Package 'cchsflow'

October 25, 2021

Type Package

Title Transforming and Harmonizing CCHS Variables

Version 2.0.0

Date 2021-10-25

Depends R (>= 3.2), haven (>= 1.1.2), dplyr (>= 0.8.2), sjlabelled (>= 1.0.17), stringr (>= 1.2.0), magrittr

Description Supporting the use of the Canadian Community Health Survey (CCHS) by transforming variables from each cycle into harmonized, consistent versions that span survey cycles (currently, 2001 to 2018). CCHS data used in this library is accessed and adapted in accordance to the Statistics Canada Open Licence Agreement. This package uses rec_with_table(), which was developed from 'sjmisc' rec(). Lüdecke D (2018). ``sjmisc: Data and Variable Transformation Functions". Journal of Open Source Software, 3(26), 754. <doi:10.21105/joss.00754>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/Big-Life-Lab/cchsflow

BugReports https://github.com/Big-Life-Lab/cchsflow/issues

RoxygenNote 7.1.2

Suggests testthat (>= 2.1.0)

NeedsCompilation no

Author Doug Manuel [aut, cph] (<https://orcid.org/0000-0003-0912-0845>),
Warsame Yusuf [aut, cre],
Rostyslav Vyuha [aut],
Kitty Chen [aut],
Carol Bennett [aut],
Yulric Sequeira [ctb],
The Ottawa Hospital [cph]

Maintainer Warsame Yusuf <waryusuf@ohri.ca>

Repository CRAN

Date/Publication 2021-10-25 18:20:01 UTC

R topics documented:

`adjusted_bmi_fun` *Adjusted Body Mass Index (BMI) derived variable*

Description

This function creates a harmonized adjusted BMI variable. A systematic review of the literature concluded that the use of self-reported data among adults underestimates weight and overestimates height, resulting in lower estimates of obesity than those obtained from measured data. Using data from the 2005 Canadian Community Health Survey (CCHS) subsample, where both measured and self-reported values were collected, correction equations have been developed (Connor Gorber et al. 2008). Differences between corrected estimates of obesity from the CCHS and measured estimates from the Canadian Health Measures Survey is monitored over time to determine if the bias in self-reported values is changing and if new correction equations need to be developed. Adjusted BMI variable is first introduced in the CCHS 2015 cycle.

adjusted_bmi_fun() creates a derived variable (HWTGCOR_der) that is harmonized across all CCHS cycles. This function takes the BMI by dividing weight by the square of height, and adds a correction value based on sex.

Usage

```
adjusted_bmi_fun(DHH_SEX, HWTGHTM, HWTGWTK)
```

Arguments

DHH_SEX	CCHS variable for sex; 1 = male, 2 = female
HWTGHTM	CCHS variable for height (in meters)
HWTGWTK	CCHS variable for weight (in kilograms)

Details

For HWTGCOR_der, there are no restrictions to age, height, weight, or pregnancy status. While pregnancy was consistent across all CCHS cycles, its variable (MAM_037) was not available in the PUMF CCHS datasets so it could not be harmonized and included into the function.

HWTGCOR_der uses the CCHS variables for sex, height and weight that have been transformed by cchsflow. In order to generate a value for adjusted BMI across CCHS cycles, sex, height and weight must be transformed and harmonized.

Value

numeric value for adjusted BMI in the HWTGCOR_der variable

Note

In earlier CCHS cycles (2001 and 2003), height was collected in inches; while in later CCHS cycles (2005+) it was collected in meters. To harmonize values across cycles, height was converted to meters (to 3 decimal points). Weight was collected in kilograms across all CCHS cycles, so no transformations were required in the harmonization process.

Examples

```
# Using adjusted_bmi_fun() to create adjusted BMI values between cycles
# adjusted_bmi_fun() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform the derived BMI variable, use rec_with_table() for each cycle
# and specify HWTGCOR_der, along with sex (DHH_SEX), height (HWTGHTM) and
# weight (HWTGWTK).Then by using merge_rec_data(), you can combined
# HWTGBMI_der across cycles.

library(cchsflow)
adjustedbmi2001 <- rec_with_table(
  cchs2001_p, c(
    "HWTGHTM",
    "HWTGWTK",
    "DHH_SEX",
    "HWTGCOR_der"
  )
)

head(adjustedbmi2001)

adjustedbmi2011_2012 <- rec_with_table(
  cchs2011_2012_p, c(
```

```
    "HWTGHTM",
    "HWTGWTK",
    "DHH_SEX",
    "HWTGCOR_der"
  )
)

tail(adjustedbmi2011_2012)

combined_bmi <- merge_rec_data(adjustedbmi2001, adjustedbmi2011_2012)
head(combined_bmi)
tail(combined_bmi)

# adjusted_bmi_fun() can also generate a value for BMI if you input your sex,
# and a value for height and weight. Let's say your sex is male, height is
# 170cm (1.7m) and your weight is 50kg, your BMI can be calculated as follows:

library(cchsflow)
adjusted_BMI <- adjusted_bmi_fun(DHH_SEX = 1, HWTGHTM = 1.7, HWTGWTK = 50)
print(adjusted_BMI)
```

`adl_fun` *Derived needs help with tasks*

Description

This derived variable (ADL_der) is based on the CCHS derived variable ADLF6R which flags respondents who need help with tasks based on their response to the various activities of daily living (ADL) variables.

