Package 'ThSQCA'

October 5, 2026

Type Package

Title Threshold-Sweep QCA

Version 2.0.9

Description Provides threshold sweep methods for Qualitative Comparative Analysis (QCA). Implements Condition Threshold Sweep (CTS, for one or several conditions), 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 explicit analytical dimension and recording the sufficiency solution obtained at each threshold setting. Also provides Fiss (2011) <doi:10.5465/amj.2011.60263120> core/peripheral condition classification via compute_fiss_core() and generate_fiss_chart(), enabling four-symbol configuration charts that distinguish core conditions (conditions of the parsimonious term contained in each configuration) from peripheral conditions (intermediate only). Built on top of the 'QCA' package by Dusa (2019) <doi: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> and established robustness protocols by Oana and Schneider (2024) <doi:10.1177/00491241211036158>. The threshold-sweep framework is described in Toyoda (2026) <doi:10.1007/s11135-026-03092-3>.
This package supersedes 'TSQCA'; see the NEWS file for migration guidance.

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/ThSQCA, https://doi.org/10.5281/zenodo.17899390

BugReports https://github.com/im-research-yt/ThSQCA/issues

Date 2026-10-04

Encoding UTF-8

LazyData true

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

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 2026-10-04 23:10:08 UTC

Contents

compute_fiss_core . . . 3
config_chart_from_paths . . . 5
config_chart_multi_solutions . . . 7
ctSweepM . . . 8
ctSweepS . . . 11
dtSweep . . . 15
extract_terms . . . 18
format_qca_solution . . . 19
format_qca_solutions . . . 20
format_qca_term . . . 20
generate_config_chart . . . 21
generate_cross_threshold_chart . . . 22
generate_fiss_chart . . . 23
generate_report . . . 25
generate_solution_note . . . 27
identify_epi . . . 28
otSweep . . . 28
print.tsqca_result . . . 32
print_fiss_summary . . . 33
sample_data . . . 33
summary.tsqca_result . . . 34

Index **35**

compute_fiss_core *Compute Fiss Core/Peripheral Classification for Sweep Results*

Description

Takes a threshold-sweep result produced by `otSweep` or `ctSweepS` and augments it with the Fiss (2011) core/peripheral classification of the intermediate solution. Results of `ctSweepM` and `dtSweep` are not supported yet and give an error.

Usage

```
compute_fiss_core(result, conditions = NULL)
```

Arguments

`result`	A result of `otSweep()` or `ctSweepS()` run with `include = "?"`, `dir.exp` and `return_details = TRUE`.
`conditions`	Character vector. Condition names (used for consistent row ordering in charts). If `NULL`, extracted automatically.

Details

The classification requires that:

- The sweep was run with `include = "?"` (to allow parsimonious computation)
- `return_details = TRUE` was used (truth tables must be stored)
- `dir.exp` was specified (i.e., the sweep produced intermediate solutions; core/peripheral is only meaningful when comparing parsimonious and intermediate solutions)

For each threshold in the result, this function:

1. Retrieves the intermediate solution already stored in `result$details`, and takes the model reported in the sweep summary (M1).
2. Identifies the parsimonious solution(s) from which `QCA::minimize()` derived that model. With `dir.exp`, QCA stores one entry in `sol$i.sol` for each pair of a complex solution and a parsimonious minimal solution (`C1P1`, `C1P2`, ...; see `print(sol)`), holding the parsimonious model in `$p.sol` and the intermediate model(s) obtained from it in `$solution`. Every entry that lists M1 is treated as a source.
3. Classifies each term (configuration) of M1: the conditions of the source parsimonious term(s) contained in that term are **core**; its other conditions are **peripheral**.

