# Data & Variable Transformation with simisc Cheat Sheet



sjmisc complements dplyr, and helps with data transformation tasks and recoding *variables*.

sjmisc works together seamlessly with dplyr and pipes. All functions are designed to support labelled data.







# Design Philosophy

The design of simisc functions follows the tidyverse-approach: first argument is always the data (either a *data frame* **or** *vector*), followed by variable names to be processed by the functions.

The returned object for each function *equals the type of the data-argument.* 

#### Vector input

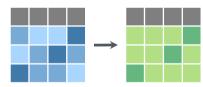
• If the data-argument is a *vector*, functions return a *vector*.



rec(mtcars\$carb, rec = "1,2=1; 3,4=2; else=3")

#### Data frame input

• If the data-argument is a *data frame*, functions return a *data frame*.



rec(mtcars, carb, rec = "1,2=1; 3,4=2; else=3")

#### The ...-ellipses Argument

Apply functions to a single variable, selected variables or to a complete data frame.

Variable selection is powered by dplyr's select(): Separate variables with comma, or use dplyr's select-helpers to select variables, e.g. *?rec*:

rec(mtcars, one\_of(c("gear", "carb")), rec = "min:3=1; 4:max=2")

rec(mtcars, gear, carb, rec = "min:3=1; 4:max=2")

## Descriptives and Summaries

Most of the simisc functions (including recodefunctions) also work on grouped data frames:

library(dplyr) efc %>% group\_by(e16sex, c172code) %>% frq(e42dep)

## Frequency Tables

frq(x, ..., sort.frq = c("none", "asc", "desc"),
 weight.by = NULL, auto.grp = NULL, ...)
Print frequency tables of (labelled) vectors. Uses
 variable labels as table header.

data(efc); frq(efc, e42dep, c161sex)

Use this data set in examples!

flat\_table(data, ..., margin = c("counts",
 "cell", "row", "col"), digits = 2,
 show.values = FALSE)

Print contingency tables of (labelled) vectors. Uses value labels.

flat\_table(efc, e42dep, c172code, e16sex)

count na(x,...)

Print frequency table of tagged NA values.

library(haven); x <- labelled(c(1:3,
tagged\_na("a", "a", "z")), labels =
c("Refused" = tagged\_na("a"), "N/A" =
tagged\_na("z")))
count\_na(x)</pre>

## **Descriptive Summary**

descr(x, ..., max.length = NULL)

Descriptive summary of data frames, including variable labels in output.

descr(efc, contains("cop"), max.length = 20)

## Finding Variables in a Data Frame

Use find\_var() to search for variables by names, value or variable labels. Returns vector/data frame.

# variables with "cop" in names and variable labels find var(efc, pattern = "cop", out = "df")

# variables with "level" in names and value labels find\_var(efc, "level", search = "name\_value")

#### Recode and Transform Variables

Recode functions add a *suffix* to new variables, so original variables are preserved.

By default, original input data frame and new created variables are returned. Use *append* = *FALSE* to return the recoded variables only.

rec(x, ..., rec, as.num = TRUE, var.label =
NULL, val.labels = NULL, append = TRUE,
suffix = "\_r")

Recode values, return result as numeric, character or categorical (factor).

rec(mtcars, carb, rec = "1,2=1; 3,4=2; else=3")

dicho(x, ..., dich.by = "median", as.num =
FALSE, var.label = NULL, val.labels = NULL,
append = TRUE, suffix = "\_d")

Dichotomise variable by median, mean or specific value.

dicho(mtcars, disp)

split\_var(x, ..., n, as.num = FALSE,
 val.labels = NULL, var.label = NULL,
 inclusive = FALSE, append = TRUE,
 suffix = "\_g")

Split variable into equal sized groups. Unlike dplyr::ntile(), does not split original categories into different values (see examples in ?split\_var(mtcars, mpg, disp, n = 3)

group\_var(x, ..., size = 5, as.num = TRUE, right.interval = FALSE, n = 30, append = TRUE, suffix = "\_gr")

Split variable into groups with equal value range, or into a max. # of groups (value range per group is adjusted to match # of groups).

group\_var(mtcars, mpg, disp, size = 5) group\_var(mtcars, mpg, size = "auto", n = 4)

std(x, ..., robust = "sd", include.fac = FALSE,
append = TRUE, suffix = "\_z")
Z-standardise variables. Also center().
std(efc, e17age, c160age)

recode\_to(x, ..., lowest = 0, highest = -1,
append = TRUE, suffix = "\_r0)
Shift ("renumber") categories or values.
recode\_to(mtcars\$gear)

#### Summarise Variables and Cases

The summary functions mostly mimic base R equivalents, but are designed to work together with pipes and dplyr.



row\_sums(x, ..., na.rm = TRUE, var = "rowsums", append = FALSE) Row sums of data frames.

row sums(efc, c82cop1:c90cop9)

row\_means(x, ..., n, var = "rowmeans",
append = FALSE)

Row means, for at least n valid (non-NA) values. row\_means(efc, c82cop1:c90cop9, n = 7)

row\_count(x, ..., count, var = "rowcount",
 append = FALSE)

Row-wise count # of values in data frames. Also col\_count().

row\_count(efc, c82cop1:c90cop9, count = 2)

#### Other Useful Functions

- add\_columns() and replace\_columns() to combine data frames, but either replace or preserve existing columns.
- set\_na() and replace\_na() to convert regular into missing values, or vice versa. replace\_na() also replaces specific tagged NA values only.
- remove\_var() and var\_rename() to remove variables from data frames, or rename variables.
- group\_str() to group similar string values. Useful for variables with similar, but not identically spelled string values that should be "merged".
- merge\_df() to full join data frames and preserve value and variable labels.
- to\_long() to gather multiple columns in data frames from wide into long format.

## Use with %>% and dplyr

# use sjmisc-functions in pipes mtcars %>% select(gear, carb) %>% rec(rec = "min:3=1; 4:max=2")

# use symisc-function inside mutate mtcars %>% select(gear, carb) %>% mutate( carb2 = rec(carb, rec = "1,2=0;3:8=1"), gear2 = rec(gear, rec = "3=1;4:max=2"))