

Package: recodeflow (via r-universe)

October 18, 2026

Type Package

Title Contains functions to interface with variable details sheets, including recoding variables and converting them to PMML

Version 0.1.2

Maintainer Yulric Sequeira <ysequeira@ohri.ca>

Description Recode and harmonize data using variable and details sheets.

Depends R (>= 3.5)

Imports sjlabelled, stringr, tidyr, haven, dplyr, magrittr,

License MIT + file LICENSE

URL <https://github.com/Big-Life-Lab/recodeflow>

BugReports <https://github.com/Big-Life-Lab/recodeflow/issues>

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests DT, kableExtra, knitr, rmarkdown, readr, testthat (>= 3.0.0), survival, tibble

Config/testthat/edition 3

VignetteBuilder knitr

Repository <https://big-life-lab.r-universe.dev>

Date/Publication 2025-04-23 17:03:17 UTC

RemoteUrl <https://github.com/big-life-lab/recodeflow>

RemoteRef HEAD

RemoteSha b304ddc737fc8736b1fe730abb668d1c36867481

Contents

get_table_name	2
is_equal	3
label_data	4
pbc	4
pbc_metadata	5
pbc_variable_details	6
pbc_variables	7
rec_with_table	7
select_vars_by_role	11
set_data_labels	11
tester_variable_details	12
tester_variables	13
Index	14

get_table_name	Returns the name of the table for a table start variable
----------------	--

Description

Returns the name of the table for a table start variable

Usage

get_table_name(table_feeder_var)

Arguments

table_feeder_var
string The table variable start

Value

string The extracted table name

Examples

```
# Extract table names from table feeder variables
get_table_name("$table:lookup_codes") # Returns "lookup_codes"
get_table_name("$table:reference") # Returns "reference"
get_table_name("$table:values") # Returns "values"
```

`is_equal`*Checks whether two values are equal including NA*

Description

Compared to the base "==" operator in R, this function returns true if the two values are NA whereas the base "==" operator returns NA

Usage

```
is_equal(v1, v2)
```

Arguments

<code>v1</code>	variable 1
<code>v2</code>	variable 2

Value

boolean value of whether or not v1 and v2 are equal

Examples

```
is_equal(1,2)
# FALSE

is_equal(1,1)
# TRUE

1==NA
# NA

is_equal(1,NA)
# FALSE

NA==NA
# NA

is_equal(NA,NA)
# TRUE
```

label_data	<i>label_data</i>
------------	-------------------

Description

Attaches labels to the data_to_label to preserve metadata

Usage

```
label_data(label_list, data_to_label)
```

Arguments

label_list the label list object that contains extracted labels from variable details
data_to_label The data that is to be labeled

Value

Returns labeled data

pbc	<i>The pbc dataset</i>
-----	------------------------

Description

The pbc dataset

Usage

```
pbc
```

Format

A data frame with 418 observations and 20 variables.

id case number

time number of days between registration and the earlier of death, transplantation, or study analysis time

status status at endpoint, 0/1/2 for censored, transplant, dead

trt 1/2/NA for D-penicillamine, placebo, or not randomized

age age in years

sex m/f

ascites presence of ascites

hepato presence of hepatomegaly or enlarged liver

spiders blood vessel malformations in the skin
edema 0 no edema, 0.5 untreated or successfully treated, 1 edema despite diuretic therapy
bili serum bilirubin (mg/dl)
chol serum cholesterol (mg/dl)
albumin serum albumin (g/dl)
copper urine copper (ug/day)
alk.phos alkaline phosphatase (U/liter)
ast aspartate aminotransferase (U/ml)
trig triglycerides (mg/dl)
platelet platelet count
protime standardised blood clotting time
stage histologic stage of disease (1, 2, 3, or 4)

Source

<https://cran.r-project.org/web/packages/survival/survival.pdf>

pbm_metadata	<i>Metadata for the pbm dataset using the DCIM standard</i>
--------------	---