Value

The original `result` object with an additional `$fiss_core` slot: a named list keyed by threshold (character), each entry containing:

- `parsim_expression` — the parsimonious solution(s) M1 was compared with; several are shown with QCA's labels, e.g. `"P1: ...; P2: ..."`

- `interm_expression` — the intermediate solution classified (M1)
- `parsim_n_solutions` — number of tied parsimonious solutions on the truth table
- `interm_n_solutions` — number of intermediate minimal solutions
- `parsim_sources` — names of the `i.sol` entries M1 was derived from (e.g. `"C1P1"`), or `NA` when unavailable
- `classification` — data frame with columns `term_idx`, `term_expr`, `condition`, `status`, `type`

Relation to Fiss (2011)

Fiss (2011) defines core conditions as those that are part of both the parsimonious and the intermediate solution, and peripheral conditions as those that are eliminated in the parsimonious solution and therefore appear only in the intermediate solution. His solution tables apply this configuration by configuration: solutions are grouped by their core conditions, and the same condition can be core in one configuration and peripheral in another. This function follows that practice. A parsimonious term is contained in an intermediate term when every condition it specifies has the same status (present or absent) in the intermediate term; the core conditions of the intermediate term are the conditions of all parsimonious terms contained in it (one of the configurations in Fiss's high-performance table contains two parsimonious terms and has the core conditions of both).

Example: with the parsimonious solution `~A*E + A*B`, the intermediate term `~A*~B*C*E` contains `~A*E`, so `~A` and `E` are core and `~B` and `C` are peripheral, even though `B` occurs in the other parsimonious term. Versions up to 2.0.7 compared each condition with the whole parsimonious solution and reported `~B` as core here, which does not match how Fiss's tables are built.

Fiss (2011) does not discuss two situations, which this function handles conservatively and reports with a warning:

- **Tied parsimonious solutions.** If QCA derived M1 from a single parsimonious solution, that solution alone decides. If QCA derived the same M1 from several tied parsimonious solutions (the directional expectations do not single one out), a condition is core only if it is core relative to every one of them. For example, if M1 = `SUP + TRU*PRC` is derived both from `TRU + SUP` and from `PRC + SUP`, then `TRU` is core relative to the first and `PRC` relative to the second, so both are reported as peripheral. Fiss grounds coreness in the strength of the evidence; a condition whose status depends on which tied solution is chosen is not treated as strongly supported. To report core/peripheral status relative to one particular parsimonious solution, state that choice explicitly.
- **No contained parsimonious term.** An intermediate term can be covered by the parsimonious solution without containing any single parsimonious term (for example `B*C*D*E` with the parsimonious solution `~A*E + A*B`). Its conditions are then all classified as peripheral.

Only M1 is classified. When the intermediate solution itself has several minimal models (see `n_solutions` in the sweep summary and `interm_n_solutions` below), the other models are not included in the classification or the chart. If the derivation cannot be read from the stored solution (no `$i.sol` or `$p.sol`), M1 is compared with every tied parsimonious solution.

References

Fiss, P. C. (2011). Building better causal theories: A fuzzy set approach to typologies in organization research. *Academy of Management Journal*, 54(2), 393-420.

See Also

generate_fiss_chart

Examples

```
## Not run:
library(ThSQCA)
data(sample_data)

# Step 1: Run intermediate sweep (dir.exp required)
res <- otSweep(
  dat        = sample_data,
  outcome    = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_range = 6:8,
  thrX       = c(X1 = 7, X2 = 7, X3 = 7),
  include    = "?",
  dir.exp    = c(1, 1, 1),
  return_details = TRUE
)

# Step 2: Augment with Fiss core/peripheral classification
res_fiss <- compute_fiss_core(res, conditions = c("X1", "X2", "X3"))

# Step 3: Generate Fiss-style chart
cat(generate_fiss_chart(res_fiss, symbol_set = "unicode"))

## End(Not run)
```

---

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

solutions	List of character vectors. Each element is a vector of paths for one solution.
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.
show_epi	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 *CTS (multiple): 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,
  thrX_default = NULL,
  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_list`	Named list. Each element is a numeric vector of candidate thresholds for the corresponding X. Names must match `conditions`. Variables listed in `pre_calibrated` do not need a `sweep_list` entry. Every condition must receive a threshold from one of `sweep_list`, `pre_calibrated`, or `thrX_default`; a condition covered by none of these raises an error. To hold a condition at a fixed value while sweeping others, give it a length-one entry (e.g. `list(SPV = 6:8, PRD = 7)`).
`thrY`	Numeric. Threshold for Y (fixed).
`thrX_default`	Numeric or `NULL`. Threshold used to binarize any condition that appears in neither `sweep_list` nor `pre_calibrated`, mirroring the argument of the same name in `ctSweepS`. Default is `NULL`, in which case such a condition is an error rather than being silently dropped.

