome Threshold Sweep (OTS), and Dual Threshold Sweep (DTS) for systematic exploration of threshold calibration effects on crisp-set QCA results. These methods extend traditional robustness approaches by treating threshold variation as an exploratory tool for discovering causal structures. Built on top of the 'QCA' package by Dusa (2019) <[doi:10.1007/978-3-319-75668-4](https://doi.org/10.1007/978-3-319-75668-4)>, with function arguments following 'QCA' conventions. Based on set-theoretic methods by Ragin (2008) <[doi:10.7208/chicago/9780226702797.001.0001](https://doi.org/10.7208/chicago/9780226702797.001.0001)> and established robustness protocols by Rubinson et al. (2019) <[doi:10.1177/00491241211036158](https://doi.org/10.1177/00491241211036158)>.

Depends R (>= 4.0)

Imports QCA

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

License MIT + file LICENSE

URL <https://github.com/im-research-yt/TSQCA>,
<https://doi.org/10.5281/zenodo.17899390>

BugReports <https://github.com/im-research-yt/TSQCA/issues>

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Yuki Toyoda [aut, cre],
Japan Society for the Promotion of Science [fnd] (KAKENHI Grant Number JP20K01998)

Maintainer Yuki Toyoda <yuki.toyoda.ds@hosei.ac.jp>

Repository CRAN

Date/Publication 2026-02-18 09:40:02 UTC

Contents

config_chart_from_paths	2
config_chart_multi_solutions	4
ctSweepM	5
ctSweepS	8
dtSweep	11
extract_terms	15
format_qca_solution	15
format_qca_solutions	16
format_qca_term	17
generate_config_chart	17
generate_cross_threshold_chart	19
generate_report	20
generate_solution_note	22
identify_epi	23
otSweep	24
print.tsqca_result	27
sample_data	28
summary.tsqca_result	28

Index 30

config_chart_from_paths

Generate configuration chart from paths (simple interface)

Description

A simpler interface for generating configuration charts when you have paths directly (without a full QCA solution object).

Usage

```
config_chart_from_paths(
  paths,
  symbol_set = c("unicode", "ascii", "latex"),
  language = c("en", "ja"),
  condition_order = NULL,
  n_sol = 1L,
  solution_note = TRUE,
  solution_note_style = c("simple", "detailed"),
  epi_list = NULL
)
```

Arguments

paths	Character vector. Paths in QCA notation (e.g., "A*B*~C").
symbol_set	Character. One of "unicode", "ascii", or "latex".
language	Character. "en" for English, "ja" for Japanese.
condition_order	Character vector. Optional ordering of conditions.
n_sol	Integer. Number of equivalent solutions. If > 1, a note is added explaining that multiple solutions exist and M1 is shown. Default is 1.
solution_note	Logical. Whether to add solution note when n_sol > 1. Default is TRUE.
solution_note_style	Character. "simple" or "detailed". Default is "simple".
epi_list	Character vector. Essential prime implicants for detailed notes. Only used when solution_note_style = "detailed".

Value

Character string containing Markdown-formatted table.

Examples

```
# Simple usage with paths
paths <- c("A*B", "A*C*~D", "B*E")
chart <- config_chart_from_paths(paths)
cat(chart)

# With ASCII symbols
chart <- config_chart_from_paths(paths, symbol_set = "ascii")
cat(chart)

# With multiple solution note
chart <- config_chart_from_paths(paths, n_sol = 2)
cat(chart)

# With detailed note including EPIs
chart <- config_chart_from_paths(
  paths, n_sol = 2,
  solution_note_style = "detailed",
  epi_list = c("A*B")
)
cat(chart)
```

`config_chart_multi_solutions`*Generate configuration chart for multiple solutions (simple interface)*

Description

Generates separate configuration charts for multiple solutions.

Usage

```
config_chart_multi_solutions(  
  solutions,  
  symbol_set = c("unicode", "ascii", "latex"),  
  language = c("en", "ja"),  
  condition_order = NULL,  
  show_epi = FALSE  
)
```

Arguments

<code>solutions</code>	List of character vectors. Each element is a vector of paths for one solution.
<code>symbol_set</code>	Character. One of "unicode", "ascii", or "latex".
<code>language</code>	Character. "en" for English, "ja" for Japanese.
<code>condition_order</code>	Character vector. Optional ordering of conditions.
<code>show_epi</code>	Logical. Whether to identify and display Essential Prime Implicants (EPIs) in the note. Default is FALSE.

Value

Character string containing Markdown-formatted tables.

Examples

```
# Multiple solutions  
solutions <- list(  
  c("A*B", "C"),  
  c("A*B", "D"),  
  c("A*C")  
)  
chart <- config_chart_multi_solutions(solutions)  
cat(chart)  
  
# With EPI identification  
chart <- config_chart_multi_solutions(solutions, show_epi = TRUE)  
cat(chart)
```

ctSweepM

MCTS-QCA: Multi-condition threshold sweep

Description

Performs a grid search over thresholds of multiple X variables. For each combination of thresholds in `sweep_list`, the outcome Y and all X variables are binarized, and a crisp-set QCA is executed.

