Package 'codebook'

November 23, 2018

Title Automatic Codebooks from Survey Metadata Encoded in Attributes

Description Easily automate the following tasks to describe data frames:
summarise the distributions of scales and items graphically and using descriptive statistics,
compute reliabilities (internal consistencies, retest, multilevel) for psychological scales,
combine this information with metadata (such as item labels and labelled values) that is derived from R attributes.
To do so, the package relies on 'rmarkdown' partials, so you can generate HTML, PDF, and Word documents. Codebooks
are also available as tables (CSV, Excel, etc.).
The metadata and codebooks are also available at your fingertips via RStudio Addins.

Version 0.6.3

Depends R (>= 3.0.1)

URL https://github.com/rubenarslan/codebook

BugReports https://github.com/rubenarslan/codebook/issues

License MIT + file LICENSE

Imports stats, methods, graphics, utils, rmarkdown, ggplot2 (>= 2.0.0), stringr, psych, likert, knitr, rlang, dplyr, tidyr, pander, skimr (>= 1.0.2), DT, future, haven, mice (>= 3.2.0), tibble, purrr, htmltools, labeling, labelled, rio, shiny (>= 0.13), miniUI (>= 0.1.1), rstudioapi (>= 0.5)

Suggests testthat, shinytest, lme4, roxygen2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

VignetteBuilder knitr

NeedsCompilation no

Author Ruben Arslan [aut, cre]

Maintainer Ruben Arslan <ruben.arslan@gmail.com>

Repository CRAN

Date/Publication 2018-08-01 15:40:03 UTC

R topics documented:

aggregate_and_document_scale . . . 2
asis_knit_child . . . 3
bfi . . . 4
codebook . . . 5
codebook_browser . . . 6
codebook_component_scale . . . 6
codebook_component_single_item . . . 7
codebook_items . . . 7
codebook_missingness . . . 8
codebook_survey_overview . . . 8
codebook_table . . . 9
compute_reliabilities . . . 9
detect_missings . . . 10
detect_scales . . . 11
ended . . . 11
expired . . . 12
has_label . . . 12
has_labels . . . 13
knit_print.alpha . . . 13
knit_print.htest . . . 14
knit_print.multilevel . . . 14
label_browser . . . 15
label_browser_static . . . 15
likert_from_items . . . 16
load_data_and_render_codebook . . . 16
md_pattern . . . 17
metadata_jsonld . . . 17
metadata_list . . . 18
modified . . . 18
paste.knit_asis . . . 19
plot_labelled . . . 19
print.knit_asis . . . 20
rescue_attributes . . . 20
reverse_labelled_values . . . 21
summary.labelled . . . 21
summary.labelled_spss . . . 22
zap_attributes . . . 22
zap_label . . . 23
zap_labelled . . . 23

Index **24**

`aggregate_and_document_scale`

Aggregate variables and remember which variables this were

Description

The resulting variables will have the attribute `scale_item_names` containing the basis for aggregation. Its `label` attribute will refer to the common stem of the aggregated variable names (if any), the number of variables, and the aggregation function.

Usage

```
aggregate_and_document_scale(items, fun = rowMeans, stem = NULL)
```

Arguments

`items`	data.frame of the items that should be aggregated
`fun`	aggregation function, defaults to rowMeans with na.rm = FALSE
`stem`	common stem for the variables, specify if it should not be auto-detected as the longest common stem of the variable names

Examples

```
testdf <- data.frame(bfi_neuro_1 = rnorm(20), bfi_neuro_2 = rnorm(20),
                     bfi_neuro_3R = rnorm(20), age = rpois(20, 30))
item_names <- c('bfi_neuro_1', 'bfi_neuro_2', 'bfi_neuro_3R')
testdf$bfi_neuro <- aggregate_and_document_scale(testdf[, item_names])
testdf$bfi_neuro
```

`asis_knit_child` *Knit a child document and output as is (render markup)*

Description

This slightly modifies the `knitr::knit_child()` function to have different defaults.

