Labelling data is typically a task for end-users and is applied in own scripts or functions rather than in packages. However, sometimes it can be useful for both end-users and package developers to have a flexible way to add variable and value labels to their data. In such cases, quasiquotation is helpful.

This vignette demonstrate how to use quasiquotation in *sjlabelled* to label your data.

**Adding value labels to variables using quasiquotation**

Usually, set\_labels() can be used to add value labels to variables. The syntax of this function is easy to use, and set\_labels()allows to add value labels to multiple variables at once, if these variables share the same value labels.

In the following examples, we will use the frq() function, that shows an extra **label**-column containing *value labels*, if the data is labelled. If the data has *no* value labels, this column is not shown in the output.

library(sjlabelled)

library(sjmisc) # for frq()-function

library(rlang)

library(quasiquotation)

# unlabelled data

dummies <- data.frame(

dummy1 = sample(1:3, 40, replace = TRUE),

dummy2 = sample(1:3, 40, replace = TRUE),

dummy3 = sample(1:3, 40, replace = TRUE)

)

# set labels for all variables in the data frame

test <- set\_labels(dummies, labels = c("low", "mid", "hi"))

[attr](https://www.rdocumentation.org/packages/base/topics/attr?tap_a=5644-dce66f&tap_s=10907-287229)(test$dummy1, "labels")

#> low mid hi

#> 1 2 3

frq(test, dummy1)

#>

#> # dummy1

#> # total N=40 valid N=40 mean=2.23 sd=0.86

#>

#> val label frq raw.prc valid.prc cum.prc

#> 1 low 11 27.5 27.5 27.5

#> 2 mid 9 22.5 22.5 50.0

#> 3 hi 20 50.0 50.0 100.0

#> NA NA 0 0.0 NA NA

# and set same value labels for two of three variables

test <- set\_labels(

dummies, dummy1, dummy2,

labels = c("low", "mid", "hi")

)

frq(test)

#>

#> # dummy1

#> # total N=40 valid N=40 mean=2.23 sd=0.86

#>

#> val label frq raw.prc valid.prc cum.prc

#> 1 low 11 27.5 27.5 27.5

#> 2 mid 9 22.5 22.5 50.0

#> 3 hi 20 50.0 50.0 100.0

#> NA NA 0 0.0 NA NA

#>

#> # dummy2

#> # total N=40 valid N=40 mean=2.10 sd=0.74

#>

#> val label frq raw.prc valid.prc cum.prc

#> 1 low 9 22.5 22.5 22.5

#> 2 mid 18 45.0 45.0 67.5

#> 3 hi 13 32.5 32.5 100.0

#> NA NA 0 0.0 NA NA

#>

#> # dummy3

#> # total N=40 valid N=40 mean=1.98 sd=0.83

#>

#> val frq raw.prc valid.prc cum.prc

#> 1 14 35.0 35.0 35.0

#> 2 13 32.5 32.5 67.5

#> 3 13 32.5 32.5 100.0

#> 0 0.0 NA NA

val\_labels() does the same job as set\_labels(), but in a different way. While set\_labels() requires variables to be specified in the ...-argument, and labels in the labels-argument, val\_labels() requires both to be specified in the ....

val\_labels() requires *named* vectors as argument, with the *left-hand side* being the name of the variable that should be labelled, and the *right-hand side* containing the labels for the values.

test <- val\_labels(dummies, dummy1 = c("low", "mid", "hi"))

attr(test$dummy1, "labels")

#> low mid hi

#> 1 2 3

# remaining variables are not labelled

frq(test)