Usage

```
ctSweepM(
  dat,
  outcome = NULL,
  conditions = NULL,
  sweep_list,
  thrY,
  pre_calibrated = NULL,
  dir.exp = NULL,
  include = "",
  incl.cut = 0.8,
  n.cut = 1,
  pri.cut = 0,
  extract_mode = c("first", "all", "essential"),
  return_details = TRUE,
  Yvar = NULL,
  Xvars = NULL
)
```

Arguments

<code>dat</code>	Data frame containing the outcome and condition variables.
<code>outcome</code>	Character. Outcome variable name. Supports negation with tilde prefix (e.g., " <code>~Y</code> ") following QCA package conventions.
<code>conditions</code>	Character vector. Names of condition variables.
<code>sweep_list</code>	Named list. Each element is a numeric vector of candidate thresholds for the corresponding X. Names must match <code>conditions</code> . Variables listed in <code>pre_calibrated</code> do not need a <code>sweep_list</code> entry.
<code>thrY</code>	Numeric. Threshold for Y (fixed).
<code>pre_calibrated</code>	Character vector or NULL. Names of condition variables that have been pre-calibrated (e.g., via <code>QCA::calibrate()</code>) and should be passed through to <code>QCA::truthTable()</code> without binarization. These variables must contain values in the $[0, 1]$ range. Variables not listed here will be binarized using <code>sweep_list</code> thresholds as usual. Default is NULL (all variables binarized). It is recommended to sweep variables on their original (raw) scale rather than as pre-calibrated fuzzy values, because raw-scale thresholds are easier to interpret substantively.

<code>dir.exp</code>	Directional expectations for minimize. If NULL (default), no directional expectations are applied. To compute the intermediate solution , specify a numeric vector (1, 0, or -1 for each condition). Example: <code>dir.exp = c(1, 1, 1)</code> for three conditions all expected to contribute positively.
<code>include</code>	Inclusion rule for minimize. "" (default, QCA compatible) computes the complex solution without logical remainders. Use "?" to include logical remainders for parsimonious (with <code>dir.exp = NULL</code>) or intermediate solutions (with <code>dir.exp</code> specified).
<code>incl.cut</code>	Consistency cutoff for truthTable.
<code>n.cut</code>	Frequency cutoff for truthTable.
<code>pri.cut</code>	PRI cutoff for minimize.
<code>extract_mode</code>	Character. How to handle multiple solutions: "first" (default), "all", or "essential". See qca_extract for details.
<code>return_details</code>	Logical. If TRUE (default), returns both summary and detailed objects for use with <code>generate_report()</code> .
<code>Yvar</code>	Deprecated. Use <code>outcome</code> instead.
<code>Xvars</code>	Deprecated. Use <code>conditions</code> instead.

Value

If `return_details = FALSE`, a data frame with columns:

- `combo_id` — index of the threshold combination
- `threshold` — character string summarizing thresholds, e.g. "X1=6, X2=7, X3=7"
- `expression` — minimized solution expression
- `inclS` — solution consistency
- `covS` — solution coverage
- (additional columns depending on `extract_mode`)

If `return_details = TRUE`, a list with:

- `summary` — the data frame above
- `details` — per-combination list of `combo_id`, `thrX_vec`, `truth_table`, `solution`

Examples

```
# Load sample data
data(sample_data)

# === Three Types of QCA Solutions ===

# Quick demonstration with 2 conditions
sweep_list <- list(X1 = 7, X2 = 7)

# 1. Complex Solution (default, QCA compatible)
result_comp <- ctSweepM(
  dat = sample_data,
```

```

    outcome = "Y",
    conditions = c("X1", "X2"),
    sweep_list = sweep_list,
    thrY = 7
    # include = "" (default), dir.exp = NULL (default)
  )
  head(result_comp$summary)

# 2. Parsimonious Solution (include = "?")
result_pars <- ctSweepM(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2"),
  sweep_list = sweep_list,
  thrY = 7,
  include = "?" # Include logical remainders
)
head(result_pars$summary)

# 3. Intermediate Solution (include = "?" + dir.exp)
result_int <- ctSweepM(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2"),
  sweep_list = sweep_list,
  thrY = 7,
  include = "?",
  dir.exp = c(1, 1) # Positive expectations
)
head(result_int$summary)

# === Threshold Sweep Example ===

# Using 2 conditions and 2 threshold levels
sweep_list <- list(
  X1 = 6:7,
  X2 = 6:7
)

# Run multiple condition threshold sweep (complex solutions by default)
result_quick <- ctSweepM(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2"),
  sweep_list = sweep_list,
  thrY = 7
)
head(result_quick$summary)

# Run with negated outcome (~Y)
result_neg <- ctSweepM(
  dat = sample_data,
  outcome = "~Y",

```

```

    conditions = c("X1", "X2"),
    sweep_list = sweep_list,
    thrY = 7
  )
  head(result_neg$summary)

# Full multi-condition analysis (27 combinations)
sweep_list_full <- list(
  X1 = 6:8,
  X2 = 6:8,
  X3 = 6:8
)

result_full <- ctSweepM(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_list = sweep_list_full,
  thrY = 7
)
head(result_full$summary)

```

ctSweepS

CTS-QCA: Single-condition threshold sweep

Description

Performs a threshold sweep for one focal condition X. For each threshold in sweep_range, the outcome Y and all X variables are binarized using user-specified thresholds, and a crisp-set QCA is executed.