Usage

```
adl_fun(ADL_01, ADL_02, ADL_03, ADL_04, ADL_05)
```

Arguments

ADL_01	Needs help preparing meals
ADL_02	Needs help getting to appointments/errands
ADL_03	Needs help doing housework
ADL_04	Needs help doing personal care
ADL_05	Needs help moving inside house

Details

The CCHS derived variable ADLF6R uses different ADL variables across the various CCHS survey cycles. This newly derived variable (ADL_der) uses ADL variables that are consistent across CCHS cycles.

In the 2001 CCHS survey cycle, the ADLF6R variable examines the following ADL variables:

1. ADL_01 - Needs help preparing meals
2. ADL_02 - Needs help getting to appointments/errands
3. ADL_03 - Needs help doing housework
4. ADL_04 - Needs help doing personal care
5. ADL_05 - Needs help moving inside house
6. ADL_07 - Needs help doing heavy household chores

In the 2003-2005 CCHS survey cycles, the ADLF6R variable examines the following ADL variables:

1. ADL_01 - Needs help preparing meals
2. ADL_02 - Needs help getting to appointments/errands
3. ADL_03 - Needs help doing housework
4. ADL_04 - Needs help doing personal care
5. ADL_05 - Needs help moving inside house
6. ADL_06 - Needs help doing finances
7. ADL_07 - Needs help doing heavy household chores

In the 2007-2014 CCHS survey cycles, the ADLF6R variable examines the following ADL variables:

1. ADL_01 - Needs help preparing meals
2. ADL_02 - Needs help getting to appointments/errands
3. ADL_03 - Needs help doing housework
4. ADL_04 - Needs help doing personal care
5. ADL_05 - Needs help moving inside house
6. ADL_06 - Needs help doing finances

This newly derived variable (ADL_der) uses ADL_01 to ADL_05 which are consistent across all survey cycles. For any single CCHS survey year, it is appropriate to use ADLF6R. ADL_der is recommended when using multiple survey cycles.

Value

A derived variable (ADL_der) with 2 categories:

1. - Needs help with tasks
2. - Does not need help with tasks

Examples

```
# Using adl_fun() to create ADL_der values across CCHS cycles
# adl_fun() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform ADL_der, use rec_with_table() for each CCHS cycle
# and specify ADL_der, along with the various ADL variables.
# Then by using merge_rec_data() you can combine ADL_der across cycles.

library(cchsflow)
adl2001 <- rec_with_table(
  cchs2001_p, c(
    "ADL_01", "ADL_02", "ADL_03", "ADL_04", "ADL_05", "ADL_der"
  )
)

head(adl2001)

adl2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "ADL_01", "ADL_02", "ADL_03", "ADL_04", "ADL_05", "ADL_der"
  )
)

tail(adl2009_2010)

combined_adl <- merge_rec_data(adl2001, adl2009_2010)

head(combined_adl)

tail(combined_adl)

# Using adl_fun() to generate to ADL_der based on user inputted values.
#
# Let's say you do not need help preparing meals, you need help getting to
# appointments or errands, you need help doing housework, do not need help
# doing personal care, and do not need help moving inside the house. Using
# adl_fun() we can check if you need help doing tasks

ADL_der <- adl_fun(2, 1, 1, 2, 2)

print(ADL_der)
```

`adl_score_5_fun` *The number of activities of daily living tasks that require help.*

Description

A 6 category variable (ADL_score_5) representing the number of activities of daily living tasks that require help. This variable tallies the number of daily living tasks that a respondent requires help

with based on various ADL variables that a respondent answered yes or no to. The ADL variables used are common across all CCHS cycles from 2001 to 2014.

Usage

```
adl_score_5_fun(ADL_01, ADL_02, ADL_03, ADL_04, ADL_05)
```

Arguments

ADL_01	Needs help preparing meals.
ADL_02	Needs help getting to appointments/errands.
ADL_03	Needs help doing housework.
ADL_04	Needs help doing personal care.
ADL_05	Needs help moving inside house.

Value

A derived variable (ADL_score_5) with 6 categories:

1. 0 - Needs help with 0 tasks
2. 1 - Needs help with at least 1 task
3. 2 - Needs help with at least 2 tasks
4. 3 - Needs help with at least 3 tasks
5. 4 - Needs help with at least 4 tasks
6. 5 - Needs help with at least 5 tasks

Examples

```
# Use adl_score_5_fun() to create the variable ADL_score_5 across CCHS
# cycles adl_score_5_fun() is specified in variable_details.csv along with
# the CCHS variables and cycles included.

# To transform ADL_score_5, use rec_with_table() for each CCHS cycle
# and specify ADL_score_5, along with the various ADL variables.
# Then by using merge_rec_data() you can combine ADL_der across cycles.

library(cchsflow)
adl2001 <- rec_with_table(
  cchs2001_p, c(
    "ADL_01", "ADL_02", "ADL_03", "ADL_04", "ADL_05", "ADL_score_5"
  )
)

head(adl2001)

adl2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "ADL_01", "ADL_02", "ADL_03", "ADL_04", "ADL_05", "ADL_score_5"
  )
```

```
)

tail(adl2009_2010)

combined_adl <- merge_rec_data(adl2001, adl2009_2010)

head(combined_adl)

tail(combined_adl)

# Using adl_score_5_fun() to generate to ADL_score_5 based on user inputted
# values.
# Let's say you do not need help preparing meals, you need help getting to
# appointments or errands, you need help doing housework, do not need help
# doing personal care, and do not need help moving inside the house. Using
# adl_score_5_fun() we can check the number of tasks you need help with

ADL_score_5 <- adl_score_5_fun(2, 1, 1, 2, 2)

print(ADL_score_5)
```

`age_cat_fun` *Derived categorical age*

Description

This is a derived categorical age variable (DHHGAGE_C) that groups various age categories across all CCHS cycles. This is based on the continuous age variable (DHHGAGE_cont) that is harmonious across all CCHS cycles.

