Package 'surveydata'

January 17, 2026

Version 0.2.8

License GPL-2 | GPL-3

Title Tools to Work with Survey Data

LazyData true

LazyLoad true

Copyright Andrie de Vries

Description Data obtained from surveys contains information not only about the survey responses, but also the survey metadata, e.g. the original survey questions and the answer options. The 'surveydata' package makes it easy to keep track of this metadata, and to easily extract columns with specific questions.

URL https://github.com/andrie/surveydata, https://andrie.github.io/surveydata/

BugReports https://github.com/andrie/surveydata/issues

ByteCompile yes

Depends R (>= 4.1.0)

Imports dplyr, rlang, magrittr, purrr, ggplot2, scales, tidyr, DT, assertthat

Suggests testthat, knitr, rmarkdown, withr, covr, rprojroot, spelling

RoxygenNote 7.3.3

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

Config/testthat/edition 3

NeedsCompilation no

Author Andrie de Vries [aut, cre, cph], Evan Odell [ctb]

Maintainer Andrie de Vries <apdevries@gmail.com>

Repository CRAN

Date/Publication 2026-01-17 14:00:02 UTC

Contents

`surveydata-package` *Tools, classes and methods to manipulate survey data.*

Description

Tools, classes and methods to manipulate survey data.

Details

Surveydata objects have been designed to function with SPSS export data, i.e. the result of an SPSS import, `foreign::read.spss()`. This type of data is contained in a data.frame, with information about the questionnaire text in the `variable.labels` attribute. Surveydata objects keep track of the variable labels, by offering methods for renaming, subsetting, etc.

Coercion functions:

- `as.surveydata()`
- `is.surveydata()`
- `as.data.frame.surveydata()`

To access and modify attributes:

- `pattern()`
- `varlabels()`

To subset or merge surveydata objects:

- `merge()`
- `Extract()`
- `cbind.surveydata()`

To extract question text from varlabels:

- `question_text()`
- `question_text_common()`
- `question_text_unique()`

To fix common encoding problems:

- `encToInt()`
- `intToEnc()`
- `fix_common_encoding_problems()`

To clean data:

- `remove_dont_know()` to remove "Don't know" responses
- `remove_all_dont_know()` to remove "Don't know" responses from all questions
- `fix_levels_01()` to fix level formatting of all question with Yes/No type answers

Miscellaneous tools:

- `dropout()` to determine questions where respondents drop out

Author(s)

Andrie de Vries <apdevries@gmail.com>

See Also

Useful links:

- https://github.com/andrie/surveydata
- https://andrie.github.io/surveydata/
- Report bugs at https://github.com/andrie/surveydata/issues

Examples

```
library(surveydata)

# Create surveydata object

sdat <- data.frame(
    id   = 1:4,
    Q1   = c("Yes", "No", "Yes", "Yes"),
    Q4_1 = c(1, 2, 1, 2),
    Q4_2 = c(3, 4, 4, 3),
    Q4_3 = c(5, 5, 6, 6),
    Q10 = factor(c("Male", "Female", "Female", "Male")),
    crossbreak  = c("A", "A", "B", "B"),
    weight      = c(0.9, 1.1, 0.8, 1.2)
)

varlabels(sdat) <- c(
    "RespID",
    "Question 1",
    "Question 4: red", "Question 4: green", "Question 4: blue",
    "Question 10",
    "crossbreak",
    "weight"
  )

sv <- as.surveydata(sdat, renameVarlabels = TRUE)

# Extract specific questions
sv[, "Q1"]
sv[, "Q4"]

# Query attributes
varlabels(sv)
pattern(sv)

# Find unique questions

questions(sv)
which.q(sv, "Q1")
which.q(sv, "Q4")

# Find question text
question_text(sv, "Q1")
question_text(sv, "Q4")
```

```
question_text_common(sv, "Q4")
question_text_unique(sv, "Q4")


# Basic operations on a surveydata object, illustrated with the example dataset membersurvey

class(membersurvey)

questions(membersurvey)

which.q(membersurvey, "Q1")
which.q(membersurvey, "Q3")
which.q(membersurvey, c("Q1", "Q3"))

question_text(membersurvey, "Q3")
question_text_unique(membersurvey, "Q3")
question_text_common(membersurvey, "Q3")

# Extracting columns from a surveydata object

head(membersurvey[, "Q1"])
head(membersurvey["Q1"])
head(membersurvey[, "Q3"])
head(membersurvey[, c("Q1", "Q3")])

# Note that the result is always a surveydata object, even if only one column is extracted

head(membersurvey[, "id"])
str(membersurvey[, "id"])
```

`as.data.frame.surveydata` *Coerces surveydata object to data.frame.*

Description

Coerces surveydata object to data.frame.