- the environment defaults to the calling environment.
- the output receives the class `knit_asis`, so that the output will be rendered "as is" by knitr when calling inside a chunk (no need to set `results='asis'` as a chunk option).
- defaults to `quiet = TRUE`

Usage

```
asis_knit_child(input = NULL, text = NULL, ..., quiet = TRUE,
  options = NULL, envir = parent.frame(), use_strings = TRUE)
```

Arguments

`input`	if you specify a file path here, it will be read in before being passed to knitr (to avoid a working directory mess)
`text`	passed to `knitr::knit_child()`
`...`	passed to `knitr::knit_child()`
`quiet`	passed to `knitr::knit_child()`
`options`	defaults to NULL.
`envir`	passed to `knitr::knit_child()`
`use_strings`	whether to read in the child file as a character string (solves working directory problems but harder to debug)

Details

Why default to the calling environment? Typically this function defaults to the global environment. This makes sense if you want to use knit children in the same context as the rest of the document. However, you may also want to use knit children inside functions to e.g. summarise a regression using a set of commands (e.g. plot some diagnostic graphs and a summary for a regression nicely formatted).

Some caveats:

- the function has to return to the top-level. There's no way to `cat()` this from loops or an if-condition without without setting `results='asis'`. You can however concatenate these objects with `paste.knit_asis()`

Examples

```
## Not run:
# an example of a wrapper function that calls asis_knit_child with an argument
# ensures distinct paths for cache and figures, so that these calls can be looped in para
regression_summary <- function(model) {
   hash <- digest::digest(model)
   options <- list(
       fig.path = paste0(knitr::opts_chunk$get("fig.path"), hash, "-"),
       cache.path = paste0(knitr::opts_chunk$get("cache.path"), hash, "-"))
   asis_knit_child("_regression_summary.Rmd", options = options)
}

## End(Not run)
```

---

bfi *Mock BFI data*

Description

a small mock BFI dataset with realistic values, exported from formr. The dataset is self-documenting via its attributes.

Usage

```
bfi
```

Format

A data frame with 28 rows and 29 variables:

`codebook` *Generate rmarkdown codebook*

Description

Pass a data frame to this function to make a codebook for that dataset. If the dataset has metadata (attributes) set on its variables, these will be used to make the codebook more informative. Examples are item, value, and missing labels. Data frames imported via `haven::read_dta()`, `haven::read_sav()`, or from formr.org will have these attributes in the right format. By calling this function inside a knitr code chunk, the codebook will become part of the document you are generating.

Usage

```
codebook(results, reliabilities = NULL, survey_repetition = c("auto",
  "single", "repeated_once", "repeated_many"), survey_overview = TRUE,
  missingness_report = TRUE, metadata_table = TRUE,
  metadata_json = TRUE, indent = "#")
```

Arguments

`results`	a data frame, ideally with attributes set on variables
`reliabilities`	a named list with one entry per scale and one or several printable reliability computations for this scale. if NULL, computed on-the-fly using compute_reliabilities
`survey_repetition`	defaults to "auto" which is to try to determine the level of repetition from the "session" and "created" variables. Other values are: single, repeated_once, repeated_many
`survey_overview`	whether to print an overview of survey entries, durations (depends on presence of columns session, created, modified, ended, expired)
`missingness_report`	whether to print a missingness report. Turn off if this gets too complicated and you need a custom solution (e.g. in case of random missings).
`metadata_table`	whether to print a metadata table/tabular codebook.
`metadata_json`	whether to include machine-readable metadata as JSON-LD (not visible)
`indent`	add # to this to make the headings in the components lower-level. defaults to beginning at h2

Examples

```
# will generate figures in a temporary directory
old_base_dir <- knitr::opts_knit$get("base.dir")
knitr::opts_knit$set(base.dir = tempdir())
on.exit(knitr::opts_knit$set(base.dir = old_base_dir))
data("bfi")
bfi <- bfi[, c("BFIK_open_1", "BFIK_open_1")]
md <- codebook(bfi, survey_repetition = "single", metadata_table = FALSE)
```

codebook_browser *Browse and search codebook*

Description

Usable as an Addin in RStudio. You can select it from a menu at the top, when this package is installed. If you're currently selecting the name of a data frame in your source code, this will be the dataset shown by default. If you don't select text, you can pick a dataset from a dropdown. You can add a keyboard shortcut for this command by following the instructions by RStudio. How about Cmd+Ctrl+C?