The categories of this new age variable are as follows:

1. 12 to 14 years
2. 15 to 17 years
3. 18 to 19 years
4. 20 to 24 years
5. 25 to 29 years
6. 30 to 34 years
7. 35 to 39 years
8. 40 to 44 years
9. 45 to 49 years
10. 50 to 54 years
11. 55 to 59 years
12. 60 to 64 years
13. 65 to 69 years

14. 70 to 74 years
15. 75 to 79 years
16. 80 years or more

Usage

```
age_cat_fun(DHHGAGE_cont)
```

Arguments

DHHGAGE_cont	continuous age variable

Details

The categories in the grouped age variable (DHHGAGE) vary between CCHS cycles. As such, a continuous age variable (DHHGAGE_cont) was created that harmonized age across all CCHS cycles by taking the midpoint of each age category. This new age variable (DHHGAGE_C) categorizes age based on the categories used in CCHS cycles from 2007 to 2014.

Value

a categorical age variable (DHHGAGE_C)

Examples

```
# Using age_cat_fun() to create categorical age values from DHHGAGE_cont
# age_cat_fun() is specified in variable_details.csv along with the CCHS
# variables and cycles included.

# To generate DHHGAGE_C in a cycle, use rec_with_table() and specify
# DHHGAGE_C along with DHHGAGE_cont.

library(cchsflow)

cat_age2009_2010 <- rec_with_table(
   cchs2009_2010_p,  c(
     "DHHGAGE_cont", "DHHGAGE_C"
     )
   )
```

ALCDTTM *Type of drinker (12 months)*

Description

NOTE: this is not a function.

This is a categorical variable derived by Statistics Canada that uses various intermediate alcohol variables to categorize individuals into 3 distinct groups:

1. Regular Drinker
2. Occasional Drinker
3. No drink in the last 12 months.

Usage

```
ALCDTTM(ALCDTTM)
```

Arguments

ALCDTTM	cchsflow variable name for type of drinker (12 months)

Details

This variable was introduced in the 2007-2008 cycle of the CCHS, and became the sole derived variable that categorized people into various drinker types from 2009 onwards. Unlike ALCDTYP, this variable does not distinguish between former and never drinkers.

Examples

```
library(cchsflow)
 ?ALCDTTM
```

ALCDTYP *Type of drinker*

Description

NOTE: this is not a function.

This is a categorical variable derived by Statistics Canada that uses various intermediate alcohol variables to categorize individuals into 4 distinct groups:

1. Regular Drinker
2. Occasional Drinker
3. Former Drinker
4. Never Drinker

Usage

```
ALCDTYP(ALCDTYP)
```

Arguments

ALCDTYP	cchsflow variable name for type of drinker

Details

This variable is used in CCHS cycles from 2001 to 2007. How it was derived remained consistent during these years.

Starting in 2007, Statistics Canada created a derived variable that looked at drinking type in the last 12 months. This new derived variable did not distinguish between former and never drinkers. If your research requires you to differentiate between former and never drinkers, we recommend using earlier cycles of the CCHS.

Examples

```
library(cchsflow)
 ?ALCDTYP
```

ALWDDLY *Average daily alcohol consumption*

Description

NOTE: this is not a function.

This is a continuous variable derived by Statistics Canada that quantifies the mean daily consumption of alcohol. This takes the value of ALWDWKY and divides it by 7.

Usage

```
ALWDDLY(ALWDDLY)
```

Arguments

ALWDDLY	cchsflow variable name for average daily alcohol consumption

Details

This variable is present in every CCHS cycle used in cchsflow, and how it was derived remains consistent.

Examples

```
library(cchsflow)
 ?ALWDDLY
```

ALWDWKY *Number of drinks consumed in the past week*

Description

NOTE: this is not a function.

This is a continuous variable derived by Statistics Canada that quantifies the amount of alcohol that is consumed in a week. This is calculated by adding the number of drinks consumed each day in the past week. Respondents of each CCHS cycle are asked how much alcohol they have consumed each day in the past week (ie. how much alcohol did you consume on Sunday, how much did you consume on Monday etc.). Each day is considered an individual variable and ALWDWKY takes the sum of all daily variables.

Usage

```
ALWDWKY(ALWDWKY)
```

Arguments

ALWDWKY	cchsflow variable name for number of drinks consumed in the past week

Details

This variable is present in every CCHS cycle used in cchsflow, and how it was derived remains consistent.

Examples

```
library(cchsflow)
 ?ALWDWKY
```

binge_drinker_fun *Binge drinking*

Description

This function creates a derived categorical variable that flags for binge drinking based on the number drinks consumed on a single day.