Description

Metadata for the pbm dataset using the DCIM standard

Usage

```
pbm_metadata
```

Format

A list containing DCMI metadata:

title title
creator creator
subject subject
description description
publisher publisher
date date
type type
format format
identifier identifier
source source
language language
rights rights
references references

pbc_variable_details *Variable details sheet for the pbc dataset*

Description

Variable details sheet for the pbc dataset

Usage

pbc_variable_details

Format

A data frame with 69 rows and 16 columns:

variable variable name

dummyVariable dummy variable name

typeEnd end type

databaseStart database start

variableStart variable start

typeStart start type

recEnd record end

recStart record start

catLabel category label

catLabelLong category long label

numValidCat number of valid categories (numeric)

units logical indicating presence of units

notes logical indicating presence of notes

catStartLabel category start label

variableStartShortLabel variable start short label

variableStartLabel variable start label

pbv_variables	<i>Variables sheet for the pbv dataset</i>
---------------	--

Description

Variables sheet for the pbv dataset

Usage

pbv_variables

Format

A data frame with 24 rows and 11 columns:

variable variable name

label variable label

labelLong variable label long

subject subject

section section

variableType variable type

databaseStart database start

units units

variableStart variable start

notes logical indicating presence of notes

description logical indicating presence of description

rec_with_table	<i>Recode with Table</i>
----------------	--------------------------

Description

Creates new variables by recoding variables in a dataset using the rules specified in a variables details sheet

Usage

```

rec_with_table(
  data,
  variables = NULL,
  database_name = NULL,
  variable_details = NULL,
  else_value = NA,
  append_to_data = FALSE,
  log = FALSE,
  notes = TRUE,
  var_labels = NULL,
  custom_function_path = NULL,
  attach_data_name = FALSE,
  id_role_name = NULL,
  name_of_environment_to_load = NULL,
  append_non_db_columns = FALSE,
  tables = list()
)

```

Arguments

<code>data</code>	A dataframe containing the variables to be recoded. Can also be a named list of dataframes.
<code>variables</code>	Character vector containing the names of the new variables to recode to or a dataframe containing a variables sheet.
<code>database_name</code>	A String containing the name of the database containing the original variables which should match up with a database from the databaseStart column in the variables details sheet. Should be a character vector if data is a named list where each vector item matches a name in the data list and also matches with a value in the databaseStart column of a variable details sheet.
<code>variable_details</code>	A dataframe containing the specifications for recoding.
<code>else_value</code>	Value (string, number, integer, logical or NA) that is used to replace any values that are outside the specified ranges (no rules for recoding).
<code>append_to_data</code>	Logical, if TRUE (default), the newly created variables will be appended to the original dataset.
<code>log</code>	Logical, if FALSE (default), a log containing information about the recoding will not be printed.
<code>notes</code>	Logical, if FALSE (default), will not print the content inside the ‘Note‘ column of the variable being recoded.
<code>var_labels</code>	labels vector to attach to variables in variables
<code>custom_function_path</code>	string containing the path to the file containing functions to run for derived variables. This file will be sourced and its functions loaded into the R environment.
<code>attach_data_name</code>	logical to attach name of database to end table

id_role_name	name for the role to be used to generate id column
name_of_environment_to_load	Name of package to load variables and variable_details from
append_non_db_columns	boolean determining if data not present in this cycle should be appended as NA
tables	named list of data.frame A list of reference tables that can be passed as parameters into the function for a derived variable

Details

The **variable_details** dataframe needs the following columns:

variable Name of the new variable created. The name of the new variable can be the same as the original variable if it does not change the original variable definition

toType type the new variable *cat = categorical, cont = continuous*

databaseStart Names of the databases that the original variable can come from. Each database name should be separated by a comma. For eg., "cchs2001_p, cchs2003_p, cchs2005_p, cchs2007_p"

variableStart Names of the original variables within each database specified in the databaseStart column. For eg. , "cchs2001_p::RACA_6A, cchs2003_p::RACC_6A, ADL_01". The final variable specified is the name of the variable for all other databases specified in databaseStart but not in this column. For eg., ADL_01 would be the original variable name in the cchs2005_p and cchs2007_p databases.