Usage

```
codebook_browser(data = NULL, labels_only = FALSE,
  title = "Codebook metadata", viewer = rstudioapi::viewer)
```

Arguments

data	the dataset to display. If left empty will try to use selected text in RStudio or offer a dropdown
labels_only	defaults to false called with TRUE from label_browser()
title	title of the gadget
viewer	defaults to displaying in the RStudio viewer

codebook_component_scale *Codebook component for scales*

Description

Codebook component for scales

Usage

```
codebook_component_scale(scale, scale_name, items, reliabilities,
  indent = "##")
```

Arguments

scale	a scale with attributes set
scale_name	the variable name of this scale
items	a data.frame with the items constituting the scale
reliabilities	a list with one or several results from calls to psych package functions for computing reliability
indent	add # to this to make the headings in the components lower-level. defaults to beginning at h2

Examples

```
# will generate figures in a temporary directory
old_base_dir <- knitr::opts_knit$get("base.dir")
knitr::opts_knit$set(base.dir = tempdir())
on.exit(knitr::opts_knit$set(base.dir = old_base_dir))
data("bfi")
bfi <- bfi[,c("BFIK_open", paste0("BFIK_open_", 1:4))]
codebook_component_scale(bfi[,1], "BFIK_open", bfi[,-1],
   reliabilities = list(BFIK_open = psych::alpha(bfi[,-1])))
```

---

codebook_component_single_item

Codebook component for single items

Description

Codebook component for single items

Usage

```
codebook_component_single_item(item, item_name, indent = "##")
```

Arguments

item	an item with attributes set
item_name	the item name
indent	add # to this to make the headings in the components lower-level. defaults to beginning at h2

Examples

```
# will generate figure in a temporary directory
old_base_dir <- knitr::opts_knit$get("base.dir")
knitr::opts_knit$set(base.dir = tempdir())
on.exit(knitr::opts_knit$set(base.dir = old_base_dir))
data("bfi")
codebook_component_single_item(bfi$BFIK_open_1, "BFIK_open_1")
```

---

codebook_items *Tabular codebook*

Description

Renders a tabular codebook including attributes and data summaries. The table is generated using `DT::datatable()` and can be exported to CSV, Excel, etc.

Usage

```
codebook_items(results, indent = "##")
```

Arguments

results	a data frame, ideally with attributes set on variables
indent	add # to this to make the headings in the components lower-level. defaults to beginning at h2

Examples

```
data("bfi")
## Not run:
# doesn't show interactively, because a html widget needs to be registered
codebook_items(bfi)

## End(Not run)
```

---

codebook_missingness

Codebook missingness

Description

An overview table of missingness patterns generated using `md_pattern()`.

Usage

```
codebook_missingness(results, indent = "##")
```

Arguments

results	a data frame
indent	add # to this to make the headings in the components lower-level. defaults to beginning at h2

Examples

```
data("bfi")
codebook_missingness(bfi)
```

---

codebook_survey_overview

Codebook survey overview

Description

An overview of the number of rows and groups, and of the durations participants needed to respond (if those data are available).