Usage

```
binge_drinker_fun(
  DHH_SEX,
  ALW_1,
  ALW_2A1,
  ALW_2A2,
  ALW_2A3,
  ALW_2A4,
  ALW_2A5,
  ALW_2A6,
  ALW_2A7
)
```

Arguments

DHH_SEX	sex of respondent (1 - male, 2 - female)
ALW_1	Drinks in the last week (1 - yes, 2 - no)
ALW_2A1	Number of drinks on Sunday
ALW_2A2	Number of drinks on Monday
ALW_2A3	Number of drinks on Tuesday
ALW_2A4	Number of drinks on Wednesday
ALW_2A5	Number of drinks on Thursday
ALW_2A6	Number of drinks on Friday
ALW_2A7	Number of drinks on Saturday

Details

In health research, binge drinking is defined as having an excess amount of alcohol in a single day. For males, this is defined as having five or more drinks; and for females it is four or more drinks. In the CCHS, respondents are asked to count the number of drinks they had during each day of the last week.

Value

Categorical variable (binge_drinker) with two categories:

1. 1 - binge drinker
2. 2 - non-binge drinker

Examples

```
# Using binge_drinker_fun() to create binge_drinker values across CCHS cycles
# binge_drinker_fun() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform binge_drinker, use rec_with_table() for each CCHS cycle
```

```
# and specify binge_drinker, along with the various alcohol and sex
# variables. Then by using bind_rows() you can combine binge_drinker
# across cycles.

library(cchsflow)
binge2001 <- rec_with_table(
  cchs2001_p, c(
    "ALW_1", "DHH_SEX", "ALW_2A1", "ALW_2A2", "ALW_2A3", "ALW_2A4",
    "ALW_2A5", "ALW_2A6", "ALW_2A7", "binge_drinker"
  )
)

head(binge2001)

binge2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "ALW_1", "DHH_SEX", "ALW_2A1", "ALW_2A2", "ALW_2A3", "ALW_2A4",
    "ALW_2A5", "ALW_2A6", "ALW_2A7", "binge_drinker"
  )
)

tail(binge2009_2010)

combined_binge <- bind_rows(binge2001, binge2009_2010)

head(combined_binge)

tail(combined_binge)

# Using binge_drinker_fun() to generate binge_drinker with user-inputted
# values.
#
# Let's say you are a male, and you had drinks in the last week. Let's say
# you had 3 drinks on Sunday, 1 drink on
# Monday, 6 drinks on Tuesday, 0 drinks on Wednesday, 3 drinks on Thurday,
# 8 drinks on Friday, and 2 drinks on Saturday. Using binge_drinker_fun(),
# we can check if you would be classified as a drinker.

binge <- binge_drinker_fun(DHH_SEX = 1, ALW_1 = 1, ALW_2A1 = 3, ALW_2A2 = 1,
                           ALW_2A3 = 6, ALW_2A4 = 0, ALW_2A5 = 3,
                           ALW_2A6 = 8, ALW_2A7 = 2)

print(binge)
```

---

bmi_fun *Body Mass Index (BMI) derived variable*

Description

This function creates a harmonized BMI variable. The BMI variable provided by the CCHS calculates BMI using methods that vary across cycles, leading to measurement error when using multiple

CCHS cycles. In certain CCHS cycles (2001-2003, 2007+), there are age restrictions in which respondents under the age of 20 and over the age of 64 were not included. Across all CCHS cycles, female respondents who identified as being pregnant were excluded; and in certain CCHS cycles (2003-2007, 2013-2014), females who did not answer the pregnancy question were coded as NS (not stated) for HWTGBMI. As well, in certain CCHS cycles (2001-2003, 2009-2014), respondents outside certain height and weight ranges (0.914-2.108m for height, 0-260kg for weight) were excluded from HWTGBMI.

bmi_fun() creates a derived variable (HWTGBMI_der) that is harmonized across all CCHS cycles. This function divides weight by the square of height.

Usage

```
bmi_fun(HWTGHTM, HWTGWTK)
```

Arguments

HWTGHTM	CCHS variable for height (in meters)
HWTGWTK	CCHS variable for weight (in kilograms)

Details

For HWTGBMI_der, there are no restrictions to age, height, weight, or pregnancy status. While pregnancy was consistent across all CCHS cycles, its variable (MAM_037) was not available in the PUMF CCHS datasets so it could not be harmonized and included into the function.

For any single CCHS survey year, it is appropriate to use the CCHS BMI variable (HWTGBMI) that is also available on cchsflow. HWTGBMI_der is recommended when using multiple survey cycles.

HWTGBMI_der uses the CCHS variables for height and weight that have been transformed by cchsflow. In order to generate a value for BMI across CCHS cycles, height and weight must be transformed and harmonized.

Value

numeric value for BMI in the HWTGBMI_der variable

Note

In earlier CCHS cycles (2001 and 2003), height was collected in inches; while in later CCHS cycles (2005+) it was collected in meters. To harmonize values across cycles, height was converted to meters (to 3 decimal points). Weight was collected in kilograms across all CCHS cycles, so no transformations were required in the harmonization process.

Examples

```
# Using bmi_fun() to create BMI values between cycles
# bmi_fun() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform the derived BMI variable, use rec_with_table() for each cycle
```

```
# and specify HWTGBMI_der, along with height (HWTGHTM) and weight (HWTGWTK).
# Then by using merge_rec_data(), you can combined HWTGBMI_der across
# cycles.

library(cchsflow)
bmi2001 <- rec_with_table(
  cchs2001_p, c(
    "HWTGHTM",
    "HWTGWTK", "HWTGBMI_der"
  )
)

head(bmi2001)

bmi2011_2012 <- rec_with_table(
  cchs2011_2012_p, c(
    "HWTGHTM",
    "HWTGWTK", "HWTGBMI_der"
  )
)

tail(bmi2011_2012)

combined_bmi <- merge_rec_data(bmi2001, bmi2011_2012)
head(combined_bmi)
tail(combined_bmi)

# Using bmi_fun() to generate a BMI value with user inputted height and
# weight values. bmi_fun() can also generate a value for BMI if you input a
# value for height and weight. Let's say your height is 170cm (1.7m) and
# your weight is 50kg, your BMI can be calculated as follows:

library(cchsflow)
BMI <- bmi_fun(HWTGHTM = 1.7, HWTGWTK = 50)
print(BMI)
```

---

bmi_fun_cat *Categorical BMI (international standard)*

Description

This function creates a categorical derived variable (HWTGBMI_der_cat4) that categorizes derived BMI (HWTGBMI_der).