Usage

```

ctSweepS(
  dat,
  outcome = NULL,
  conditions = NULL,
  sweep_var,
  sweep_range,
  thrY,
  thrX_default = 7,
  pre_calibrated = NULL,
  dir.exp = NULL,
  include = "",
  incl.cut = 0.8,
  n.cut = 1,
  pri.cut = 0,

```

```

extract_mode = c("first", "all", "essential"),
return_details = TRUE,
Yvar = NULL,
Xvars = NULL
)

```

Arguments

<code>dat</code>	Data frame containing the outcome and condition variables.
<code>outcome</code>	Character. Outcome variable name. Supports negation with tilde prefix (e.g., " <code>~Y</code> ") following QCA package conventions.
<code>conditions</code>	Character vector. Names of condition variables.
<code>sweep_var</code>	Character. Name of the condition to be swept. Must be one of conditions.
<code>sweep_range</code>	Numeric vector. Candidate thresholds for <code>sweep_var</code> .
<code>thrY</code>	Numeric. Threshold for Y (fixed).
<code>thrX_default</code>	Numeric. Default threshold for non-swept X variables. Variables listed in <code>pre_calibrated</code> do not require this threshold.
<code>pre_calibrated</code>	Character vector or NULL. Names of condition variables that have been pre-calibrated (e.g., via <code>QCA::calibrate()</code>) and should be passed through to <code>QCA::truthTable()</code> without binarization. These variables must contain values in the $[0, 1]$ range. Variables not listed here will be binarized using <code>thrX_default</code> as usual. Default is NULL (all variables binarized). It is recommended to sweep variables on their original (raw) scale rather than as pre-calibrated fuzzy values, because raw-scale thresholds are easier to interpret substantively.
<code>dir.exp</code>	Directional expectations for minimize. If NULL (default), no directional expectations are applied. To compute the intermediate solution , specify a numeric vector (1, 0, or -1 for each condition). Example: <code>dir.exp = c(1, 1, 1)</code> for three conditions all expected to contribute positively.
<code>include</code>	Inclusion rule for minimize. "" (default, QCA compatible) computes the complex solution without logical remainders. Use "?" to include logical remainders for parsimonious (with <code>dir.exp = NULL</code>) or intermediate solutions (with <code>dir.exp</code> specified).
<code>incl.cut</code>	Consistency cutoff for <code>truthTable</code> .
<code>n.cut</code>	Frequency cutoff for <code>truthTable</code> .
<code>pri.cut</code>	PRI cutoff for minimize.
<code>extract_mode</code>	Character. How to handle multiple solutions: "first" (default), "all", or "essential". See qca_extract for details.
<code>return_details</code>	Logical. If TRUE (default), returns both summary and detailed objects for use with <code>generate_report()</code> .
<code>Yvar</code>	Deprecated. Use <code>outcome</code> instead.
<code>Xvars</code>	Deprecated. Use <code>conditions</code> instead.

Value

If `return_details = FALSE`, a data frame with columns:

- `threshold` — swept threshold for `sweep_var`
- `expression` — minimized solution expression
- `inclS` — solution consistency
- `covS` — solution coverage
- (additional columns depending on `extract_mode`)

If `return_details = TRUE`, a list with:

- `summary` — the data frame above
- `details` — per-threshold list of `threshold`, `thrX_vec`, `truth_table`, `solution`

Examples

```
# Load sample data
data(sample_data)

# === Three Types of QCA Solutions ===

# 1. Complex Solution (default, QCA compatible)
result_comp <- ctSweepS(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_var = "X3",
  sweep_range = 7,
  thrY = 7,
  thrX_default = 7
  # include = "" (default), dir.exp = NULL (default)
)
head(result_comp$summary)

# 2. Parsimonious Solution (include = "?")
result_pars <- ctSweepS(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_var = "X3",
  sweep_range = 7,
  thrY = 7,
  thrX_default = 7,
  include = "?" # Include logical remainders
)
head(result_pars$summary)

# 3. Intermediate Solution (include = "?" + dir.exp)
result_int <- ctSweepS(
  dat = sample_data,
  outcome = "Y",
```

```

    conditions = c("X1", "X2", "X3"),
    sweep_var = "X3",
    sweep_range = 7,
    thrY = 7,
    thrX_default = 7,
    include = "?",
    dir.exp = c(1, 1, 1) # All conditions expected positive
  )
  head(result_int$summary)

# === Threshold Sweep Example ===

# Run single condition threshold sweep on X3 (complex solutions by default)
result <- ctSweepS(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_var = "X3",
  sweep_range = 6:8,
  thrY = 7,
  thrX_default = 7
)
head(result$summary)

# Run with negated outcome (~Y)
result_neg <- ctSweepS(
  dat = sample_data,
  outcome = "~Y",
  conditions = c("X1", "X2", "X3"),
  sweep_var = "X3",
  sweep_range = 6:8,
  thrY = 7,
  thrX_default = 7
)
head(result_neg$summary)