pre_calibrated	Character vector or NULL. Names of condition variables that have been pre-calibrated (e.g., via QCA::calibrate()) and should be passed through to QCA::truthTable() without binarization. These variables must contain values in the [0, 1] range. Variables not listed here will be binarized using sweep_list thresholds as usual. Default is NULL (all variables binarized). Variables listed here are never swept: they are used as they are. To sweep a variable that holds membership scores, leave it out of this argument; it is then binarized at the thresholds you supply (for example 0.3, 0.5, 0.7), like any other numeric variable.
dir.exp	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: dir.exp = c(1, 1, 1) for three conditions all expected to contribute positively.
include	Inclusion rule for minimize. "" (default, QCA compatible) computes the **complex solution** without logical remainders. Use "?" to include logical remainders for **parsimonious** (with dir.exp = NULL) or **intermediate** solutions (with dir.exp specified).
incl.cut	Consistency cutoff for truthTable.
n.cut	Frequency cutoff for truthTable.
pri.cut	PRI cutoff for minimize.
extract_mode	Character. How to handle multiple solutions: "first" (default), "all", or "essential". See qca_extract for details.
return_details	Logical. If TRUE (default), returns both summary and detailed objects for use with generate_report().
Yvar	Deprecated. Use outcome instead.
Xvars	Deprecated. Use conditions 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

Note that return_details changes the *type* of the returned object, not just its contents: with TRUE the summary table is at result$summary, whereas with FALSE the summary table *is* the returned object and result$summary is NULL. Code intended to work under both settings should branch on inherits(result, "data.frame") (or simply always pass return_details = TRUE) rather than assuming result$summary exists.

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 (single): 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

`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_var`	Character. Name of the condition to be swept. Must be one of `conditions`.
`sweep_range`	Numeric vector. Candidate thresholds for `sweep_var`.
`thrY`	Numeric. Threshold for Y (fixed).
`thrX_default`	Numeric. Default threshold for non-swept X variables. Variables listed in `pre_calibrated` do not require this threshold.
`pre_calibrated`	Character vector or `NULL`. Names of condition variables that have been pre-calibrated (e.g., via `QCA::calibrate()`) and should be passed through to `QCA::truthTable()` without binarization. These variables must contain values in the `[0, 1]` range. Variables not listed here will be binarized using `thrX_default` as usual. Default is `NULL` (all variables binarized). Variables listed here are never swept: they are used as they are. To sweep a variable that holds membership scores, leave it out of this argument; it is then binarized at the thresholds you supply (for example 0.3, 0.5, 0.7), like any other numeric variable.
`dir.exp`	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: `dir.exp = c(1, 1, 1)` for three conditions all expected to contribute positively.
`include`	Inclusion rule for `minimize`. `""` (default, QCA compatible) computes the **complex solution** without logical remainders. Use `"?"` to include logical remainders for **parsimonious** (with `dir.exp = NULL`) or **intermediate** solutions (with `dir.exp` specified).
`incl.cut`	Consistency cutoff for `truthTable`.

n.cut	Frequency cutoff for truthTable.
pri.cut	PRI cutoff for minimize.
extract_mode	Character. How to handle multiple solutions: "first" (default), "all", or "essential". See qca_extract for details.
return_details	Logical. If TRUE (default), returns both summary and detailed objects for use with generate_report().
Yvar	Deprecated. Use outcome instead.
Xvars	Deprecated. Use conditions 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

Note that return_details changes the *type* of the returned object, not just its contents: with TRUE the summary table is at result$summary, whereas with FALSE the summary table *is* the returned object and result$summary is NULL. Code intended to work under both settings should branch on inherits(result, "data.frame") (or simply always pass return_details = TRUE) rather than assuming result$summary exists.