Usage

```
bmi_fun_cat(HWTGBMI_der)
```

Arguments

HWTGBMI_der	derived variable that calculates numeric value for BMI. See bmi_fun for documentation on how variable was derived.

Details

The categories were based on international standards and are divided into four categories: underweight for BMI < 18.5 (1), normal weight for BMI between 18.5 to 25 (2), overweight for BMI between 25 to 30 (3), and obese for BMI over 30 (4).

HWTGBMI_der_cat4 uses the derived variable HWTGBMI_der. HWTGBMI_der uses height and weight that have been transformed by cchsflow. In order to categorize BMI across CCHS cycles, height and weight variables must be transformed and harmonized.

Value

value for BMI categories in the HWTGBMI_der_cat4 variable.

Examples

```
# Using bmi_fun_cat() to categorize BMI across CCHS cycles
# bmi_fun_cat() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform HWTGBMI_der_cat4 across all cycles, use rec_with_table() for
# each CCHS cycle.
# Since HWTGBMI_der is also a derived variable, you will have to specify
# the variables that are derived from it.

library(cchsflow)

bmi_cat_2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "HWTGHTM",
    "HWTGWTK",
    "HWTGBMI_der",
    "HWTGBMI_der_cat4"
  )
)

head(bmi_cat_2009_2010)

bmi_cat_2011_2012 <- rec_with_table(
  cchs2011_2012_p,c(
    "HWTGHTM",
    "HWTGWTK",
    "HWTGBMI_der",
    "HWTGBMI_der_cat4"
  )
)
```

```
tail(bmi_cat_2011_2012)

combined_bmi_cat <- suppressWarnings(merge_rec_data
(bmi_cat_2009_2010,bmi_cat_2011_2012))

head(combined_bmi_cat)
tail(combined_bmi_cat)
```

`cchs2001_p` *2001 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2001 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-82M0013-E-2001-c1-1-general-file

DDI: https://osf.io/jtd9h/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

`cchs2001_p` a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=3359

Examples

```
data(cchs2001_p)
str(cchs2001_p)
```

cchs2003_p *2003 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2003 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-82M0013-E-2003-c2-1-General File

DDI: https://osf.io/nzq37/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2003_p a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=4995

Examples

```
data(cchs2003_p)
str(cchs2003_p)
```

cchs2005_p *2005 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2005 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-82M0013-E-2005-c3-1-main-file

DDI: https://osf.io/35mhq/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2005_p a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=22642

Examples

```
data(cchs2005_p)
str(cchs2005_p)
```

---

cchs2007_2008_p *2007-2008 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2007-2008 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-E-2007-2008-AnnualComponent

DDI: https://osf.io/emzsp/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2007_2008_p
a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=29539

Examples

```
data(cchs2007_2008_p)
str(cchs2007_2008_p)
```

---

cchs2009_2010_p *2009-2010 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2009-2010 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: CCHS-82M0013-E-2009-2010-Annualcomponent

DDI: https://osf.io/ynzpe/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2009_2010_p
: a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=67251

Examples

```
data(cchs2009_2010_p)
str(cchs2009_2010_p)
```

cchs2010_p *2010 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2010 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

NOTE: this subset of respondents may also be in the 2009-2010 PUMF subset. Please see the "CCHS datasets that overlap each other" article to see how the two datasets contain overlap.

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: CCHS-82M0013-E-2010-AnnualComponent

DDI: https://osf.io/7stpz/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2010_p a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=81424

Examples

```
data(cchs2010_p)
str(cchs2010_p)
```

cchs2011_2012_p *2011-2012 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2011-2012 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-82M0013-E-2011-2012-Annual-component

DDI: https://osf.io/zk2vw/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2011_2012_p
: a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=114112

Examples

```
data(cchs2011_2012_p)
str(cchs2011_2012_p)
```

---

cchs2012_p *2012 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2012 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

NOTE: this subset of respondents may also be in the 2011-2012 PUMF subset. Please see the "CCHS datasets that overlap each other" article to see how the two datasets contain overlap.

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-82M0013-E-2012-Annual-component

DDI: https://osf.io/sbem8/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2012_p
: a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=135927

Examples

```
data(cchs2012_p)
str(cchs2012_p)
```

---

cchs2013_2014_p *2013-2014 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2013-2014 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-82M0013-E-2013-2014-Annual-component

DDI: https://osf.io/gy25d/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2013_2014_p
: a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=144170

Examples

```
data(cchs2013_2014_p)
str(cchs2013_2014_p)
```

cchs2014_p *2014 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2014 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

NOTE: this subset of respondents may also be in the 2013-2014 PUMF subset. Please see the "CCHS datasets that overlap each other" article to see how the two datasets contain overlap.