```

dtSweep

DTS-QCA: Two-dimensional X-Y threshold sweep

Description

Sweeps thresholds for multiple X variables and the outcome Y jointly. For each combination of X thresholds and each candidate Y threshold, the data are binarized and a crisp-set QCA is executed.

Usage

```

dtSweep(
  dat,
  outcome = NULL,
  conditions = NULL,

```

```

sweep_list_X,
sweep_range_Y,
pre_calibrated = NULL,
dir.exp = NULL,
include = "",
incl.cut = 0.8,
n.cut = 1,
pri.cut = 0,
extract_mode = c("first", "all", "essential"),
return_details = TRUE,
Yvar = NULL,
Xvars = NULL
)

```

Arguments

<code>dat</code>	Data frame containing the outcome and condition variables.
<code>outcome</code>	Character. Outcome variable name. Supports negation with tilde prefix (e.g., " <code>~Y</code> ") following QCA package conventions.
<code>conditions</code>	Character vector. Names of condition variables.
<code>sweep_list_X</code>	Named list. Each element is a numeric vector of candidate thresholds for the corresponding X. Variables listed in <code>pre_calibrated</code> do not need a <code>sweep_list_X</code> entry.
<code>sweep_range_Y</code>	Numeric vector. Candidate thresholds for Y.
<code>pre_calibrated</code>	Character vector or NULL. Names of condition variables that have been pre-calibrated (e.g., via <code>QCA::calibrate()</code>) and should be passed through to <code>QCA::truthTable()</code> without binarization. These variables must contain values in the $[0, 1]$ range. Variables not listed here will be binarized using <code>sweep_list_X</code> thresholds as usual. Default is NULL (all variables binarized). It is recommended to sweep variables on their original (raw) scale rather than as pre-calibrated fuzzy values, because raw-scale thresholds are easier to interpret substantively.
<code>dir.exp</code>	Directional expectations for minimize. If NULL (default), no directional expectations are applied. To compute the intermediate solution , specify a numeric vector (1, 0, or -1 for each condition). Example: <code>dir.exp = c(1, 1, 1)</code> for three conditions all expected to contribute positively.
<code>include</code>	Inclusion rule for minimize. "" (default, QCA compatible) computes the complex solution without logical remainders. Use "?" to include logical remainders for parsimonious (with <code>dir.exp = NULL</code>) or intermediate solutions (with <code>dir.exp</code> specified).
<code>incl.cut</code>	Consistency cutoff for <code>truthTable</code> .
<code>n.cut</code>	Frequency cutoff for <code>truthTable</code> .
<code>pri.cut</code>	PRI cutoff for minimize.
<code>extract_mode</code>	Character. How to handle multiple solutions: "first" (default), "all", or "essential". See qca_extract for details.
<code>return_details</code>	Logical. If TRUE (default), returns both summary and detailed objects for use with <code>generate_report()</code> .

Yvar	Deprecated. Use outcome instead.
Xvars	Deprecated. Use conditions instead.

Value

If `return_details = FALSE`, a data frame with columns:

- `combo_id` — index of threshold combination
- `thrY` — threshold for Y
- `thrX` — character summary of X thresholds
- `expression` — minimized solution expression
- `inclS` — solution consistency
- `covS` — solution coverage
- (additional columns depending on `extract_mode`)

If `return_details = TRUE`, a list with:

- `summary` — the data frame above
- `details` — list of runs with `combo_id`, `thrY`, `thrX_vec`, `truth_table`, `solution`

Examples

```
# Load sample data
data(sample_data)

# === Three Types of QCA Solutions ===

# Quick demonstration with 2 conditions
sweep_list_X <- list(X1 = 7, X2 = 7)
sweep_range_Y <- 7

# 1. Complex Solution (default, QCA compatible)
result_comp <- dtSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2"),
  sweep_list_X = sweep_list_X,
  sweep_range_Y = sweep_range_Y
  # include = "" (default), dir.exp = NULL (default)
)
head(result_comp$summary)

# 2. Parsimonious Solution (include = "?")
result_pars <- dtSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2"),
  sweep_list_X = sweep_list_X,
  sweep_range_Y = sweep_range_Y,
  include = "?" # Include logical remainders
```

```
)
head(result_pars$summary)

# 3. Intermediate Solution (include = "?" + dir.exp)
result_int <- dtSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2"),
  sweep_list_X = sweep_list_X,
  sweep_range_Y = sweep_range_Y,
  include = "?",
  dir.exp = c(1, 1) # Positive expectations
)
head(result_int$summary)

# === Threshold Sweep Example ===

# Using 2 conditions and 2 threshold levels
sweep_list_X <- list(
  X1 = 6:7,
  X2 = 6:7
)
sweep_range_Y <- 6:7

# Run dual threshold sweep (complex solutions by default)
result_quick <- dtSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2"),
  sweep_list_X = sweep_list_X,
  sweep_range_Y = sweep_range_Y
)
head(result_quick$summary)

# Full analysis with 3 conditions (81 combinations)
sweep_list_X_full <- list(
  X1 = 6:8,
  X2 = 6:8,
  X3 = 6:8
)
sweep_range_Y_full <- 6:8

result_full <- dtSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_list_X = sweep_list_X_full,
  sweep_range_Y = sweep_range_Y_full
)
head(result_full$summary)
```