fromType variable type of start variable. *cat = categorical or factor variable cont = continuous variable (real number or integer)*

recTo Value to recode to

recFrom Value/range being recoded from

Each row in the *variables details* sheet encodes the rule for recoding value(s) of the original variable to a category in the new variable. The categories of the new variable are encoded in the *recTo* column and the value(s) of the original variable that recode to this new value are encoded in the *recFrom* column. These recode columns follow a syntax similar to the *sjmisc::rec()* function. Whereas in the *sjmisc::rec()* function the recoding rules are in one string, in the variables details sheet they are encoded over multiple rows and columns (recFrom and recTo). For eg., a recoding rule in the *sjmisc* function would look like "1=2;2=3" whereas in the variables details sheet this would be encoded over two rows with recFrom and recTo values of the first row being 1 and 2 and similarly for the second row it would be 2 and 3. The rules for describing recoding pairs are shown below:

recode pairs Each recode pair is a row

multiple values Multiple values from the old variable that should be recoded into a new category of the new variable should be separated with a comma. e.g., *recFrom = "1,2"; recTo = 1* will recode values of 1 and 2 in the original variable to 1 in the new variable

value range A value range is indicated by a colon, e.g. *recFrom = "1:4"; recTo = 1* will recode all values from 1 to 4 into 1

min and max minimum and maximum values are indicated by *min* (or *lo*) and *max* (or *hi*), e.g. *recFrom = "min:4"; recTo = 1* will recode all values from the minimum value of the original variable to 4 into 1

"else" All other values, which have not been specified yet, are indicated by *else*, e.g. *recFrom* = "else"; *recTo* = NA will recode all other values (not specified in other rows) of the original variable to "NA")

"copy" the *else* token can be combined with *copy*, indicating that all remaining, not yet recoded values should stay the same (are copied from the original value), e.g. *recFrom* = "else"; *recTo* = "copy"

NA's NA values are allowed both for the original and the new variable, e.g. *recFrom* "NA"; *recTo* = 1. or "*recFrom* = "3:5"; *recTo* = "NA" (recodes all NA into 1, and all values from 3 to 5 into NA in the new variable)

Value

a dataframe that is recoded according to rules in *variable_details*.