Usage

```
codebook_survey_overview(results, survey_repetition = "single",
  indent = "##")
```

Arguments

results	a data frame which has the following columns: session, created, modified, expired, ended
survey_repetition	defaults to single (other values: repeated_once, repeated_many). controls whether internal consistency, retest reliability or multilevel reliability is computed
indent	add # to this to make the headings in the components lower-level. defaults to beginning at h2

Examples

```
# will generate figures in a figure/ subdirectory
data("bfi")
codebook_survey_overview(bfi)
```

---

codebook_table *Codebook metadata table*

Description

will generate a table combining metadata from variable attributes with data summaries generated using `skimr::skim_to_wide()`

Usage

```
codebook_table(results)
```

Arguments

results	a data frame, ideally with attributes set on variables

Examples

```
data("bfi")
codebook_table(bfi)
```

---

compute_reliabilities *Compute reliabilities*

Description

If you pass the object resulting from a call to formr_results to this function, it will compute reliabilities for each scale. Internally, each reliability computation is passed to a `future::future()`. If you are calculating multilevel reliabilities, it may be worthwhile to parallelise this operation using `future::plan()`. If you don't plan on any complicated parallelisation, you probably do not need to call this function directly, but can rely on it being automatically called during codebook generation. If you do plan to do that, you can pass the results of this operation to the codebook function.

Usage

```
compute_reliabilities(results, survey_repetition = "single")
```

Arguments

results	a formr results table with attributes set on items and scales
survey_repetition	defaults to "single". Can also be "repeated_once" or "repeated_many"

Examples

```
data("bfi", package = "codebook")
reliabilities <- compute_reliabilities(bfi)
```

---

detect_missings *Detect missing values*

Description

SPSS users frequently label their missing values, but don't set them as missing. This function will rectify that for negative values and for the values 99 and 999 (only if they're 5*MAD away from the median). Using different settings, you can also easily tag other missings.

Usage

```
detect_missings(data, only_labelled_missings = TRUE,
  negative_values_are_missing = TRUE, ninety_nine_problems = TRUE,
  learn_from_labels = TRUE, missing = c(), non_missing = c(),
  vars = names(data), use_labelled_spss = FALSE)
```

Arguments

data	the data frame with labelled missings
only_labelled_missings	don't set values to missing if there's no label for them
negative_values_are_missing	by default we label negative values as missing
ninety_nine_problems	SPSS users often store values as 99/999, should we do this for values with 5*MAD of the median
learn_from_labels	if there are labels for missings of the form `[-1] no answer`, set -1 in the data to the corresponding tagged missing
missing	also set these values to missing (or enforce for 99/999 within 5*MAD)
non_missing	don't set these values to missing
vars	only edit these variables
use_labelled_spss	the labelled_spss class has a few drawbacks. Since R can't store missings like -1 and 99, we're replacing them with letters unless this option is enabled. If you prefer to keep your -1 etc, turn this on.

detect_scales *Detect item scales*

Description

Did you create aggregates of items like this `scale <- scale_1 + scale_2R + scale_3R`? If you run this function on a dataset, it will detect these relationships and set the appropriate attributes. Once they are set, the codebook package can perform reliability computations for you.

Usage

```
detect_scales(data, quiet = FALSE)
```

Arguments

`data`	the data frame
`quiet`	defaults to false. Suppresses messages about found items.

Examples

```
bfi <- data.frame(matrix(data = rnorm(500), ncol = 5))
names(bfi) <- c("bfi_e1", "bfi_e2R", "bfi_e3", "bfi_n1", "bfi_n2")
bfi$bfi_e <- rowMeans(bfi[, c("bfi_e1", "bfi_e2R", "bfi_e3")])
bfi <- detect_scales(bfi)
bfi$bfi_e
```

ended *How many surveys were ended?*

Description

Just a simple to check how many times a survey (e.g. diary) was finished. It defaults to checking the "ended" variable for this.

Usage

```
ended(survey, variable = "ended")
```

Arguments

`survey`	which survey are you asking about?
`variable`	which variable should be filled out, defaults to "ended"

Examples

```
survey <- data.frame(ended = c("2016-05-28 10:11:00", NA, "2016-05-30 11:18:28"))
ended(survey = survey)
```

expired *How many surveys were expired?*

Description

Just a simple to check how many times a survey (e.g. diary) has expired (i.e. user missed it). It defaults to checking the "expired" variable for this.