Usage

```
## S3 method for class 'surveydata'
as.data.frame(x, ..., rm.pattern = FALSE)
```

Arguments

`x`	Surveydata object to coerce to class data.frame
`...`	ignored
`rm.pattern`	If TRUE removes `pattern()` attributes from x

See Also

surveydata-package

`as.surveydata` *Coercion from and to surveydata.*

Description

Methods for creating `surveydata` objects, testing for class, and coercion from other objects.

Usage

```
as.surveydata(
  x,
  sep = "_",
  exclude = "other",
  ptn = pattern(x),
  defaultPtn = list(sep = sep, exclude = exclude),
  renameVarlabels = FALSE
)

un_surveydata(x)
```

Arguments

`x`	Object to coerce to surveydata
`sep`	Separator between question and sub-question names
`exclude`	Excludes from pattern search
`ptn`	A list with two elements, `sep` and `exclude`. See `pattern()` and `which.q()` for more detail.
`defaultPtn`	The default for `ptn`, if it doesn't exist in the object that is being coerced.
`renameVarlabels`	If TRUE, turns variable.labels attribute into a named vector, using `names(x)` as names.

Details

The function`un_surveydata()` removes the `surveydata` class from the object, leaving intact the other classes, e.g. `data.frame` or `tibble`

See Also

surveydata-package, `is.surveydata()`

Examples

```
library(surveydata)

# Create surveydata object

sdat <- data.frame(
    id   = 1:4,
    Q1   = c("Yes", "No", "Yes", "Yes"),
    Q4_1 = c(1, 2, 1, 2),
    Q4_2 = c(3, 4, 4, 3),
    Q4_3 = c(5, 5, 6, 6),
    Q10 = factor(c("Male", "Female", "Female", "Male")),
    crossbreak  = c("A", "A", "B", "B"),
    weight      = c(0.9, 1.1, 0.8, 1.2)
)

varlabels(sdat) <- c(
    "RespID",
    "Question 1",
    "Question 4: red", "Question 4: green", "Question 4: blue",
    "Question 10",
    "crossbreak",
    "weight"
  )

sv <- as.surveydata(sdat, renameVarlabels = TRUE)

# Extract specific questions
sv[, "Q1"]
sv[, "Q4"]

# Query attributes
varlabels(sv)
pattern(sv)

# Find unique questions

questions(sv)
which.q(sv, "Q1")
which.q(sv, "Q4")

# Find question text
question_text(sv, "Q1")
question_text(sv, "Q4")

question_text_common(sv, "Q4")
question_text_unique(sv, "Q4")


# Basic operations on a surveydata object, illustrated with the example dataset membersurvey

class(membersurvey)
```

```
questions(membersurvey)

which.q(membersurvey, "Q1")
which.q(membersurvey, "Q3")
which.q(membersurvey, c("Q1", "Q3"))

question_text(membersurvey, "Q3")
question_text_unique(membersurvey, "Q3")
question_text_common(membersurvey, "Q3")

# Extracting columns from a surveydata object

head(membersurvey[, "Q1"])
head(membersurvey["Q1"])
head(membersurvey[, "Q3"])
head(membersurvey[, c("Q1", "Q3")])

# Note that the result is always a surveydata object, even if only one column is extracted

head(membersurvey[, "id"])
str(membersurvey[, "id"])
```

`as_opentext_datatable` *Converts free format question text to datatable using the* `DT` *package.*

Description

Converts free format question text to datatable using the `DT` package.

Usage

```
as_opentext_datatable(data, q)
```

Arguments

`data`	surveydata object
`q`	Question

See Also

Other open text functions: `print_opentext()`

Examples

```
as_opentext_datatable(membersurvey, "Q33")
```

cbind.surveydata *Combines surveydata object by columns.*

Description

Combines surveydata object by columns.

