

# Labelled data with labelled :: CHEAT SHEET



The **labelled** package provides a set of functions and methods to handle and to manipulate labelled data, as imported with **haven** package.

## Basics

Labelled data is a common data structure in other statistical environment such as Stata, SAS or SPSS.


It consists of a set of additional attributes for numeric and character vectors (including columns of a data frame).

There are 3 types of attributes:

1. **Variable labels** (a short description of a variable)
2. **Value labels** (labels associated to specific values)
3. **Missing values:**
  - User-defined missing values (SPSS style)
  - Tagged NA (Stata and SAS style)

## Variable labels


### MANIPULATING A VECTOR

 **var\_label(x)** or **var\_label(df\$v1)**  
Get the variable label associated to a vector x

**var\_label(x) <- "variable description"**  
Add/modify a variable label to x

**var\_label(x) <- NULL**  
Remove the variable label associated to x

### MANIPULATING A DATA.FRAME

 **var\_label(df)**  
List all variable labels associated with columns of df

**var\_label(df) <- list(v1 = "variable 1", v2 = "variable 2")**  
Update variable labels of some columns of df

**df %>% set\_variable\_labels(v1 = "variable 1", v2 = "variable 2", v3 = NULL)**  
Update variable labels using dplyr syntax

## Searching variable / Dictionary

**df %>% look\_for()** or **df %>% look\_for(details = "full")**  
Generate a data dictionary


**df %>% look\_for("s")**  
Search variables containing "s" in their name or label

**df %>% look\_for\_and\_select("s")**  
Search variables and apply dplyr::select()

## Value labels

When value labels are attached to a numeric or character vector, the vector's class becomes **haven\_labelled**. A major difference with a factor is that values of the vector are not changed and it is not mandatory to attach a label to each value.

### MANIPULATING A VECTOR

 **val\_label(x, value)** or **val\_label(df\$v1, value)**  
Get the label attached to a specific value of a vector

**val\_label(x, value) <- "label"**  
Set/Update the label attached to a specific value

**val\_label(x, value) <- NULL**  
Remove the label attached to a specific value

**val\_labels(x)**  
Get all value labels attached to a vector

**val\_labels(x) <- c(no = 0, yes = 1, maybe = 9)**  
Set/Update all value labels attached to a vector


**val\_labels(x) <- NULL**  
Remove all value labels attached to a vector

**labelled(c("F", "F", "M"), c(Female = "F", Male = "M"))**  
Create a labelled vector

**sort\_val\_labels(x, according\_to = "values")**  
Sort value labels according to values (or labels)

**drop\_unused\_value\_labels(x)**  
Remove value labels not observed in the data

### MANIPULATING A DATA.FRAME

 **df %>% set\_value\_labels(v1 = c(Yes = 1, No = 2), v2 = c(Male = "M", Female = "F"))**  
Define value labels of several variables

**df %>% add\_value\_labels(v1 = c(Unknown = 9))**  
Add specific value labels to a variable (other already defined value labels remains unchanged)

**df %>% remove\_value\_labels(v1 = 9)**  
Remove specific value labels to a variable

**df %>% set\_value\_labels(v1 = NULL)**  
Remove all value labels attached to a variable

**df %>% drop\_unused\_value\_labels()**  
Remove value labels not observed in the data

## Missing values

### USER-DEFINED MISSING VALUES (SPSS STYLE)

Used to indicate that some values should be considered as missing. However, they will not be treated as NA as long as they are not converted to proper NA.

When missing values are attached to a numeric or character vector, the vector's class becomes **haven\_labelled\_spss**.

When importing a SPSS file, use the option `user_na = TRUE` to keep defined missing values (otherwise, they will be converted to NA).

### na\_values(x)

Get individual missing values attached to a vector

**na\_values(x) <- c(8, 9, 10)**  
**df %>% set\_na\_values(v1 = c(8, 9, 10))**

Set/Update individual missing values (NULL to remove)

### na\_range(x)

Get a range of missing values attached to a vector

**na\_range(x) <- c(8, 10)**  
**df %>% set\_na\_range(v1 = c(8, 10))**  
Set/Update a range of missing values (NULL to remove)

**user\_na\_to\_na(x)** or **df %>% user\_na\_to\_na()**  
Convert user-defined missing values to NA

### is.na(x)

TRUE if NA or if a user-defined missing value

### TAGGED NAs (STATA & SAS STYLE)

“Tagged” missing values work exactly like regular R missing values except that they store one additional byte of information: a tag, which is usually a letter ("a" to "z").