extract_terms	<i>Extract and format terms from solutions</i>
---------------	--

Description

Extracts individual terms from solution expressions and returns formatted unique terms.

Usage

```
extract_terms(solutions, var_names, use_tilde = TRUE)
```

Arguments

solutions	Character vector. Solution expressions.
var_names	Character vector. Variable names used in the analysis.
use_tilde	Logical. If TRUE, negation is represented as ~VAR.

Value

List with:

- all_terms — all terms (with duplicates)
- unique_terms — unique terms
- n_total — total term count
- n_unique — unique term count

Examples

```
var_names <- c("X1", "X2", "X3")
solutions <- c("X1*X2 + X3", "X1*X2 + X1*X3")
extract_terms(solutions, var_names)
```

format_qca_solution	<i>Format a QCA solution expression</i>
---------------------	---

Description

Formats a complete solution expression (multiple terms joined by +).

Usage

```
format_qca_solution(solution, var_names, use_tilde = TRUE)
```

Arguments

`solution` Character. A solution expression (e.g., "KSPRVT + ~KPRPRD").

`var_names` Character vector. Variable names used in the analysis.

`use_tilde` Logical. If TRUE, negation is represented as ~VAR.

Value

Character. The formatted solution expression.

Examples

```
var_names <- c("KSP", "KPR", "PRD", "RVT", "RCM")
format_qca_solution("KSPRVT + ~KPRPRD + RCM", var_names)
# Returns: "KSP*RVT + ~KPR*PRD + RCM"
```

`format_qca_solutions` *Format multiple QCA solutions*

Description

Formats a vector of solution expressions.

Usage

```
format_qca_solutions(solutions, var_names, use_tilde = TRUE)
```

Arguments

`solutions` Character vector. Solution expressions from `minimize()`.

`var_names` Character vector. Variable names used in the analysis.

`use_tilde` Logical. If TRUE, negation is represented as ~VAR.

Value

Character vector. Formatted solution expressions.

Examples

```
var_names <- c("KSP", "KPR", "PRD", "RVT", "RCM")
solutions <- c("KSPRVT + RCM", "~KPRPRD")
format_qca_solutions(solutions, var_names)
```

format_qca_term	<i>Format a single QCA term</i>
-----------------	---------------------------------

Description

Inserts * between variables in a term where it may have been omitted.

Usage

```
format_qca_term(term, var_names, use_tilde = TRUE)
```

Arguments

term	Character. A single term (e.g., "KSPRVT" or "~KPR*PRD").
var_names	Character vector. Variable names used in the analysis.
use_tilde	Logical. If TRUE, negation is represented as ~VAR. If FALSE, negation is represented as lowercase (e.g., var).

Value

Character. The formatted term with * between all variables.

Examples

```
var_names <- c("KSP", "KPR", "PRD", "RVT", "RCM")
format_qca_term("KSPRVTRCM", var_names)
# Returns: "KSP*RVT*RCM"

format_qca_term("~KPRPRD", var_names)
# Returns: "~KPR*PRD"
```

generate_config_chart	<i>Generate Configuration Chart from QCA Solution</i>
-----------------------	---

Description

Creates a Markdown-formatted configuration chart (Fiss-style table) from QCA minimization results. Supports single solution with multiple paths, and multiple solutions (displayed as separate tables).

Usage

```
generate_config_chart(
  sol,
  symbol_set = c("unicode", "ascii", "latex"),
  include_metrics = TRUE,
  language = c("en", "ja"),
  condition_order = NULL
)
```

Arguments

<code>sol</code>	A solution object returned by <code>QCA::minimize()</code> , or a list containing solution information.
<code>symbol_set</code>	Character. One of "unicode", "ascii", or "latex". Default is "unicode".
<code>include_metrics</code>	Logical. Whether to include consistency/coverage metrics in the table. Default is TRUE.
<code>language</code>	Character. "en" for English, "ja" for Japanese. Default is "en".
<code>condition_order</code>	Character vector. Optional ordering of conditions in the table rows. If NULL, conditions are ordered as they appear in paths.

Value

Character string containing Markdown-formatted table(s).

