# Package ‘codebookr’

July 22, 2025

**Title** Create Codebooks from Data Frames

**Version** 0.1.8

**Maintainer** Brad Cannell <brad.cannell@gmail.com>

**Description** Quickly and easily create codebooks (i.e. data dictionaries) directly from a data frame.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.2

**URL** https://github.com/brad-cannell/codebookr,
https://brad-cannell.github.io/codebookr/

**BugReports** https://github.com/brad-cannell/codebookr/issues

**Depends** R (>= 2.10)

**LazyData** true

**Suggests** hms, knitr, rmarkdown, testthat

**Imports** haven (>= 2.5.0), flextable, dplyr, officer, purrr, rlang,
stringr, tibble, tidyr

**NeedsCompilation** no

**Author** Brad Cannell [aut, cre, cph]

**Repository** CRAN

**Date/Publication** 2024-02-19 08:20:08 UTC

## Contents

## `cb_add_col_attributes` *Add Attributes to Columns*

### Description

Add arbitrary attributes to columns (e.g., description, source, column type). These attributes can later be accessed to fill in the column attributes table.

### Usage

```
cb_add_col_attributes(df, .x, ...)
```

### Arguments

| | |
|---|---|
| `df` | Data frame of interest |
| `.x` | Column of interest in df |
| `...` | Arbitrary list of attributes (i.e., attribute = "value") |

### Details

Typically, though not necessarily, the first step in creating your codebook will be to add column attributes to your data. The `cb_add_col_attributes()` function is a convenience function that allows you to add arbitrary attributes to the columns of the data frame. These attributes can later be accessed to fill in the column attributes table of the codebook document. Column attributes *can* serve a similar function to variable labels in SAS or Stata; however, you can assign many different attributes to a column and they can contain any kind of information you want.

Although the `cb_add_col_attributes()` function will allow you to add any attributes you want, there are currently **only five** special attributes that the `codebook()` function will recognize and add to the column attributes table of the codebook document. They are:

**description:** Although you may add any text you desire to the `description` attribute, it is intended to be used describe the question/process that generated the data contained in the column. Many statistical software packages refer to this as a variable label. If the data was imported from SAS, Stata, or SPSS with variable labels using the `haven` package, `codebook` will automatically recognize them. There is no need to manually create them. However, you may overwrite the imported variable label for any column by adding a `description` attribute.

**source:** Although you may add any text you desire to the `source` attribute, it is intended to be used describe where the data contained in the column originally came from. For example, if the current data frame was created by merging multiple data sets together, you may want to use the source attribute to identify the data set it originates from. As another example, if the current data frame contains longitudinal data, you may want to use the source attribute to identify the wave(s) in which data for this column was collected.

**col_type:** The `col_type` attribute is intended to provide additional information above and beyond the `Data type` (i.e., column class) about the values in the column. For example, you may have a column of 0's and 1's, which will have a *numeric* data type. However, you may want to inform data users that this is really a dummy variable where the 0's and 1's represent discrete categories (No and Yes). Another way to think about it is that the `Data type` attribute is how *R* understands the column and the `Column type` attribute is how *humans* should understand the column. Currently accepted values are:

- `Numeric`
- `Categorical`
- `Time`

Perhaps even more importantly, setting the `col_type` attribute helps R determine which descriptive statistics to calculate for the bottom half of the column attributes table. Inside of the `codebook()` function, the `cb_add_summary_stats()` function will attempt to figure out whether the column is:

- numeric
- categorical - many categories (e.g. participant id)
- categorical - few categories (e.g. sex)
- time - including dates

Again, this matters because the table of summary stats shown in the codebook document depends on the value `cb_add_summary_stats()` chooses. However, the user can directly tell `cb_add_summary_stats()` which summary stats to calculate by providing a `col_type` attribute to a column with one of the following values: `Numeric`, `Categorical`, or `Time`.

**value_labels:** Although you may pass any named vector you desire to the `value_labels` attribute, it is intended to inform your data users about how to correctly interpret numerically coded categorical variables. For example, you may have a column of 0's and 1's that represent discrete categories (i.e., "No" and "Yes") instead of numerical quantities. In many other software packages (e.g., SAS, Stata, and SPSS), you can layer "No" and "Yes" labels on top of the 0's and 1's to improve the readability of your analysis output. These are commonly referred to as *value labels*. The R programming language does not really have value labels in the same way that other popular statistical software applications do. R users can (and typically should) coerce numerically coded categorical variables into factors; however, coercing a numeric vector to a factor is not the same as adding value labels to a numeric vector because the underlying numeric values can change in the process of creating the factor. For this, and other reasons, many R programmers choose to create a *new* factor version of a numerically encoded variable as opposed to overwriting/transforming the numerically encoded variable. In those cases, you may want to inform your data users about how to correctly interpret numerically coded categorical variables. Adding value labels to your codebook is one way of doing so.