Usage

```
## S3 method for class 'surveydata'
cbind(..., deparse.level = 1)
```

Arguments

...	surveydata objects
deparse.level	ignored

dropout *Calculates at which questions respondents drop out.*

Description

The number of respondents for each question is calculated as the length of the vector, after omitting NA values.

Usage

```
dropout(x, summary = TRUE)
```

Arguments

x	surveydata object, list or data.frame
summary	If TRUE, returns a shortened vector that contains only the points where respondents drop out. Otherwise, returns the number of respondents for each question.

Value

Named numeric vector of respondent counts

Examples

```
dropout(membersurvey[-(127:128)])
```

encToInt *Converts a character vector to an integer vector.*

Description

Conversion of character vector to integer vector. The encoding of the character vector can be specified but defaults to the current locale.

Usage

```
encToInt(x, encoding = localeToCharset())
```

Arguments

x	Character vector
encoding	A character string describing the encoding of x. Defaults to the current locale. See also iconvlist()

Value

An integer vector

See Also

iconv()

Other Functions to clean data: fix_common_encoding_problems(), fix_levels_01_spss(), has_dont_know(), intToEnc(), leveltest, remove_all_dont_know(), remove_dont_know()

Examples

```
encToInt("\xfa")
```

fix_common_encoding_problems *Fix common encoding problems when working with web imported data.*

Description

This function tries to resolve typical encoding problems when importing web data on Windows. Typical problems occur with pound and emdash (-), especially when these originated in MS-Word.

Usage

```
fix_common_encoding_problems(x, encoding = localeToCharset())
```

Arguments

x	A character vector
encoding	A character string describing the encoding of x. Defaults to the current locale. See also iconvlist()

See Also

Other Functions to clean data: encToInt(), fix_levels_01_spss(), has_dont_know(), intToEnc(), leveltest, remove_all_dont_know(), remove_dont_know()

---

fix_levels_01_spss *Fix level formatting of all question with Yes/No type answers.*

Description

Fix level formatting of all question with Yes/No type answers.

Usage

```
fix_levels_01_spss(dat)

fix_levels_01_r(dat)

fix_levels_01(dat, origin = c("R", "SPSS"))
```

Arguments

dat	surveydata object
origin	Either R or SPSS

See Also

Other Functions to clean data: encToInt(), fix_common_encoding_problems(), has_dont_know(), intToEnc(), leveltest, remove_all_dont_know(), remove_dont_know()

has_dont_know *Tests whether levels contain "Don't know".*

Description

Returns TRUE if x contains any instances of dk

Usage

```
has_dont_know(x, dk = "Don't Know")
```

Arguments

x	Character vector or factor
dk	Character vector, containing search terms, e.g. `c("Don't know", "Don't Know")`

Value

TRUE or FALSE

See Also

Other Functions to clean data: `encToInt()`, `fix_common_encoding_problems()`, `fix_levels_01_spss()`, `intToEnc()`, `leveltest`, `remove_all_dont_know()`, `remove_dont_know()`

intToEnc *Converts an integer vector to a character vector.*

Description

Conversion of integer vector to character vector. The encoding of the character vector can be specified but defaults to the current locale.

Usage

```
intToEnc(x, encoding = localeToCharset())
```

Arguments

x	Integer vector
encoding	A character string describing the encoding of x. Defaults to the current locale. See also `iconvlist()`

Value

A character vector

See Also

iconv()

Other Functions to clean data: encToInt(), fix_common_encoding_problems(), fix_levels_01_spss(), has_dont_know(), leveltest, remove_all_dont_know(), remove_dont_know()

Examples

```
intToEnc(8212)
```

---

is.surveydata *Tests whether an object is of class surveydata.*

Description

Tests whether an object is of class surveydata.

Usage

```
is.surveydata(x)
```

Arguments

x	Object to check for being of class surveydata

See Also

surveydata-package

---

lapply_names *Applies function only to named elements of a list.*

Description

This is useful to clean only some columns in a list (or data.frame or surveydata object). This is a simple wrapper around lapply() where only the named elements are changed.

Usage

```
lapply_names(x, names, FUN, ...)
```

Arguments

x	list
names	character vector identifying which elements of the list to apply FUN
FUN	function to apply.
...	additional arguments passed to FUN

See Also

Other Tools: question_order()

---

leveltest *Fix level formatting of all question with Yes/No type answers.*

Description

Fix level formatting of all question with Yes/No type answers.