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: 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

`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_list_X`	Named list. Each element is a numeric vector of candidate thresholds for the corresponding X. Variables listed in `pre_calibrated` do not need a `sweep_list_X` entry.
`sweep_range_Y`	Numeric vector. Candidate thresholds for Y.
`pre_calibrated`	Character vector or `NULL`. Names of condition variables that have been pre-calibrated (e.g., via `QCA::calibrate()`) and should be passed through to `QCA::truthTable()` without binarization. These variables must contain values in the `[0, 1]` range. Variables not listed here will be binarized using `sweep_list_X` thresholds as usual. Default is `NULL` (all variables binarized). Variables listed here are never swept: they are used as they are. To sweep a variable that holds membership scores, leave it out of this argument; it is then binarized at the thresholds you supply (for example 0.3, 0.5, 0.7), like any other numeric variable.

dir.exp	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: dir.exp = c(1, 1, 1) for three conditions all expected to contribute positively.
include	Inclusion rule for minimize. "" (default, QCA compatible) computes the **complex solution** without logical remainders. Use "?" to include logical remainders for **parsimonious** (with dir.exp = NULL) or **intermediate** solutions (with dir.exp specified).
incl.cut	Consistency cutoff for truthTable.
n.cut	Frequency cutoff for truthTable.
pri.cut	PRI cutoff for minimize.
extract_mode	Character. How to handle multiple solutions: "first" (default), "all", or "essential". See qca_extract for details.
return_details	Logical. If TRUE (default), returns both summary and detailed objects for use with generate_report().
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

Note that return_details changes the *type* of the returned object, not just its contents: with TRUE the summary table is at result$summary, whereas with FALSE the summary table *is* the returned object and result$summary is NULL. Code intended to work under both settings should branch on inherits(result, "data.frame") (or simply always pass return_details = TRUE) rather than assuming result$summary exists.

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` *Extract and format terms from solutions*

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` *Format a QCA solution expression*

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 *Format a single QCA term*

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 *Generate Configuration Chart from QCA Solution*

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

sol	A solution object returned by QCA::minimize(), or a list containing solution information.
symbol_set	Character. One of "unicode", "ascii", or "latex". Default is "unicode".
include_metrics	Logical. Whether to include consistency/coverage metrics in the table. Default is TRUE.
language	Character. "en" for English, "ja" for Japanese. Default is "en".
condition_order	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., rticles)
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

result	A result object from any Sweep function (otSweep, ctSweepS, ctSweepM, or dtSweep).
conditions	Character vector. Condition names for row ordering. If NULL, automatically extracted from expressions.
symbol_set	Character. One of "unicode", "ascii", or "latex". Default is "unicode".

chart_level	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.
language	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_fiss_chart *Generate Fiss-Style Configuration Chart from Sweep Results*

Description

Produces a Markdown-formatted configuration chart following Fiss (2011), using four symbols to distinguish core conditions (conditions of the parsimonious term contained in each configuration) from peripheral conditions (present in the intermediate solution only).

Usage

```
generate_fiss_chart(
  result,
  conditions = NULL,
  symbol_set = c("unicode", "ascii", "latex"),
  language = c("en", "ja")
)
```

Arguments

result	Sweep result augmented by compute_fiss_core.
conditions	Character vector. Condition names (row order). If NULL, extracted from stored settings.
symbol_set	Character. One of "unicode" (default), "ascii", or "latex".
language	Character. "en" (default) or "ja".

Details

Call compute_fiss_core first to augment the sweep result.

Value

Character string: Markdown-formatted Fiss configuration chart.

References

Fiss, P. C. (2011). Building better causal theories: A fuzzy set approach to typologies in organization research. *Academy of Management Journal*, 54(2), 393-420.