Examples

```
var_details <-
  data.frame(
    "variable" = c("time", rep("status", times = 3), rep("trt", times = 2), "age", rep("sex", times = 2), rep("ascites", times = 2), rep("hepato", times = 2), rep("spiders", times = 2), rep("edema", times = 2), rep("bili", times = 2), rep("chol", times = 2), rep("albumin", times = 2)),
    "dummyVariable" = c("NA", "status0", "status1", "status2", "trt1", "trt2", "NA", "sexM", "sexF", "ascites0", "ascites1", "hepato0", "hepato1", "spiders0", "spiders1", "edema0", "edema1", "bili0", "bili1", "chol0", "chol1", "albumin0", "albumin1"),
    "typeEnd" = c("cont", rep("cat", times = 3), rep("cat", times = 2), "cont", rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2)),
    "databaseStart" = rep("tester1, tester2", times = 31),
    "variableStart" = c("[time]", rep("[status]", times = 3), rep("[trt]", times = 2), "[age]", rep("[sex]", times = 2), rep("[ascites]", times = 2), rep("[hepato]", times = 2), rep("[spiders]", times = 2), rep("[edema]", times = 2), rep("[bili]", times = 2), rep("[chol]", times = 2), rep("[albumin]", times = 2)),
    "typeStart" = c("cont", rep("cat", times = 3), rep("cat", times = 2), "cont", rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2), rep("cat", times = 2)),
    "recEnd" = c("copy", "0", "1", "2", "1", "2", "copy", "m", "f", "0", "1", "0", "1", "0", "1", "0.0", "0.5", "1.0", rep("copy", times = 13)),
    "catLabel" = c("", "status 0", "status 1", "status 2", "trt 1", "trt 2", "", "sex m", "sex f", "ascites 0", "ascites 1", "hepato 0", "hepato 1", "spiders 0", "spiders 1", "edema 0", "edema 1", "bili 0", "bili 1", "chol 0", "chol 1", "albumin 0", "albumin 1"),
    "catLabelLong" = c("", "status 0", "status 1", "status 2", "trt 1", "trt 2", "", "sex m", "sex f", "ascites 0", "ascites 1", "hepato 0", "hepato 1", "spiders 0", "spiders 1", "edema 0", "edema 1", "bili 0", "bili 1", "chol 0", "chol 1", "albumin 0", "albumin 1"),
    "recStart" = c("else", "0", "1", "2", "1", "2", "else", "m", "f", "0", "1", "0", "1", "0", "1", "0.0", "0.5", "1.0", rep("else", times = 13)),
    "catStartLabel" = c("", "status 0", "status 1", "status 2", "trt 1", "trt 2", "", "sex m", "sex f", "ascites 0", "ascites 1", "hepato 0", "hepato 1", "spiders 0", "spiders 1", "edema 0", "edema 1", "bili 0", "bili 1", "chol 0", "chol 1", "albumin 0", "albumin 1"),
    "variableStartShortLabel" = c("time", rep("status", times = 3), rep("trt", times = 2), "age", rep("sex", times = 2), rep("ascites", times = 2), rep("hepato", times = 2), rep("spiders", times = 2), rep("edema", times = 2), rep("bili", times = 2), rep("chol", times = 2), rep("albumin", times = 2)),
    "variableStartLabel" = c("time", rep("status", times = 3), rep("trt", times = 2), "age", rep("sex", times = 2), rep("ascites", times = 2), rep("hepato", times = 2), rep("spiders", times = 2), rep("edema", times = 2), rep("bili", times = 2), rep("chol", times = 2), rep("albumin", times = 2)),
    "units" = rep("NA", times = 31),
    "notes" = rep("This is sample survival pbc data", times = 31)
  )
var_sheet <-
  data.frame(
    "variable" = c("time", "status", "trt", "age", "sex", "ascites", "hepato", "spiders", "edema", "bili", "chol", "albumin"),
    "label" = c("time", "status", "trt", "age", "sex", "ascites", "hepato", "spiders", "edema", "bili", "chol", "albumin"),
    "labelLong" = c("time", "status", "trt", "age", "sex", "ascites", "hepato", "spiders", "edema", "bili", "chol", "albumin"),
    "section" = rep("tester", times = 19),
    "subject" = rep("tester", times = 19),
    "variableType" = c("cont", "cat", "cat", "cont", "cat", "cat", "cat", "cat", "cat", rep("cont", times = 9), "cat"),
    "databaseStart" = rep("tester1, tester2", times = 19),
    "units" = rep("NA", times = 19),
    "variableStart" = c("[time]", "[status]", "[trt]", "[age]", "[sex]", "[ascites]", "[hepato]", "[spiders]", "[edema]", "[bili]", "[chol]", "[albumin]"),
    "typeEnd" = c("cont", "cat", "cat", "cont", "cat", "cat", "cat", "cat", "cat", rep("cont", times = 9), "cat"),
    "typeStart" = c("cont", "cat", "cat", "cont", "cat", "cat", "cat", "cat", "cat", rep("cont", times = 9), "cat"),
    "recEnd" = c("copy", "0", "1", "2", "1", "2", "copy", "m", "f", "0", "1", "0", "1", "0", "1", "0.0", "0.5", "1.0", rep("copy", times = 13)),
    "catLabel" = c("", "status 0", "status 1", "status 2", "trt 1", "trt 2", "", "sex m", "sex f", "ascites 0", "ascites 1", "hepato 0", "hepato 1", "spiders 0", "spiders 1", "edema 0", "edema 1", "bili 0", "bili 1", "chol 0", "chol 1", "albumin 0", "albumin 1"),
    "catLabelLong" = c("", "status 0", "status 1", "status 2", "trt 1", "trt 2", "", "sex m", "sex f", "ascites 0", "ascites 1", "hepato 0", "hepato 1", "spiders 0", "spiders 1", "edema 0", "edema 1", "bili 0", "bili 1", "chol 0", "chol 1", "albumin 0", "albumin 1"),
    "recStart" = c("else", "0", "1", "2", "1", "2", "else", "m", "f", "0", "1", "0", "1", "0", "1", "0.0", "0.5", "1.0", rep("else", times = 13)),
    "catStartLabel" = c("", "status 0", "status 1", "status 2", "trt 1", "trt 2", "", "sex m", "sex f", "ascites 0", "ascites 1", "hepato 0", "hepato 1", "spiders 0", "spiders 1", "edema 0", "edema 1", "bili 0", "bili 1", "chol 0", "chol 1", "albumin 0", "albumin 1"),
    "variableStartShortLabel" = c("time", "status", "trt", "age", "sex", "ascites", "hepato", "spiders", "edema", "bili", "chol", "albumin"),
    "variableStartLabel" = c("time", "status", "trt", "age", "sex", "ascites", "hepato", "spiders", "edema", "bili", "chol", "albumin"),
    "units" = rep("NA", times = 19),
    "notes" = rep("This is sample survival pbc data", times = 19)
  )
library(survival)
tester1 <- survival::pbc[1:209,]
tester2 <- survival::pbc[210:418,]
db_name1 <- "tester1"
db_name2 <- "tester2"
```

```

rec_sample1 <- rec_with_table(data = tester1,
  variables = var_sheet,
  variable_details = var_details,
  database_name = db_name1)

rec_sample2 <- rec_with_table(data = tester2,
  variables = var_sheet,
  variable_details = var_details,
  database_name = db_name2)