Examples

```
## Not run:
# After running QCA::minimize()
library(QCA)
tt <- truthTable(data, outcome = "Y", conditions = c("A", "B", "C"))
sol <- minimize(tt, include = "?", details = TRUE)

# Generate configuration chart
chart <- generate_config_chart(sol)
cat(chart)

# For LaTeX/PDF output (e.g., rarticles)
chart <- generate_config_chart(sol, symbol_set = "latex")

# ASCII for maximum compatibility
chart <- generate_config_chart(sol, symbol_set = "ascii")

# Japanese labels
chart <- generate_config_chart(sol, language = "ja")

## End(Not run)
```

`generate_cross_threshold_chart`*Generate cross-threshold configuration chart from sweep results*

Description

Creates a configuration chart from threshold sweep results. Supports two levels of aggregation: solution-term level (Fiss-style, default) and threshold-level summary.

Usage

```
generate_cross_threshold_chart(  
  result,  
  conditions = NULL,  
  symbol_set = c("unicode", "ascii", "latex"),  
  chart_level = c("term", "summary"),  
  language = c("en", "ja")  
)
```

Arguments

<code>result</code>	A result object from any Sweep function (<code>otSweep</code> , <code>ctSweepS</code> , <code>ctSweepM</code> , or <code>dtSweep</code>).
<code>conditions</code>	Character vector. Condition names for row ordering. If <code>NULL</code> , automatically extracted from expressions.
<code>symbol_set</code>	Character. One of "unicode", "ascii", or "latex". Default is "unicode".
<code>chart_level</code>	Character. Chart aggregation level: "term" (default) produces solution-term level charts following Fiss (2011) notation, where each column represents one prime implicant. "summary" produces threshold-level summaries where each column represents one threshold, aggregating all configurations.
<code>language</code>	Character. "en" for English, "ja" for Japanese.

Value

Character string containing Markdown-formatted table.

Examples

```
## Not run:  
data(sample_data)  
result <- otSweep(  
  dat = sample_data,  
  outcome = "Y",  
  conditions = c("X1", "X2", "X3"),  
  sweep_range = 6:8,  
  thrX = c(X1 = 7, X2 = 7, X3 = 7)  
)
```

```

# Solution-term level, Fiss-style (default)
chart <- generate_cross_threshold_chart(result, c("X1", "X2", "X3"))
cat(chart)

# Threshold-level summary
chart <- generate_cross_threshold_chart(result, c("X1", "X2", "X3"),
                                       chart_level = "summary")

cat(chart)

## End(Not run)

```

generate_report

Generate Markdown Report for QCA Analysis

Description

Creates a markdown report from QCA analysis results. Supports two formats: "full" (comprehensive) and "simple" (for manuscripts).

Usage

```

generate_report(
  result,
  output_file = "qca_report.md",
  format = c("full", "simple"),
  title = "QCA Analysis Report",
  dat = NULL,
  desc_vars = NULL,
  include_chart = TRUE,
  chart_symbol_set = c("unicode", "ascii", "latex"),
  chart_level = c("term", "summary"),
  solution_note = TRUE,
  solution_note_style = c("simple", "detailed"),
  solution_note_lang = c("en", "ja"),
  include_raw_output = TRUE
)

```

Arguments

result	A result object from any Sweep function with return_details = TRUE.
output_file	Character. Path to output markdown file.
format	Character. Report format: "full" or "simple".
title	Character. Report title.
dat	Optional data frame. Original data for descriptive statistics.
desc_vars	Optional character vector. Variables for descriptive statistics. If NULL and dat is provided, uses Yvar and Xvars from params.

include_chart	Logical. If TRUE (default), includes configuration charts (Fiss-style tables) in the report for each threshold.
chart_symbol_set	Character. Symbol set for configuration charts: "unicode" (default), "ascii", or "latex".
chart_level	Character. Chart aggregation level: "term" (default) produces solution-term level charts following Fiss (2011) notation, where each column represents one prime implicant (sufficient configuration). This format is recommended for academic publications. "summary" produces threshold-level summaries where each column represents one threshold, aggregating all configurations.
solution_note	Logical. If TRUE (default), adds a note when multiple equivalent solutions exist explaining that M1 is shown.
solution_note_style	Character. Style of solution note: "simple" (default) or "detailed" (includes EPIs).
solution_note_lang	Character. Language for solution notes: "en" (default) or "ja".
include_raw_output	Logical. If TRUE (default), includes the raw QCA package output (print(sol)) for each threshold for verification purposes.

Value

Invisibly returns the path to the generated report.

Examples

```
## Not run:
data(sample_data)
thrX <- c(X1 = 7, X2 = 7, X3 = 7)

result <- otSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_range = 6:8,
  thrX = thrX,
  return_details = TRUE
)

# With descriptive statistics and configuration charts
generate_report(result, "my_report.md", format = "full",
  dat = sample_data, include_chart = TRUE)

# Without configuration charts
generate_report(result, "my_report.md", format = "simple",
  include_chart = FALSE)

# With Fiss-style term-level charts (default, recommended for publications)
```

```

generate_report(result, "my_report.md", format = "full")

# With threshold-level summary charts
generate_report(result, "my_report.md", format = "full",
               chart_level = "summary")

# With detailed solution notes (including EPIs)
generate_report(result, "my_report.md", format = "full",
               solution_note_style = "detailed")

## End(Not run)

```

```
generate_solution_note
```

Generate solution note for multiple solutions

Description

Creates a note explaining that multiple equivalent solutions exist and that the displayed configuration is based on M1.