#>

#> # dummy1

#> # total N=40 valid N=40 mean=2.23 sd=0.86

#>

#> val label frq raw.prc valid.prc cum.prc

#> 1 low 11 27.5 27.5 27.5

#> 2 mid 9 22.5 22.5 50.0

#> 3 hi 20 50.0 50.0 100.0

#> NA NA 0 0.0 NA NA

#>

#> # dummy2

#> # total N=40 valid N=40 mean=2.10 sd=0.74

#>

#> val frq raw.prc valid.prc cum.prc

#> 1 9 22.5 22.5 22.5

#> 2 18 45.0 45.0 67.5

#> 3 13 32.5 32.5 100.0

#> 0 0.0 NA NA

#>

#> # dummy3

#> # total N=40 valid N=40 mean=1.98 sd=0.83

#>

#> val frq raw.prc valid.prc cum.prc

#> 1 14 35.0 35.0 35.0

#> 2 13 32.5 32.5 67.5

#> 3 13 32.5 32.5 100.0

#> 0 0.0 NA NA

Unlike set\_labels(), val\_labels() allows the user to add *different* value labels to different variables in one function call. Another advantage, or difference, of val\_labels() is it’s flexibility in defining variable names and value labels by using quasiquotation.

**Add labels that are stored in a vector**

To use quasiquotation, we need the **rlang** package to be installed and loaded. Now we can have labels in a character vector, and use !! to unquote this vector.

labels <- c("low\_quote", "mid\_quote", "hi\_quote")

test <- val\_labels(dummies, dummy1 = !! labels)

attr(test$dummy1, "labels")

#> low\_quote mid\_quote hi\_quote

#> 1 2 3

**Define variable names that are stored in a vector**

The same can be done with the names of *variables* that should get new value labels. We then need !! to unquote the variable name and := as assignment.

variable <- "dummy2"

test <- val\_labels(dummies, !! variable := c("lo\_var", "mid\_var", "high\_var"))

# no value labels

attr(test$dummy1, "labels")

#> NULL

# value labels

attr(test$dummy2, "labels")

#> lo\_var mid\_var high\_var

#> 1 2 3

**Both variable names and value labels are stored in a vector**

Finally, we can combine the above approaches to be flexible regarding both variable names and value labels.

variable <- "dummy3"

labels <- c("low", "mid", "hi")

test <- val\_labels(dummies, !! variable := !! labels)

attr(test$dummy3, "labels")

#> low mid hi

#> 1 2 3

**Adding variable labels using quasiquotation**

set\_label() is the equivalent to set\_labels() to add variable labels to a variable. The equivalent to val\_labels() is var\_labels(), which works in the same way as val\_labels(). In case of *variable* labels, a label-attribute is added to a vector or factor (instead of a labels-attribute, which is used for *value* labels).

The following examples show how to use var\_labels() to add variable labels to the data. We demonstrate this function without further explanation, because it is actually very similar to val\_labels().

dummy <- data.frame(

a = sample(1:4, 10, replace = TRUE),

b = sample(1:4, 10, replace = TRUE),

c = sample(1:4, 10, replace = TRUE)

)

# simple usage

test <- var\_labels(dummy, a = "first variable", c = "third variable")

attr(test$a, "label")

#> [1] "first variable"

attr(test$b, "label")

#> NULL

attr(test$c, "label")

#> [1] "third variable"

# quasiquotation for labels

v1 <- "First variable"

v2 <- "Second variable"

test <- var\_labels(dummy, a = !! v1, b = !! v2)

attr(test$a, "label")

#> [1] "First variable"

attr(test$b, "label")

#> [1] "Second variable"

attr(test$c, "label")

#> NULL

# quasiquotation for variable names

x1 <- "a"

x2 <- "c"

test <- var\_labels(dummy, !! x1 := "First", !! x2 := "Second")

attr(test$a, "label")

#> [1] "First"

attr(test$b, "label")

#> NULL

attr(test$c, "label")

#> [1] "Second"

# quasiquotation for both variable names and labels

test <- var\_labels(dummy, !! x1 := !! v1, !! x2 := !! v2)

attr(test$a, "label")

#> [1] "First variable"

attr(test$b, "label")

#> NULL

attr(test$c, "label")

#> [1] "Second variable"

**Conclusion**

As we have demonstrated, var\_labels() and val\_labels() are one of the most flexible and easy-to-use ways to add value and variable labels to our data. Another advantage is the consistent design of all functions in **sjlabelled**, which allows seamless integration into pipe-workflows.