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Jan 2020. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-82M0013-E-2014-Annual-component

DDI: https://osf.io/tbmdn/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2014_p a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=164081

Examples

```
data(cchs2014_p)
str(cchs2014_p)
```

cchs2015_2016_p *2015-2016 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2015-2016 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

NOTE: this subset of respondents may also be in the 2015-2016 PUMF subset. Please see the "CCHS datasets that overlap each other" article to see how the two datasets contain overlap.

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Oct 2021. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-82M0013-E-2015-2016-Annual-component

DDI: https://osf.io/m948q/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

`cchs2015_2016_p`
: a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=238854

Examples

```
data(cchs2015_2016_p)
str(cchs2015_2016_p)
```

---

cchs2017_2018_p *2017-2018 CCHS PUMF subset data (200 respondents)*

Description

This is a subset of 200 observations from the 2017-2018 cycle of the Canadian Community Health Survey (CCHS) Public Use Microdata file (PUMF) dataset. The CCHS survey is conducted by Statistics Canada.

Details

NOTE: this subset of respondents may also be in the 2017-2018 PUMF subset. Please see the "CCHS datasets that overlap each other" article to see how the two datasets contain overlap.

See here for the open license. Source from Statistics Canada, Canadian Community Health Survey PUMF, accessed Oct 2021. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada.

Long name: cchs-82M0013-E-2017-2018-Annual-component

DDI: https://osf.io/q8g7y/

Additional documentation (PDFs): https://osf.io/hkuy3/

Value

cchs2017_2018_p
: a data frame

Source

https://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=329241

Examples

```
data(cchs2017_2018_p)
str(cchs2017_2018_p)
```

---

compare_value_based_on_interval

Compare Value Based On Interval

Description

Compare values on the scientific notation interval

Usage

```
compare_value_based_on_interval(
  left_boundary,
  right_boundary,
  data,
  compare_columns,
  interval
)
```

Arguments

left_boundary	the min value
right_boundary	the max value
data	the data that contains values being compared
compare_columns	The columns inside data being checked
interval	The scientific notation interval

Value

a boolean vector containing true for rows where the comparison is true

COPD_Emph_der_fun1 *COPD_Emph_der_fun1*

Description

This is one of 2 functions used to create a derived variable (COPD_Emph_der) that determines if a respondents has either COPD or Emphysema. 2 different functions have been created to account for the fact that different respiratory variables are used across CCHS cycles. This function is for CCHS cycles (2005-2008) that use COPD and Emphysema as a combined variable.

Usage

```
COPD_Emph_der_fun1(DHHGAGE_cont, CCC_91E, CCC_91F)
```

Arguments

DHHGAGE_cont	continuous age variable.
CCC_91E	variable indicating if respondent has Emphysema
CCC_91F	variable indicating if respondent has COPD

Value

a categorical variable (COPD_Emph_der) with 3 levels:

1. respondent is over the age of 35 and has a respiratory condition
2. respondent is under the age of 35 and has a respiratory condition
3. respondent does not have a respiratory condition

See Also

COPD_Emph_der_fun2

Examples

```
# COPD_Emph_der_fun1() to create values across CCHS cycles
# (2005-2008) COPD_Emph_der_fun1() is specified in
# variable_details.csv along with the CCHS variables and cycles included.

# To transform COPD_Emph_der, use rec_with_table() for each CCHS cycle
# and specify COPD_Emph_der, along with the various respiratory
# variables. Then by using merge_rec_data() you can combine COPD_Emph_der
# across cycles.

library(cchsflow)

COPD2005 <- suppressWarnings(rec_with_table(
  cchs2005_p,  c(
    "DHHGAGE_cont", "CCC_91E", "CCC_91F",
```

```
    "COPD_Emph_der"
  )
))

head(COPD2005)

COPD2007_2008 <- suppressWarnings(rec_with_table(
  cchs2007_2008_p, c(
    "DHHGAGE_cont", "CCC_91E", "CCC_91F",
    "COPD_Emph_der"
  )
))

tail(COPD2007_2008)

combined_COPD <- suppressWarnings(merge_rec_data(COPD2005, COPD2007_2008))

head(combined_COPD)
tail(combined_COPD)
```

`COPD_Emph_der_fun2` *COPD_Emph_der_fun2*

Description

This is one of 2 functions used to create a derived variable (COPD_Emph_der) that determines if a respondents has either COPD or Emphysema. 2 different functions have been created to account for the fact that different respiratory variables are used across CCHS cycles. This function is for CCHS cycles (2001-2003, 2009-2014) that use COPD and Emphysema as a combined variable.

Usage

```
COPD_Emph_der_fun2(DHHGAGE_cont, CCC_091)
```

Arguments

`DHHGAGE_cont`	continuous age variable.
`CCC_091`	variable indicating if respondent has either COPD or Emphysema

Value

a categorical variable (COPD_Emph_der) with 3 levels:

1. respondent is over the age of 35 and has a respiratory condition
2. respondent is under the age of 35 and has a respiratory condition
3. respondent does not have a respiratory condition

