

# Package ‘ThSQCA’

May 30, 2026

**Type** Package

**Title** Threshold-Sweep QCA

**Version** 2.0.0

**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. Also provides Fiss (2011) <[doi:10.5465/amj.2011.60263120](https://doi.org/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 (present in both parsimonious and intermediate solutions) from peripheral conditions (intermediate only). 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)>. 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>

**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** 2026-05-30 09:00:08 UTC

## Contents

compute_fiss_core . . . . .	2
config_chart_from_paths . . . . .	5
config_chart_multi_solutions . . . . .	7
ctSweepM . . . . .	8
ctSweepS . . . . .	11
dtSweep . . . . .	14
extract_terms . . . . .	18
format_qca_solution . . . . .	18
format_qca_solutions . . . . .	19
format_qca_term . . . . .	20
generate_config_chart . . . . .	20
generate_cross_threshold_chart . . . . .	21
generate_fiss_chart . . . . .	22
generate_report . . . . .	24
generate_solution_note . . . . .	26
identify_epi . . . . .	27
otSweep . . . . .	28
print.tsqca_result . . . . .	31
print_fiss_summary . . . . .	32
sample_data . . . . .	33
summary.tsqca_result . . . . .	33
<b>Index</b>	<b>35</b>

---

compute_fiss_core	<i>Compute Fiss Core/Peripheral Classification for Sweep Results</i>
-------------------	--

---

## Description

Takes an existing threshold-sweep result object (produced by `otSweep`, `ctSweepS`, `ctSweepM`, or `dtSweep`) and augments it with Fiss (2011) core/peripheral classification.

Takes an existing threshold-sweep result object (produced by `otSweep`, `ctSweepS`, `ctSweepM`, or `dtSweep`) and augments it with Fiss (2011) core/peripheral classification.

**Usage**

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

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

**Arguments**

result	A sweep result object with \$details and \$settings slots (e.g., from otSweep(..., 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 vs intermediate)

For each threshold in the result, this function:

1. Retrieves the intermediate solution already stored in result\$details.
2. Re-runs QCA::minimize() on the same truth table with dir.exp = NULL to obtain the parsimonious solution.
3. Compares the two solutions: conditions appearing in both are **core**; conditions appearing only in the intermediate solution are **peripheral**.

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 vs intermediate)

For each threshold in the result, this function:

1. Retrieves the intermediate solution already stored in result\$details.
2. Re-runs QCA::minimize() on the same truth table with dir.exp = NULL to obtain the parsimonious solution.
3. Compares the two solutions: conditions appearing in both are **core**; conditions appearing only in the intermediate solution 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` — parsimonious solution expression
- `interm_expression` — intermediate solution expression
- `classification` — data frame with columns `term_idx`, `term_expr`, `condition`, `status`, `type`

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

- `parsim_expression` — parsimonious solution expression
- `interm_expression` — intermediate solution expression
- `classification` — data frame with columns `term_idx`, `term_expr`, `condition`, `status`, `type`

**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.

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](#)

[generate\\_fiss\\_chart](#)

**Examples**

```
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"))

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"))

```

---

```
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 <b>intermediate solution</b> , 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 <b>complex solution</b> without logical remainders. Use "?" to include logical remainders for <b>parsimonious</b> (with <code>dir.exp = NULL</code> ) or <b>intermediate</b> 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 <a href="#">qca_extract</a> 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 <b>intermediate solution</b> , 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 <b>complex solution</b> without logical remainders. Use "?" to include logical remainders for <b>parsimonious</b> (with <code>dir.exp = NULL</code> ) or <b>intermediate</b> 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 <a href="#">qca_extract</a> 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 <- dtSweepS(
  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 <- dtSweepS(
  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

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 <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 sweep_list_X 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.
dir.exp	Directional expectations for minimize. If NULL (default), no directional expectations are applied. To compute the <b>intermediate solution</b> , 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.
include	Inclusion rule for minimize. "" (default, QCA compatible) computes the <b>complex solution</b> without logical remainders. Use "?" to include logical remainders for <b>parsimonious</b> (with <code>dir.exp = NULL</code> ) or <b>intermediate</b> solutions (with <code>dir.exp</code> 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 <a href="#">qca_extract</a> for details.
return_details	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**

sol	A solution object returned by <code>QCA::minimize()</code> , 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).

---

```
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**

```
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)
```

---

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 (present in both parsimonious and intermediate solutions) from peripheral conditions (present in intermediate solution only).