Usage

```
expired(survey, variable = "expired")
```

Arguments

survey	which survey are you asking about?
variable	which variable should be filled out, defaults to "expired"

Examples

```
survey <- data.frame(expired = c(NA, "2016-05-29 10:11:00", NA))
expired(survey = survey)
```

has_label *Has label*

Description

Has label

Usage

```
has_label(x)
```

Arguments

x	a vector

Examples

```
example("labelled", "haven")
has_label(x)
```

has_labels *Has labels*

Description

Has labels

Usage

```
has_labels(x)
```

Arguments

x	a vector

Examples

```
example("labelled", "haven")
has_labels(x)
```

knit_print.alpha *Pretty-print a Cronbach's alpha object*

Description

Turn a `psych::alpha()` object into HTML tables.

Usage

```
knit_print.alpha(x, indent = "#####", ...)
```

Arguments

x	a psych alpha object
indent	add # to this to make the headings in the components lower-level. defaults to beginning at h5
...	ignored

Examples

```
example("alpha", "psych")
knitr::knit_print(a4)
```

knit_print.htest *Print a stats::cor.test() object for knitr*

Description

Just prints the normal output of stats::cor.test().

Usage

```
knit_print.htest(x, indent = "#####", ...)
```

Arguments

x	a psych alpha object
indent	add # to this to make the headings in the components lower-level. defaults to beginning at h5
...	ignored

Examples

```
knitr::knit_print(cor.test(rnorm(100), rnorm(100)))
```

knit_print.multilevel *Print a psych::multilevel.reliability() object for knitr*

Description

Just prints the normal output of psych::multilevel.reliability().

Usage

```
knit_print.multilevel(x, indent = "#####", ...)
```

Arguments

x	a psych alpha object
indent	add # to this to make the headings in the components lower-level. defaults to beginning at h5
...	ignored

Examples

```
example("mlr", "psych")
knitr::knit_print(mg)
```

label_browser *Browse and search variable and value labels*

Description

Same as the `codebook_browser()`, but doesn't show data summaries and additional attributes.

Usage

```
label_browser(data = NULL, viewer = rstudioapi::viewer)
```

Arguments

`data`	the dataset to display. If left empty will try to use selected text in RStudio or offer a dropdown
`viewer`	defaults to displaying in the RStudio viewer

label_browser_static *Browse and search variable and value labels*

Description

Same as the `codebook_browser()`, but doesn't show data summaries and additional attributes. This yields a static table, so you can continue to edit code while viewing the labels, but you cannot switch the dataset via a dropdown menu.

Usage

```
label_browser_static(data = NULL, viewer = rstudioapi::viewer)
```

Arguments

`data`	data frame. if left empty, will use the text you currently select in RStudio as the label or the first data frame in your environment
`viewer`	where to show. defaults to viewer tab

Examples

```
label_browser_static(bfi)
```

likert_from_items *Derive a likert object from items*

Description

Pass a data.frame containing several items composing one scale, get a `likert::likert()` object, which you can plot. Intelligently makes use of labels and value labels if present.

Usage

```
likert_from_items(items)
```

Arguments

items	a data frame of items composing one scale

Examples

```
data("bfi", package = "codebook")
open_items <- paste0("BFIK_open_",1:4)
graphics::plot(likert_from_items(bfi[, open_items]))
```

load_data_and_render_codebook *Render codebook based on file*

Description

Submit a data file and an rmarkdown template as a file to generate a codebook. Used chiefly in the webapp.

Usage

```
load_data_and_render_codebook(file, text, remove_file = FALSE, ...)
```

Arguments

file	path to a file to make codebook from (sav, rds, dta, por, xpt, csv, csv2, tsv, etc.)
text	codebook template
remove_file	whether to remove file after rendering
...	all other arguments passed to `rmarkdown::render()`

md_pattern *Missing data patterns*

Description

Generate missingness patterns using `mice::md.pattern()`, with options to reduce the complexity of the output.