- Add value labels to columns as a named vector to the `value_labels` attribute. For example, `value_labels` = c("No" = 0, "Yes" = 1).
- If the data was imported from SAS, Stata, or SPSS with value labels using the `haven` package, `codebook` will automatically recognize them. There is no need to manually create them. However, you may overwrite the imported value labels for any column by adding a `value_labels` attribute as shown in the example below.

**skip_pattern:** Although you may add any text you desire to the `skip_pattern` attribute, it is intended to be used describe skip patterns in the data collection tools that impact which study participants were exposed to each study item. For example, If a question in your data was only asked of participants who were enrolled in the study for at least 10 days, then you may want to add a note like "Not asked if days < 10" to the skip pattern section of the column attributes table.

## Value

Returns the same data frame (or tibble) passed to the df argument with column attributes added.

## Examples

```
library(dplyr, warn.conflicts = FALSE)
library(codebookr)
data(study)

study <- study %>%
  cb_add_col_attributes(
    .x = likert,
    description = "An example Likert scale item",
    source = "Exposure questionnaire",
    col_type = "categorical",
    value_labels = c(
      "Very dissatisfied" = 1,
      "Somewhat dissatisfied" = 2,
      "Neither satisfied nor dissatisfied" = 3,
      "Somewhat satisfied" = 4,
      "Very satisfied" = 5
    ),
    skip_pattern = "Not asked if days < 10"
  )
```

---

codebook *Automate creation of a data codebook*

---

## Description

The codebook function assists with the creation of a codebook for a given data frame.

## Usage

```
codebook(
  df,
  title = NA,
  subtitle = NA,
  description = NA,
  keep_blank_attributes = FALSE,
  no_summary_stats = NULL
)
```

## Arguments

| | |
|---|---|
| df | The data frame the codebook will describe |
| title | An optional title that will appear at the top of the Word codebook document |
| subtitle | An optional subtitle that will appear at the top of the Word codebook document |

| | |
|---|---|
| `description` | An optional text description of the dataset that will appear on the first page of the Word codebook document |
| `keep_blank_attributes` | TRUE or FALSE. By default, the column attributes table will omit the Column description, Source information, Column type, value labels, and skip pattern rows from the column attributes table in the codebook document if those attributes haven't been set. In other words, it won't show blank rows for those attributes. Passing `TRUE` to the keep_blank_attributes argument will cause the opposite to happen. The column attributes table will include a Column description, Source information, Column type, and value labels row for every column in the data frame - even if they don't have those attributes set. |
| `no_summary_stats` | A character vector of column names. The summary statistics will not be added to column attributes table for any column passed to this argument. This can be useful when a column contains values that are sensitive or may be used to identify individual people (e.g., names, addresses, etc.) and the individual values for that column should not appear in the codebook. |

## Details

Codebook expects that `df` is a data frame that you have read into memory from a saved data file. Please provide the path to the saved data file. This function gets selected attributes about file saved at `path` and stores those attributes in a data frame, which is later turned into a flextable and added to the codebook document.

Typically, though not necessarily, the first step in creating your codebook will be to add column attributes to your data. The `cb_add_col_attributes()` function is a convenience function that allows you to add arbitrary attributes to the columns of the data frame. These attributes can later be accessed to fill in the column attributes table of the codebook document. Column attributes *can* serve a similar function to variable labels in SAS or Stata; however, you can assign many different attributes to a column and they can contain any kind of information you want. For details see cb_add_col_attributes

## Value

An rdocx object that can be printed to a Word document

## Examples

```
## Not run:
study_codebook <- codebook(
  df = study,
  title = "My Example Study",
  subtitle = "A Subtitle for My Example Study Codebook",
  description = "Brief (or long) description of the data."
)

# Create the Word codebook document
print(study_codebook, path = "example_codebook.docx")

## End(Not run)
```

`study` *Simulated study data.*

## Description

This is the code to create the study data - a simulated dataset that can be used to demonstrate how to use the codebook package.

## Usage

```
study
```

## Format

A data frame with 20 rows and 10 variables:

**id** Participant's study identification number

**address** Participant's home address

**sex** Biological sex of the participant assigned at birth, female/male

**date** Participant's date of enrollment

**time** Participant's time of enrollment

**date_time** Participant's date and time of enrollment

**days** Total number of days the participant was enrolled in the study

**height** Participant's height in inches at date of enrollment

**likert** An example Likert scale item, 1-5

**outcome** Participant experienced the outcome of interest, TRUE or FALSE

# Index