```

select_vars_by_role	<i>Vars selected by role</i>
---------------------	------------------------------

Description

Selects variables from variables sheet based on passed roles

Usage

```
select_vars_by_role(roles, variables)
```

Arguments

roles	a vector containing a single or multiple roles to match by
variables	the variables sheet containing variable info

Value

a vector containing the variable names that match the passed roles

set_data_labels	<i>Set Data Labels</i>
-----------------	------------------------

Description

sets labels for passed database, Uses the names of final variables in variable_details/variables_sheet as well as the labels contained in the passed dataframes

Usage

```
set_data_labels(data_to_label, variable_details, variables_sheet = NULL)
```

Arguments

data_to_label newly transformed dataset
variable_details
 variable_details.csv
variables_sheet
 variables.csv

Value

labeled data_to_label

tester_variable_details
<i>Example variable details sheet for vignettes</i>

Description

Example variable details sheet for vignettes

Usage

tester_variable_details

Format

A data frame with 69 rows and 16 columns:

variable variable name
dummyVariable dummy variable name
typeEnd end type
databaseStart database start
variableStart variable start
typeStart start type
recEnd record end
recStart record start
catLabel category label
catLabelLong category long label
numValidCat number of valid categories (numeric)
units logical indicating presence of units
notes logical indicating presence of notes
catStartLabel category start label
variableStartShortLabel variable start short label
variableStartLabel variable start label

tester_variables	<i>Example variables sheet for vignettes</i>
------------------	--

Description

Example variables sheet for vignettes

Usage

tester_variables

Format

A data frame with 24 rows and 11 columns:

variable variable name

label variable label

labelLong variable label long

subject subject

section section

variableType variable type

databaseStart database start

units units

variableStart variable start

notes logical indicating presence of notes

description logical indicating presence of description

Index

* datasets

- [pbc](#), [4](#)
- [pbc_metadata](#), [5](#)
- [pbc_variable_details](#), [6](#)
- [pbc_variables](#), [7](#)
- [tester_variable_details](#), [12](#)
- [tester_variables](#), [13](#)

[get_table_name](#), [2](#)

[is_equal](#), [3](#)

[label_data](#), [4](#)

[pbc](#), [4](#)

[pbc_metadata](#), [5](#)

[pbc_variable_details](#), [6](#)

[pbc_variables](#), [7](#)

[rec_with_table](#), [7](#)

[select_vars_by_role](#), [11](#)

[set_data_labels](#), [11](#)

[tester_variable_details](#), [12](#)

[tester_variables](#), [13](#)