Usage

```
leveltest_spss(x)

leveltest_r(x)
```

Arguments

x	surveydata object

See Also

Other Functions to clean data: encToInt(), fix_common_encoding_problems(), fix_levels_01_spss(), has_dont_know(), intToEnc(), remove_all_dont_know(), remove_dont_know()

---

membersurvey *Data frame with survey data of member satisfaction survey.*

Description

Data frame with survey data of member satisfaction survey.

Usage

```
membersurvey
```

Format

data frame

merge *Merge surveydata objects.*

Description

The base R merge will merge data but not all of the attributes. This function also merges the variable.labels attribute.

Usage

```
## S3 method for class 'surveydata'
merge(x, y, ...)
```

Arguments

x	surveydata object
y	surveydata object
...	Other parameters passed to merge()

print_opentext *Print open text questions.*

Description

Print open text questions.

Usage

```
print_opentext(data, q, cat = TRUE)
```

Arguments

data	data
q	Question number
cat	If TRUE, prints results using cat()

See Also

Other open text functions: as_opentext_datatable()

Examples

```
print_opentext(membersurvey, "Q33")
```

questions *Returns a list of all the unique questions in the surveydata object.*

Description

In many survey systems, sub-questions take the form Q1_a, Q1_b, with the main question and sub-question separated by an underscore. This function conveniently returns all of the main questions in a surveydata() object. It does this by using the pattern() attribute of the surveydata object.

Usage

```
questions(x, ptn = pattern(x))
```

Arguments

x	Object to coerce to surveydata
ptn	A list with two elements, sep and exclude. See pattern() and which.q() for more detail.

Value

numeric vector

See Also

which.q

Other Question functions: question_text(), question_text_common(), question_text_unique(), split_common_unique(), which.q()

Examples

```
# Basic operations on a surveydata object, illustrated with the example dataset membersurvey

class(membersurvey)

questions(membersurvey)

which.q(membersurvey, "Q1")
which.q(membersurvey, "Q3")
which.q(membersurvey, c("Q1", "Q3"))

question_text(membersurvey, "Q3")
question_text_unique(membersurvey, "Q3")
question_text_common(membersurvey, "Q3")

# Extracting columns from a surveydata object

head(membersurvey[, "Q1"])
```

```
head(membersurvey["Q1"])
head(membersurvey[, "Q3"])
head(membersurvey[, c("Q1", "Q3")])

# Note that the result is always a surveydata object, even if only one column is extracted

head(membersurvey[, "id"])
str(membersurvey[, "id"])
```

`question_order` *Changes vector to ordered factor, adding NA levels if applicable.*

Description

Changes vector to ordered factor, adding NA levels if applicable.

Usage

```
question_order(x)
```

Arguments

x	character vector

See Also

Other Tools: `lapply_names()`

`question_text` *Returns question text.*

Description

Given a question id, e.g. "Q4", returns question text for this question. Note that this returns. The functions `question_text_unique()` and `question_text_common()` returns the unique and common components of the question text.

Usage

```
question_text(x, Q)
```

Arguments

x	A surveydata object
Q	The question id, e.g. "Q4". If not supplied, returns the text for all questions.

Value

character vector

See Also

Other Question functions: question_text_common(), question_text_unique(), questions(), split_common_unique(), which.q()

Examples

```
# Basic operations on a surveydata object, illustrated with the example dataset membersurvey

class(membersurvey)

questions(membersurvey)

which.q(membersurvey, "Q1")
which.q(membersurvey, "Q3")
which.q(membersurvey, c("Q1", "Q3"))

question_text(membersurvey, "Q3")
question_text_unique(membersurvey, "Q3")
question_text_common(membersurvey, "Q3")

# Extracting columns from a surveydata object

head(membersurvey[, "Q1"])
head(membersurvey["Q1"])
head(membersurvey[, "Q3"])
head(membersurvey[, c("Q1", "Q3")])

# Note that the result is always a surveydata object, even if only one column is extracted

head(membersurvey[, "id"])
str(membersurvey[, "id"])
```

---

question_text_common *Returns common element of question text.*

Description

Given a question id, e.g. "Q4", finds all sub-questions, e.g. "Q4_1", "Q4_2", etc, and returns the question text that is common to each.