See Also

COPD_Emph_der_fun2

Examples

```
# COPD_Emph_der_fun2() to create values across CCHS cycles
# (2001-2003, 2009-2014) COPD_Emph_der_fun2() is specified in
# variable_details.csv along with the CCHS variables and cycles included.

# To transform COPD_Emph_der, use rec_with_table() for each CCHS cycle
# and specify COPD_Emph_der, along with the various respiratory
# variables. Then by using merge_rec_data() you can combine COPD_Emph_der
# across cycles.

library(cchsflow)

COPD2001 <- suppressWarnings(rec_with_table(
  cchs2001_p,  c(
    "DHHGAGE_cont", "CCC_091",
    "COPD_Emph_der"
  )
))

head(COPD2001)

COPD2014 <- suppressWarnings(rec_with_table(
  cchs2007_2008_p, c(
    "DHHGAGE_cont", "CCC_091",
    "COPD_Emph_der"
  )
))

tail(COPD2014)

combined_COPD <- suppressWarnings(merge_rec_data(COPD2001, COPD2014))

head(combined_COPD)
tail(combined_COPD)
```

---

diet_score_fun *Diet score*

Description

This function creates a derived diet variable (diet_score) based on consumption of fruit, salad, potatoes, carrots, other vegetables and juice. 2 baseline points plus summation of total points for diet attributes. Negative overall scores are recoded to 0, resulting in a range from 0 to 10.

- 1 point per daily fruit and vegetable consumption, excluding fruit juice (maximum 8 points).

- -2 points for high potato intake (>=7 (males), >=5 (females) times/week)
- -2 points for no carrot intake
- -2 points per daily frequency of fruit juice consumption greater than once/day (maximum -10 points)

Usage

```
diet_score_fun(FVCDFRU, FVCDSAL, FVCDPOT, FVCDCAR, FVCDVEG, FVCDJUI, DHH_SEX)
```

Arguments

FVCDFRU	daily consumption of fruit
FVCDSAL	daily consumption of green salad
FVCDPOT	daily consumption of potatoes
FVCDCAR	daily consumption of carrots
FVCDVEG	daily consumption of other vegetables
FVCDJUI	daily consumption of fruit juice
DHH_SEX	sex; 1 = male, 2 = female

Details

While diet score can be calculated for all survey respondents, in the 2005 CCHS survey cycle, fruit and vegetable consumption was an optional section in which certain provinces had opted in to be asked to respondents. In this survey cycle, fruit and vegetable consumption was asked to respondents in British Columbia, Ontario, Alberta, and Prince Edward Island. As such, diet score has a large number of missing respondents for this cycle.

Examples

```
# Using the diet_score_fun function to create the derived diet variable
# across CCHS cycles.
# diet_score_fun() is specified in the variable_details.csv.

# To create a harmonized diet_score variable across CCHS cycles, use
# rec_with_table() for each CCHS cycle and specify diet_score_fun and the
# required base variables.
# Using merge_rec_data(), you can combine diet_score across cycles.

library(cchsflow)

diet_score2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "FVCDFRU", "FVCDSAL", "FVCDPOT", "FVCDCAR", "FVCDVEG", "FVCDJUI",
    "DHH_SEX", "diet_score"
  )
)

head(diet_score2009_2010)
```

```
diet_score2011_2012 <- rec_with_table(
  cchs2011_2012_p,c(
    "FVCDFRU", "FVCDSAL", "FVCDPOT", "FVCDCAR", "FVCDVEG", "FVCDJUI",
    "DHH_SEX", "diet_score"
  )
)

tail(diet_score2011_2012)

combined_diet_score <- suppressWarnings(merge_rec_data(diet_score2009_2010,
 diet_score2011_2012))

head(combined_diet_score)
tail(combined_diet_score)
```

`diet_score_fun_cat` *Categorized diet score*

Description

This function creates a categorical derived diet variable (diet_score_cat3) that categorizes derived diet score (diet_score).

Usage

```
diet_score_fun_cat(diet_score)
```

Arguments

`diet_score`	derived variable that calculates diet score. See `diet_score_fun` for documentation on how variable was derived.

Details

The diet score is based on consumption of fruit, salad, potatoes, carrots, other vegetables and juice. 2 baseline points plus summation of total points for diet attributes. Negative overall scores are recoded to 0, resulting in a range from 0 to 10.The categories were based on the Mortality Population Risk Tool (Douglas Manuel et al. 2016).

diet_score_cat3 uses the derived variable diet_score. diet_score uses sex, and fruit and vegetable variables that have been transformed by cchsflow (see documentation on diet_score). In order to categorize diet across CCHS cycles, sex, and fruit and vegetable variables must be transformed and harmonized.

Value

value for diet score categories using diet_score_cat3 variable.

Examples

```
# Using the diet_score_fun_cat function to categorize the derived diet
# variable across CCHS cycles.
# diet_score_fun_cat() is specified in the variable_details.csv.

# To create a harmonized diet_score_cat3 variable across CCHS cycles, use
# rec_with_table() for each CCHS cycle.
# Since diet_score is also a derived variable, you will have to specify
# the variables that are derived from it.
# Using merge_rec_data(), you can combine diet_score_cat3 across cycles.

library(cchsflow)

diet_score_cat2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "FVCDFRU", "FVCDSAL", "FVCDPOT", "FVCDCAR", "FVCDVEG", "FVCDJUI",
    "DHH_SEX", "diet_score", "diet_score_cat3"
  )
)

head(diet_score_cat2009_2010)

diet_score_cat2011_2012 <- rec_with_table(
  cchs2011_2012_p,c(
    "FVCDFRU", "FVCDSAL", "FVCDPOT", "FVCDCAR", "FVCDVEG", "FVCDJUI",
    "DHH_SEX", "diet_score", "diet_score_cat3"
  )
)

tail(diet_score_cat2011_2012)

combined_diet_score_cat <- suppressWarnings(merge_rec_data(
diet_score_cat2009_2010, diet_score_cat2011_2012))

head(combined_diet_score_cat)
tail(combined_diet_score_cat)
```

DPSDPP *Depression Scale - Predicted Probability*

Description

NOTE: this is not a function.