See Also

compute_fiss_core

Examples

```
## Not run:
data(sample_data)
res <- otSweep(
  dat = sample_data, outcome = "Y",
  conditions = c("X1", "X2", "X3"),
  sweep_range = 6:8,
  thrX = c(X1 = 7, X2 = 7, X3 = 7),
  include = "?", dir.exp = c(1, 1, 1),
  return_details = TRUE
)
res_fiss <- compute_fiss_core(res, conditions = c("X1", "X2", "X3"))
cat(generate_fiss_chart(res_fiss))
cat(generate_fiss_chart(res_fiss, symbol_set = "latex"))
cat(generate_fiss_chart(res_fiss, language = "ja"))

## 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_fiss_core = FALSE,
  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_fiss_core` Logical. If TRUE and `result$fiss_core` exists (i.e., `compute_fiss_core` has been run), the configuration charts use full Fiss (2011) four-symbol notation distinguishing core (conditions of the parsimonious term contained in each configuration) from peripheral conditions (intermediate only). If FALSE (default) or if `fiss_core` data is absent, the standard two-symbol chart is used.

`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 Fiss core/peripheral chart (requires compute_fiss_core)
res_fiss <- compute_fiss_core(result, conditions = c("X1", "X2", "X3"))
generate_report(res_fiss, "my_report.md", format = "full",
                include_fiss_core = TRUE)

## 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: 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 QCA::calibrate()) and should be passed through to QCA::truthTable() 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). Variables listed here are never swept: they are used as they are. To sweep a variable that holds membership scores, leave it out of this argument; it is then binarized at the thresholds you supply (for example 0.3, 0.5, 0.7), like any other numeric variable.
dir.exp	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: dir.exp = c(1, 1, 1) for three conditions all expected to contribute positively.
include	Inclusion rule for minimize. "" (default, QCA compatible) computes the **complex solution** without logical remainders. Use "?" to include logical remainders for **parsimonious** (with dir.exp = NULL) or **intermediate** solutions (with dir.exp specified).
incl.cut	Consistency cutoff for truthTable.

n.cut	Frequency cutoff for truthTable.
pri.cut	PRI cutoff for minimize.
extract_mode	Character. How to handle multiple solutions: "first" (default), "all", or "essential". See qca_extract for details.
return_details	Logical. If TRUE (default), returns both summary and detailed objects for use with generate_report().
Yvar	Deprecated. Use outcome instead.
Xvars	Deprecated. Use conditions 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

Note that return_details changes the *type* of the returned object, not just its contents: with TRUE the summary table is at result$summary, whereas with FALSE the summary table *is* the returned object and result$summary is NULL. Code intended to work under both settings should branch on inherits(result, "data.frame") (or simply always pass return_details = TRUE) rather than assuming result$summary exists.

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 ThSQCA results*

Description

Displays a concise overview of ThSQCA 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 ThSQCA 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)
```

print_fiss_summary *Print Fiss core/peripheral summary for a single threshold*

Description

Displays which conditions are core and which are peripheral at a given threshold, in a human-readable format.

Usage

```
print_fiss_summary(result, thr_key = NULL, language = c("en", "ja"))
```

Arguments

result	Sweep result augmented by compute_fiss_core.
thr_key	Character or numeric. Threshold key (e.g., "7" or 7).
language	Character. "en" or "ja".

Value

Invisibly returns the classification data frame for thr_key.

Examples

```
## Not run:
res_fiss <- compute_fiss_core(res)
print_fiss_summary(res_fiss, thr_key = "7")

## End(Not run)
```

sample_data *Sample dataset for ThSQCA examples*

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 *Summary method for ThSQCA results*

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 ThSQCA 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, 33

compute_fiss_core, 3, *24*, *26*, *33*
config_chart_from_paths, 5
config_chart_multi_solutions, 7
ctSweepM, *3*, 8
ctSweepS, *3*, *8*, 11

dtSweep, *3*, 15

extract_terms, 18

format_qca_solution, 19
format_qca_solutions, 20
format_qca_term, 20

generate_config_chart, 21
generate_cross_threshold_chart, 22
generate_fiss_chart, *5*, 23
generate_report, 25
generate_solution_note, 27

identify_epi, 28

otSweep, *3*, 28

print.ctSweepM_result
        *(*print.tsqca_result*)*, 32
print.ctSweepS_result
        *(*print.tsqca_result*)*, 32
print.dtSweep_result
        *(*print.tsqca_result*)*, 32
print.otSweep_result
        *(*print.tsqca_result*)*, 32
print.tsqca_result, 32
print_fiss_summary, 33

qca_extract, *9*, *13*, *16*, *30*

sample_data, 33
summary.ctSweepM_result
        *(*summary.tsqca_result*)*, 34
summary.ctSweepS_result
        *(*summary.tsqca_result*)*, 34
summary.dtSweep_result
        *(*summary.tsqca_result*)*, 34
summary.otSweep_result
        *(*summary.tsqca_result*)*, 34
summary.tsqca_result, 34

truthTable, *12*