Usage

```
question_text_common(x, Q)
```

Arguments

x	A surveydata object
Q	The question id, e.g. "Q4". If not supplied, returns the text for all questions.

Value

character vector

See Also

Other Question functions: question_text(), question_text_unique(), questions(), split_common_unique(), which.q()

Examples

```
# Basic operations on a surveydata object, illustrated with the example dataset membersurvey

class(membersurvey)

questions(membersurvey)

which.q(membersurvey, "Q1")
which.q(membersurvey, "Q3")
which.q(membersurvey, c("Q1", "Q3"))

question_text(membersurvey, "Q3")
question_text_unique(membersurvey, "Q3")
question_text_common(membersurvey, "Q3")

# Extracting columns from a surveydata object

head(membersurvey[, "Q1"])
head(membersurvey["Q1"])
head(membersurvey[, "Q3"])
head(membersurvey[, c("Q1", "Q3")])

# Note that the result is always a surveydata object, even if only one column is extracted

head(membersurvey[, "id"])
str(membersurvey[, "id"])
```

---

question_text_unique *Returns unique elements of question text.*

Description

Given a question id, e.g. "Q4", finds all sub-questions, e.g. Q4_1, Q4_2, etc, and returns the question text that is unique to each

Usage

```
question_text_unique(x, Q)
```

Arguments

x	A surveydata object
Q	The question id, e.g. "Q4". If not supplied, returns the text for all questions.

Value

character vector

See Also

Other Question functions: question_text(), question_text_common(), questions(), split_common_unique(), which.q()

Examples

```
# Basic operations on a surveydata object, illustrated with the example dataset membersurvey

class(membersurvey)

questions(membersurvey)

which.q(membersurvey, "Q1")
which.q(membersurvey, "Q3")
which.q(membersurvey, c("Q1", "Q3"))

question_text(membersurvey, "Q3")
question_text_unique(membersurvey, "Q3")
question_text_common(membersurvey, "Q3")

# Extracting columns from a surveydata object

head(membersurvey[, "Q1"])
head(membersurvey["Q1"])
head(membersurvey[, "Q3"])
head(membersurvey[, c("Q1", "Q3")])

# Note that the result is always a surveydata object, even if only one column is extracted

head(membersurvey[, "id"])
str(membersurvey[, "id"])
```

remove_all_dont_know *Removes "Do not know" and other similar words from factor levels in data frame.*

Description

Removes "Do not know" and other similar words from factor levels in data frame

Usage

```
remove_all_dont_know(x, dk = NULL, message = TRUE)
```

Arguments

x	List or data frame
dk	Character vector, containing search terms, e.g. `c("Do not know", "DK")`. These terms will be replaced by NA. If NULL, defaults to `c("I don't know", "Don't Know", "Don't know", "Dont know" , "DK")`
message	If TRUE, displays message with the number of instances that were removed.

Value

A data frame

See Also

`hasDK()` and `removeDK()`

Other Functions to clean data: `encToInt()`, `fix_common_encoding_problems()`, `fix_levels_01_spss()`, `has_dont_know()`, `intToEnc()`, `leveltest`, `remove_dont_know()`

remove_dont_know *Removes "Don't know" from levels and replaces with NA.*

Description

Tests the levels of x contain any instances of "Don't know". If so, replaces these levels with NA

Usage

```
remove_dont_know(x, dk = "Don't Know")
```

Arguments

x	Character vector or factor
dk	Character vector, containing search terms, e.g. `c("Don't know", "Don't Know")`

Value

A factor with "Dont know" removed

See Also

Other Functions to clean data: `encToInt()`, `fix_common_encoding_problems()`, `fix_levels_01_spss()`, `has_dont_know()`, `intToEnc()`, `leveltest`, `remove_all_dont_know()`

---

rm.attrs *Removes pattern and variable.labels from attributes list.*

Description

Removes pattern and variable.labels from attributes list.

Usage

```
rm.attrs(x)
```

Arguments

x	Surveydata object

---

rm.pattern *Removes pattern from attributes list.*

Description

Removes pattern from attributes list.

Usage

```
rm.pattern(x)
```

Arguments

x	Surveydata object

split_common_unique *Get common and unique text in question based on regex pattern identification.*

Description

Get common and unique text in question based on regex pattern identification.