Usage

```

generate_solution_note(
  n_sol,
  epi_list = NULL,
  style = c("simple", "detailed"),
  language = c("en", "ja"),
  format = c("markdown", "latex")
)

```

Arguments

n_sol	Integer. Number of solutions.
epi_list	Character vector. Essential prime implicants (NULL to omit).
style	Character. "simple" or "detailed".
language	Character. "en" or "ja".
format	Character. "markdown" or "latex".

Value

Character string of the note, or empty string if n_sol <= 1.

Examples

```
# Simple note
generate_solution_note(2, style = "simple")

# Detailed note with EPIs
generate_solution_note(3, epi_list = c("A*B", "C"), style = "detailed")

# Japanese
generate_solution_note(2, style = "simple", language = "ja")
```

`identify_epi`*Identify Essential Prime Implicants from multiple solutions*

Description

Finds terms that appear in ALL solutions (EPIs) versus terms that appear in only some solutions (SPIs).

Usage

```
identify_epi(solutions)
```

Arguments

`solutions` List of solution vectors. Each element is a character vector of terms for one solution.

Value

List with:

- `epi` — Essential prime implicants (in all solutions)
- `spi` — Selective prime implicants (in some solutions)
- `n_solutions` — Number of solutions

Examples

```
solutions <- list(
  c("A*B", "C", "D"),
  c("A*B", "C", "E"),
  c("A*B", "C", "F")
)
result <- identify_epi(solutions)
# result$epi = c("A*B", "C")
# result$spi = c("D", "E", "F")
```

otSweep

*OTS-QCA: Outcome threshold sweep***Description**

Sweeps the threshold of the outcome Y while keeping the thresholds of all X conditions fixed.

Usage

```
otSweep(
  dat,
  outcome = NULL,
  conditions = NULL,
  sweep_range,
  thrX,
  pre_calibrated = NULL,
  dir.exp = NULL,
  include = "",
  incl.cut = 0.8,
  n.cut = 1,
  pri.cut = 0,
  extract_mode = c("first", "all", "essential"),
  return_details = TRUE,
  Yvar = NULL,
  Xvars = NULL
)
```

Arguments

dat	Data frame containing the outcome and condition variables.
outcome	Character. Outcome variable name. Supports negation with tilde prefix (e.g., "~Y") following QCA package conventions.
conditions	Character vector. Names of condition variables.
sweep_range	Numeric vector. Candidate thresholds for Y.
thrX	Named numeric vector. Fixed thresholds for X variables. Names must match the conditions that require binarization. Variables listed in pre_calibrated do not need a thrX entry.
pre_calibrated	Character vector or NULL. Names of condition variables that have been pre-calibrated (e.g., via <code>QCA::calibrate()</code>) and should be passed through to <code>QCA::truthTable()</code> without binarization. These variables must contain values in the $[0, 1]$ range. Variables not listed here will be binarized using thrX thresholds as usual. Default is NULL (all variables binarized). It is recommended to sweep variables on their original (raw) scale rather than as pre-calibrated fuzzy values, because raw-scale thresholds are easier to interpret substantively.

<code>dir.exp</code>	Directional expectations for minimize. If NULL (default), no directional expectations are applied. To compute the intermediate solution , specify a numeric vector (1, 0, or -1 for each condition). Example: <code>dir.exp = c(1, 1, 1)</code> for three conditions all expected to contribute positively.
<code>include</code>	Inclusion rule for minimize. "" (default, QCA compatible) computes the complex solution without logical remainders. Use "?" to include logical remainders for parsimonious (with <code>dir.exp = NULL</code>) or intermediate solutions (with <code>dir.exp</code> specified).
<code>incl.cut</code>	Consistency cutoff for truthTable.
<code>n.cut</code>	Frequency cutoff for truthTable.
<code>pri.cut</code>	PRI cutoff for minimize.
<code>extract_mode</code>	Character. How to handle multiple solutions: "first" (default), "all", or "essential". See qca_extract for details.
<code>return_details</code>	Logical. If TRUE (default), returns both summary and detailed objects for use with <code>generate_report()</code> .
<code>Yvar</code>	Deprecated. Use <code>outcome</code> instead.
<code>Xvars</code>	Deprecated. Use <code>conditions</code> instead.