**x <- c(1:5, tagged\_na("a"), tagged\_na("z"), NA)**  
**tagged\_na("a")** generates a NA with a tag

### is.na(x)

Tagged NAs work identically to regular NAs

### is\_tagged\_na(x)

Test if it is a tagged NA

### na\_tag(x)

Display the tags associated to tagged NAs

### format\_tagged\_na(x)

Convert x to a character vector showing the tagged NAs



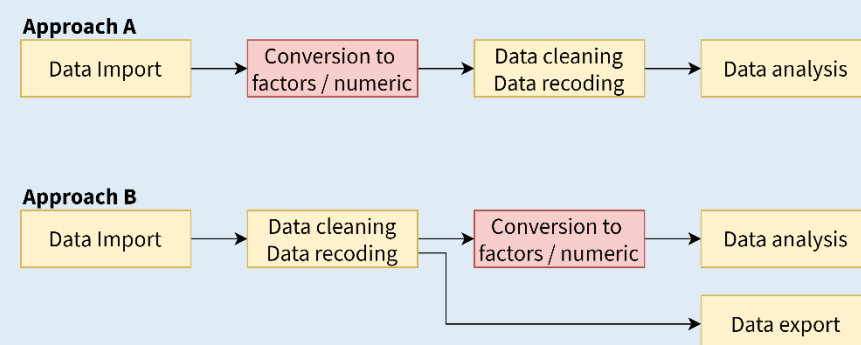
## What is labelled data?

`haven_labelled` and `haven_labelled_spss` classes introduced in **haven** package allow to add metadata (variable labels, value labels and SPSS-style missing values) to vectors / data frame columns and to properly import these metadata from SAS, Stata or SPSS.

Functions and methods provided by **labelled** package are designed for easy manipulation of such labelled data.

It should be noted that **value labels** doesn't imply that your vectors should be considered as categorical or continuous. Therefore, value labels are not intended to be used for data analysis. For example, before performing modeling or plotting, you should convert vectors with value labels into factors or into classic numeric/character vectors.

Two main approaches could be considered:



In **approach A**, labelled vectors are converted into factors or into numeric/character vectors just after data import, using `unlabelled()`, `to_factor()` or `unclass()`. Then, data cleaning, data recoding and analysis are performed using classic **R** vector types.

In **approach B**, labelled vectors are kept for data cleaning and recoding, allowing to preserve original coding, in particular if data should be reexported after that step. Functions provided by **labelled** will be useful for managing value labels.

However, as in approach A, labelled vectors will have to be converted into classic factors or numeric vectors before data analysis as this is the way categorical and continuous variables should be coded for analysis.

## Conversion

### OF LABELLED VECTORS

If all value labels and user-defined missing values are removed from a labelled vector, the `haven_labelled` class will be removed and the vector will be transformed into a basic numeric or character vector. **Values of the vector will remain unchanged.**

#### `remove_val_labels(x)`

Remove value labels attached to a vector.

#### `remove_user_na(x)`

Remove user-defined (`na_values` and `na_range`) from a vector

#### `unclass(x)`

Remove the `haven_labelled` class. Therefore, the vector will be considered as a classical numeric or character vector. Value labels and user-defined missing values will still be visible as attributes attached to the vector.

When converting a labelled vector into a factor or a character vector of the value labels, be aware that **original values of the vector will be converted.**

#### `to_character(x)`

Convert into a character vector replacing values by their corresponding value label

#### `to_factor(x)`

Convert into a factor replacing values by their corresponding value label

#### `to_factor(x, levels = "prefixed")`

Value labels will be prefixed with their original value

#### `to_factor(x, strict = TRUE)`

Convert into a factor only if all observed values have a value label

#### `df %>% to_factor()`

Convert all labelled vectors and only labelled vectors into factors

#### `df %>% to_factor(labelled_only = FALSE)`

Convert all columns (including non labelled vectors) into factors

#### `unlabelled(x)`

#### `df %>% unlabelled()`

Labelled vectors will be converted into factors only if all observed values have a value label. Otherwise, they will be unclassified. Similar to `df %>% to_factor(labelled_only = T, strict = T, unclass = T)`

#### `to_factor(x, drop_unused_labels = TRUE)`

#### `df %>% unlabelled(drop_unused_labels = TRUE)`

Unused value labels will be dropped before conversion into factors

### INTO LABELLED VECTORS

#### `to_labelled(f)`

Convert a factor into a numeric labelled vector. *Note that `to_labelled(to_factor(x))` and `x` will not be identical (original coding will be lost).*

#### `to_labelled(df)`

If `df` was imported with the **foreign** package or if it is a data set created with **memisc** package, meta data (variable labels, value labels and user-defined missing values) will be converted into **labelled** format.

If any value label or user-defined missing value is added to a numeric or a character vector, it will be automatically converted into a labelled vector.

**Values of the vector will remain unchanged.**

## Miscellaneous

#### `nolabel_to_na(x)` or `df %>% nolabel_to_na()`

For labelled vectors, values without a value label will be converted into NA

#### `val_labels_to_na(x)` or `df %>% val_labels_to_na()`

For labelled vectors, values with a value label will be converted into NA

#### `df2 %>% copy_labels_from(df1)`

Copy labels and user-defined missing values from `df1` to `df2` based on shared column names. Useful when attributes are lost after some data manipulation.

#### `x %>% recode_if(condition, value)`

Recode only some values based on condition.

#### `recode(x, `2` = 1, `3` = 2)`

Apply `dplyr::recode()` to a labelled vector. Attached value labels will remain unchanged.

#### `recode(x, `2` = 1, .combine_value_labels = TRUE)`

Will combine value labels of original values merged together to produce new value labels, to be checked.

#### `update_labelled(x)` or `df %>% update_labelled()`

If `x/df` was imported/created using an older version of **haven** or **labelled**, you may encounter some unexpected results. `update_labelled()` will update all labelled vectors to be consistent with the current implementation.