Usage

```
split_common_unique(x, ptn = NULL)
```

Arguments

x	A character vector
ptn	A regex() pattern that defines how the string should be split into common and unique elements

See Also

Other Question functions: question_text(), question_text_common(), question_text_unique(), questions(), which.q()

strCommonUnique *Finds the common and unique elements in a character vector.*

Description

Function takes a character string as input and find the common and unique elements. Assumes that the common element is at start of string.

Usage

```
strCommonUnique(string)
```

Arguments

string	Character vector

Value

list of common and unique strings

Examples

```
test <- c("Q_1", "Q_2", "Q_3")
strCommonUnique(test)$common
strCommonUnique(test)$unique
```

survey_plot_question *Plots single and as multi-response questions.*

Description

Plots single and as multi-response questions.

Usage

```
survey_plot_question(data, q)
```

Arguments

data	surveydata object
q	Question

See Also

Other survey plotting functions: survey_plot_satisfaction(), survey_plot_yes_no()

Examples

```
question_text(membersurvey)

survey_plot_question(membersurvey, "Q2")
survey_plot_yes_no(membersurvey, "Q2")
survey_plot_satisfaction(membersurvey, "Q14")
```

survey_plot_satisfaction *Plot satisfaction questions.*

Description

Plot satisfaction questions.

Usage

```
survey_plot_satisfaction(data, q, fun = c("net", "top3", "top2"))
```

Arguments

data	surveydata object
q	Question
fun	Aggregation function, one of net (compute net satisfaction score), top3 (compute top 3 box score) and top2 (compute top 2 box score)

See Also

Other survey plotting functions: survey_plot_question(), survey_plot_yes_no()

Examples

```
question_text(membersurvey)

survey_plot_question(membersurvey, "Q2")
survey_plot_yes_no(membersurvey, "Q2")
survey_plot_satisfaction(membersurvey, "Q14")
```

---

survey_plot_title *Construct plot title from the question text, wrapping at the desired width.*

Description

This creates a plot title using [ggplot2::ggtitle()]. The main title is string wrapped, and the subtitle is the number of observations in the data.

Usage

```
survey_plot_title(data, q, width = 50)
```

Arguments

data	surveydata object
q	Question
width	Passed to strwrap()

---

survey_plot_yes_no *Plot data in yes/no format.*

Description

Plot data in yes/no format.

Usage

```
survey_plot_yes_no(data, q)
```

Arguments

data	surveydata object
q	Question

See Also

Other survey plotting functions: survey_plot_question(), survey_plot_satisfaction()

Examples

```
question_text(membersurvey)

survey_plot_question(membersurvey, "Q2")
survey_plot_yes_no(membersurvey, "Q2")
survey_plot_satisfaction(membersurvey, "Q14")
```

---

which.q *Identifies the columns indices corresponding to a specific question.*

Description

In many survey systems, sub-questions take the form "Q1_a", "Q1_b", with the main question and sub-question separated by an underscore. This function conveniently returns column index of matches found for a question id in a surveydata object. It does this by using the pattern attribute of the surveydata object.

Usage

```
which.q(x, Q, ptn = pattern(x))
```

Arguments

x	Object to coerce to surveydata
Q	Character string with question number, e.g. "Q2"
ptn	A list with two elements, sep and exclude. See pattern() and which.q() for more detail.

See Also

questions() to return all questions matching the pattern()

Other Question functions: question_text(), question_text_common(), question_text_unique(), questions(), split_common_unique()

Examples

```
# Basic operations on a surveydata object, illustrated with the example dataset membersurvey

class(membersurvey)

questions(membersurvey)
```

```
which.q(membersurvey, "Q1")
which.q(membersurvey, "Q3")
which.q(membersurvey, c("Q1", "Q3"))

question_text(membersurvey, "Q3")
question_text_unique(membersurvey, "Q3")
question_text_common(membersurvey, "Q3")

# Extracting columns from a surveydata object

head(membersurvey[, "Q1"])
head(membersurvey["Q1"])
head(membersurvey[, "Q3"])
head(membersurvey[, c("Q1", "Q3")])

# Note that the result is always a surveydata object, even if only one column is extracted

head(membersurvey[, "id"])
str(membersurvey[, "id"])
```

Index