Value

If `return_details = FALSE`, a data frame with columns:

- `thrY` — threshold for Y
- `expression` — minimized solution expression
- `inclS` — solution consistency
- `covS` — solution coverage
- (additional columns depending on `extract_mode`)

If `return_details = TRUE`, a list with:

- `summary` — the data frame above
- `details` — per-Y-threshold list of `thrY`, `thrX_vec`, `truth_table`, `solution`

Examples

```
# Load sample data
data(sample_data)

# Set fixed thresholds for conditions
thrX <- c(X1 = 7, X2 = 7, X3 = 7)

# === Three Types of QCA Solutions ===

# 1. Complex Solution (default, QCA compatible)
# Does not use logical remainders (most conservative)
result_comp <- otSweep(
  dat = sample_data,
```

```

    outcome = "Y",
    conditions = c("X1", "X2", "X3"),
    sweep_range = 7,
    thrX = thrX
    # include = "" (default), dir.exp = NULL (default)
  )
  head(result_comp$summary)

# 2. Parsimonious Solution (include = "?")
# Uses logical remainders without directional expectations
result_pars <- otSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_range = 7,
  thrX = thrX,
  include = "?" # Include logical remainders
)
head(result_pars$summary)

# 3. Intermediate Solution (include = "?" + dir.exp)
# Uses logical remainders with directional expectations
result_int <- otSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_range = 7,
  thrX = thrX,
  include = "?",
  dir.exp = c(1, 1, 1) # All conditions expected positive
)
head(result_int$summary)

# === Threshold Sweep Example ===

# Sweep with complex solutions (default)
result_sweep <- otSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_range = 6:8,
  thrX = thrX
)
head(result_sweep$summary)

# Run with negated outcome (~Y)
# Analyzes conditions for Y < threshold
result_neg <- otSweep(
  dat = sample_data,
  outcome = "~Y",
  conditions = c("X1", "X2", "X3"),
  sweep_range = 6:8,
  thrX = thrX
)

```

```
)  
head(result_neg$summary)
```

```
print.tsqca_result    Print method for TSQCA results
```

Description

Displays a concise overview of TSQCA analysis results.

Usage

```
## S3 method for class 'tsqca_result'  
print(x, ...)  
  
## S3 method for class 'otSweep_result'  
print(x, ...)  
  
## S3 method for class 'dtSweep_result'  
print(x, ...)  
  
## S3 method for class 'ctSweepS_result'  
print(x, ...)  
  
## S3 method for class 'ctSweepM_result'  
print(x, ...)
```

Arguments

x A TSQCA result object returned by one of the sweep functions.
... Additional arguments (ignored).

Value

Invisibly returns x.

Examples

```
data(sample_data)  
result <- otSweep(  
  dat = sample_data,  
  outcome = "Y",  
  conditions = c("X1", "X2", "X3"),  
  sweep_range = 6:8,  
  thrX = c(X1 = 7, X2 = 7, X3 = 7)  
)  
print(result)
```

sample_data	<i>Sample dataset for TSQCA examples</i>
-------------	--

Description

A small artificial dataset with variables:

Y Outcome (numeric)

X1 Condition 1

X2 Condition 2

X3 Condition 3

Usage

sample_data

Format

A data frame with 80 rows and 4 variables.

summary.tsqca_result	<i>Summary method for TSQCA results</i>
----------------------	---

Description

Displays detailed results table with solution formulas and fit measures.

Usage

```
## S3 method for class 'tsqca_result'  
summary(object, ...)
```

```
## S3 method for class 'otSweep_result'  
summary(object, ...)
```

```
## S3 method for class 'dtSweep_result'  
summary(object, ...)
```

```
## S3 method for class 'ctSweepS_result'  
summary(object, ...)
```

```
## S3 method for class 'ctSweepM_result'  
summary(object, ...)
```

Arguments

object A TSQCA result object returned by one of the sweep functions.
... Additional arguments (ignored).

Value

Invisibly returns object.

Examples

```
data(sample_data)
result <- otSweep(
  dat = sample_data,
  outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_range = 6:8,
  thrX = c(X1 = 7, X2 = 7, X3 = 7)
)
summary(result)
```

Index

* datasets

- sample_data, 28
- config_chart_from_paths, 2
- config_chart_multi_solutions, 4
- ctSweepM, 5
- ctSweepS, 8
- dtSweep, 11
- extract_terms, 15
- format_qca_solution, 15
- format_qca_solutions, 16
- format_qca_term, 17
- generate_config_chart, 17
- generate_cross_threshold_chart, 19
- generate_report, 20
- generate_solution_note, 22
- identify_epi, 23
- otSweep, 24
- print.ctSweepM_result
 - (print.tsqca_result), 27
- print.ctSweepS_result
 - (print.tsqca_result), 27
- print.dtSweep_result
 - (print.tsqca_result), 27
- print.otSweep_result
 - (print.tsqca_result), 27
- print.tsqca_result, 27
- qca_extract, 6, 9, 12, 25
- sample_data, 28
- summary.ctSweepM_result
 - (summary.tsqca_result), 28
- summary.ctSweepS_result
 - (summary.tsqca_result), 28

- summary.dtSweep_result
 - (summary.tsqca_result), 28
- summary.otSweep_result
 - (summary.tsqca_result), 28
- summary.tsqca_result, 28
- truthTable, 9