Usage

```
md_pattern(data, only_vars_with_missings = TRUE, min_freq = 0.01)
```

Arguments

`data`	the dataset
`only_vars_with_missings`	defaults to TRUE, omitting variables that have no missings
`min_freq`	minimum number of rows to have this missingness pattern

Examples

```
data("nhanes", package = "mice")
md_pattern(nhanes)
md_pattern(nhanes, only_vars_with_missings = FALSE, min_freq = 0.2)
```

metadata_jsonld *Metadata as JSON-LD*

Description

Echo a list of a metadata, generated using `metadata_list()` as JSON-LD in a script tag.

Usage

```
metadata_jsonld(results)
```

Arguments

`results`	a data frame, ideally with attributes set on variables

Examples

```
data("bfi")
metadata_jsonld(bfi)
```

metadata_list *Metadata from dataframe*

Description

Returns a list containing variable metadata (attributes) and data summaries.

Usage

```
metadata_list(results)
```

Arguments

results	a data frame, ideally with attributes set on variables

Examples

```
data("bfi")
md_list <- metadata_list(bfi)
md_list$variableMeasured[[20]]
```

modified *How many surveys were modified?*

Description

Just a simple to check how many times a survey (e.g. diary) has expired (i.e. user missed it). It defaults to checking the "expired" variable for this.

Usage

```
modified(survey, variable = "modified")
```

Arguments

survey	which survey are you asking about?
variable	which variable should be filled out, defaults to "modified"

Examples

```
survey <- data.frame(modified = c(NA, "2016-05-29 10:11:00", NA))
modified(survey = survey)
```

paste.knit_asis *Paste and output as is (render markup)*

Description

Helper function for `knit_asis` objects, useful when e.g. `asis_knit_child()` was used in a loop.

Usage

```
paste.knit_asis(..., sep = "\n\n\n", collapse = "\n\n\n")
```

Arguments

`...`	passed to `base::paste()`
`sep`	defaults to two empty lines, passed to `base::paste()`
`collapse`	defaults to two empty lines, passed to `base::paste()`

Details

Works like `base::paste()` with both the sep and the collapse argument set to two empty lines

Examples

```
paste.knit_asis("# Headline 1", "## Headline 2")
```

plot_labelled *Plot labelled vector*

Description

Plot a labelled vector, making use of the variable name, label and value labels to make the plot more readable. This function also works for other vectors, but provides little benefit.

Usage

```
plot_labelled(item, item_name = NULL, wrap_at = 50,
  go_vertical = FALSE)
```

Arguments

`item`	a vector
`item_name`	item name, defaults to name of first argument
`wrap_at`	the subtitle (the label) will be wrapped at this number of characters
`go_vertical`	defaults to FALSE. Whether to show choices on the Y axis instead.

Examples

```
data("bfi", package = "codebook")
plot_labelled(bfi$BFIK_open_1)
```

print.knit_asis *Print new lines in* knit_asis *outputs*

Description

Print new lines in knit_asis outputs

Usage

```
## S3 method for class 'knit_asis'
print(x, ...)
```

Arguments

x	the knit_asis object
...	ignored

rescue_attributes *Rescue lost attributes*

Description

You can use this function if some of your items have lost their attributes during wrangling Variables have to have the same name (Duh) and no attributes should be overwritten. But use with care. Similar to labelled::copy_labels().

Usage

```
rescue_attributes(df_no_attributes, df_with_attributes)
```

Arguments

df_no_attributes	the data frame with missing attributes
df_with_attributes	the data frame from which you want to restore attributes

reverse_labelled_values

Reverse labelled values reverse the underlying valus for a numeric `haven::labelled()` *vector while keeping the labels correct*

Description

Reverse labelled values reverse the underlying valus for a numeric `haven::labelled()` vector while keeping the labels correct