Produces a Markdown-formatted configuration chart following Fiss (2011), using four symbols to distinguish core conditions (present in both parsimonious and intermediate solutions) from peripheral conditions (present in intermediate solution only).

**Usage**

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

**Arguments**

result	Sweep result augmented by <a href="#">compute_fiss_core</a> .
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.

Call [compute\\_fiss\\_core](#) first to augment the sweep result.

**Value**

Character string: Markdown-formatted Fiss configuration chart.

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.

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](#)

[compute\\_fiss\\_core](#)

**Examples**

```

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"))

```

```

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"))

```

---

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 = file.path(tempdir(), "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 <code>return_details = TRUE</code> .
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 <code>NULL</code> and <code>dat</code> is provided, uses <code>Yvar</code> and <code>Xvars</code> from <code>params</code> .
include_chart	Logical. If <code>TRUE</code> (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 <code>TRUE</code> (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 <code>TRUE</code> and <code>result\$fiss_core</code> exists (i.e., <code>compute_fiss_core</code> has been run), the configuration charts use full Fiss (2011) four-symbol notation distinguishing core (present in both parsimonious and intermediate solutions) from peripheral conditions (intermediate only). If <code>FALSE</code> (default) or if <code>fiss_core</code> data is absent, the standard two-symbol chart is used.
include_raw_output	Logical. If <code>TRUE</code> (default), includes the raw QCA package output ( <code>print(sol)</code> ) for each threshold for verification purposes.

**Value**

Invisibly returns the path to the generated report.

**Examples**

```
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, file.path(tempdir(), "my_report.md"), format = "full",
  dat = sample_data, include_chart = TRUE)

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

# With Fiss-style term-level charts (default, recommended for publications)
generate_report(result, file.path(tempdir(), "my_report.md"), format = "full")

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

---

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**

<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_range</code>	Numeric vector. Candidate thresholds for Y.
<code>thrX</code>	Named numeric vector. Fixed thresholds for X variables. Names must match the conditions that require binarization. Variables listed in <code>pre_calibrated</code> do not need a <code>thrX</code> entry.
<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 <code>[0, 1]</code> range. Variables not listed here will be binarized using <code>thrX</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 <b>intermediate solution</b> , 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 <b>complex solution</b> without logical remainders. Use "?" to include logical remainders for <b>parsimonious</b> (with <code>dir.exp = NULL</code> ) or <b>intermediate</b> 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 <a href="#">qca_extract</a> 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 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.

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, language = c("en", "ja"))
```

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

**Arguments**

result	Sweep result augmented by <code>compute_fiss_core</code> .
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.

Invisibly returns the classification data frame for thr\_key.

---

`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, 2, 23, 25, 32  
config\_chart\_from\_paths, 5  
config\_chart\_multi\_solutions, 7  
ctSweepM, 2, 8  
ctSweepS, 2, 11

dtSweep, 2, 14

extract\_terms, 18

format\_qca\_solution, 18  
format\_qca\_solutions, 19  
format\_qca\_term, 20

generate\_config\_chart, 20  
generate\_cross\_threshold\_chart, 21  
generate\_fiss\_chart, 4, 22  
generate\_report, 24  
generate\_solution\_note, 26

identify\_epi, 27

otSweep, 2, 28

print.ctSweepM\_result  
    (print.tsqca\_result), 31  
print.ctSweepS\_result  
    (print.tsqca\_result), 31  
print.dtSweep\_result  
    (print.tsqca\_result), 31  
print.otSweep\_result  
    (print.tsqca\_result), 31  
print.tsqca\_result, 31  
print\_fiss\_summary, 32

qca\_extract, 9, 12, 15, 29

sample\_data, 33

summary.ctSweepM\_result  
    (summary.tsqca\_result), 33  
summary.ctSweepS\_result  
    (summary.tsqca\_result), 33  
summary.dtSweep\_result  
    (summary.tsqca\_result), 33  
summary.otSweep\_result  
    (summary.tsqca\_result), 33  
summary.tsqca\_result, 33

truthTable, 12