Usage

```
reverse_labelled_values(x)
```

Arguments

`x`	a labelled vector

Value

return the labelled vector with the underlying values having been reversed

Examples

```
x <- haven::labelled(rep(1:3, each = 3), c(Bad = 1, Good = 5))
x
reverse_labelled_values(x)
```

summary.labelled *Summary function for labelled vector*

Description

Summary function for labelled vector

Usage

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

Arguments

`object`	a labelled vector
`...`	passed to summary.factor

Examples

```
example("labelled", "haven")
summary(x)
```

summary.labelled_spss *Summary function for labelled_spss vector*

Description

Summary function for labelled_spss vector

Usage

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

Arguments

object	a labelled_spss vector
...	passed to summary.factor

Examples

```
example("labelled", "haven")
summary(x)
```

zap_attributes *Zap attributes*

Description

Modelled on haven::zap_labels(), but more encompassing. By default removes the following attributes: format.spss, format.sas, format.stata, label, labels, na_values, na_range, display_width

Usage

```
zap_attributes(x, attributes = c("format.spss", "format.sas",
  "format.stata", "label", "labels", "na_values", "na_range",
  "display_width"))
```

Arguments

x	the data frame or variable
attributes	character vector of attributes to zap. NULL if everything (including factor levels etc) should be zapped

Examples

```
bfi <- data.frame(matrix(data = rnorm(300), ncol = 3))
names(bfi) <- c("bfi_e1", "bfi_e2R", "bfi_e3")
attributes(bfi$bfi_e1)$label <- "I am outgoing."
attributes(bfi$bfi_e2R)$label <- "I prefer books to people."
attributes(bfi$bfi_e3)$label <- "I love to party."
bfi$bfi_e <- rowMeans(bfi[, c("bfi_e1", "bfi_e2R", "bfi_e3")])
bfi <- detect_scales(bfi, quiet = TRUE) # create attributes
str(zap_attributes(bfi, "label"))
zap_attributes(bfi$bfi_e)
```

---

zap_label *Zap variable label*

Description

Modelled on haven::zap_labels(), zaps variable labels (not value labels).

Usage

```
zap_label(x)
```

Arguments

x	the data frame or variable

Examples

```
x <- haven::labelled(rep(1:5, each = 1), c(Bad = 1, Good = 5))
zap_label(x)
```

---

zap_labelled *Zap labelled class*

Description

Modelled on haven::zap_labels(), zaps labelled class (not other attributes).

Usage

```
zap_labelled(x)
```

Arguments

x	the data frame or variable

Index

∗Topic **datasets**
bfi, 4

aggregate_and_document_scale, 2
asis_knit_child, 3
asis_knit_child(), *19*

base::paste(), *19*
bfi, 4

cat(), *4*
codebook, 5
codebook_browser, 6
codebook_browser(), *15*
codebook_component_scale, 6
codebook_component_single_item, 7
codebook_items, 7
codebook_missingness, 8
codebook_survey_overview, 8
codebook_table, 9
compute_reliabilities, 9

detect_missings, 10
detect_scales, 11
DT::datatable(), *7*

ended, 11
expired, 12

future::future(), *9*
future::plan(), *9*

has_label, 12
has_labels, 13
haven::labelled(), *21*
haven::read_dta(), *5*
haven::read_sav(), *5*
haven::zap_labels(), *22, 23*

knit_print.alpha, 13
knit_print.htest, 14
knit_print.multilevel, 14
knitr::knit_child(), *3*

label_browser, 15
label_browser(), *6*
label_browser_static, 15
labelled::copy_labels(), *20*
likert::likert(), *16*
likert_from_items, 16
load_data_and_render_codebook, 16

md_pattern, 17
md_pattern(), *8*
metadata_jsonld, 17
metadata_list, 18
metadata_list(), *17*
mice::md.pattern(), *17*
modified, 18

paste.knit_asis, 19
paste.knit_asis(), *4*
plot_labelled, 19
print.knit_asis, 20
psych::alpha(), *13*
psych::multilevel.reliability(), *14*

rescue_attributes, 20
reverse_labelled_values, 21
rmarkdown::render(), *16*

skimr::skim_to_wide(), *9*
stats::cor.test(), *14*
summary.labelled, 21
summary.labelled_spss, 22

zap_attributes, 22
zap_label, 23
zap_labelled, 23