This is categorical variable derived by Statistics Canada that predicts the probability that a respondent would be diagnosed as having a major depressive episode if a diagnostic interview was completed. This variable is derived from DPSDSF in which probabilities are assigned to respondents based on their depression scale score. For more details on how the variable was derived click here.

Usage

```
DPSDPP(DPSDPP)
```

Arguments

DPSDPP	cchsflow variable name for derived depression scale predicted probability.

Details

While this variable was considered to be categorical in CCHS documentation, the values range from 0 to 0.90 with no distinct names or metadata for each category. As such, this variable was specified as a continuous variable in cchsflow. This has no bearing on the final output of the variable as there are no recode changes. This means that a respondent who was coded with a probability of 0.50 will still have a probability value of 0.50 when the variable goes through harmonization.

This variable is present in every CCHS cycle used in cchsflow, and how it was derived remains consistent.

See Also

DPSDSF

Examples

```
library(cchsflow)
 ?DPSDPP
```

DPSDSF *Derived Depression Scale - Short Form Score*

Description

NOTE: this is not a function.

This is a continuous variable derived by Statistics Canada that assesses the level of depression of respondents who have identified that they have felt depressed or loss of interest within the last two weeks. This variable is scaled from 0 to 8, with 0 indicating a respondent has not felt depressed or loss of interest, and 8 representing the highest level of depression.

Usage

```
DPSDSF(DPSDSF)
```

Arguments

DPSDSF	cchsflow variable name for derived depression scale.

Details

The derivation of this variable is based on the work of Kessler & Mroczek from the University of Michigan. For more details on the items used and how the variable was derived click here.

This variable is present in every CCHS cycle used in cchsflow, and how it was derived remains consistent.

See Also

DPSDPP

Examples

```
library(cchsflow)
 ?DPSDSF
```

`energy_exp_fun` *Daily energy expenditure in leisure activity*

Description

This function creates a derived variable for A MET is a conceptual value that represents energy expended during physical activity. The volume of activity is calculated by multiplying the amount of minutes of activity (by level of intensity) by the MET value associated with that intensity. A MET (metabolic equivalent) is the energy cost of activity expressed as kilocalories expended per kilogram of body weight per hour of activity.

In CCHS 2001-2014, PACDEE is the variable used to determine the daily expenditure of leisure activity for all ages. In CCHS 2015-2018, ages 12-17 and 18+ years old have separate activity variables, where 12-17 year olds use PAY_XXX and 18+ year olds use PAA_XXX. Leisure activity is not directly measured. We used the derived variable, PAADVVOL, and removed active transportation in the new function. With this function, we combined leisure activity for ages 12+. We calculate the daily energy expenditure which uses the frequency and duration per session of the physical activity as well as the MET value (3 METS for leisure and 6 METS for vigorous activity).

EE (Daily Energy Expenditure) = ((N X D X METvalue) / 60)/7 Where: N = the number of times a respondent engaged in an activity over a 7 day period D = the average duration in minutes of the activity MET value = the energy cost of the activity expressed as kilocalories expended per kilogram of body weight per hour of activity (kcal/kg per hour)

Usage

```
energy_exp_fun(
  DHHGAGE_cont,
  PAA_045,
  PAA_050,
  PAA_075,
  PAA_080,
```

```
  PAADVDYS,
  PAADVVIG,
  PAYDVTOA,
  PAYDVADL,
  PAYDVVIG,
  PAYDVDYS
)
```

Arguments

DHHGAGE_cont	continuous age variable.
PAA_045	number of hours of sports, fitness, or recreational activity that make you sweat or breathe harder for CCHS 2015-2018 for 18+ years old.
PAA_050	number of minutes of sports, fitness, or recreational activity that make you sweat or breathe harder for CCHS 2015-2018 for 18+ years old.
PAA_075	number of hours of other physical activity while at work, home or volunteering for CCHS 2015-2018 for 18+ years old.
PAA_080	number of minutes of other physical activity while at work, home or volunteering for CCHS 2015-2018 for 18+ years old.
PAADVDYS	number of active days - 7 day for CCHS 2015-2018 for 18+ years old.
PAADVVIG	number of minutes of vigorous activity over 7 days or CCHS 2015-2018 for 18+ years old.
PAYDVTOA	total minutes of other activities - 7 day for CCHS 2015-2018 for 12-17 years old.
PAYDVADL	total minutes of physical activity - leisure - 7 day for CCHS 2015-2018 for 12-17 years old.
PAYDVVIG	total minutes - vigorous physical activity - 7 d for CCHS 2015-2018 for 12-17 years old.
PAYDVDYS	total days physically active - 7 day for CCHS 2015-2018 for 12-17 years old.

Value

Continuous variable for energy expenditure (energy_exp)

Examples

```
# Using energy_exp_fun() to create energy expenditure values across CCHS
# cycles
# energy_exp_fun() is specified in variable_details.csv along with the CCHS
# variables and cycles included.

# To transform energy_exp across cycles, use rec_with_table() for each
# CCHS cycle and specify energy_exp, along with each activity variable.
# Then by using merge_rec_data(), you can combine energy_exp across
# cycles

library(cchsflow)
```

```
energy_exp2015_2016 <- rec_with_table(
  cchs2015_2016_p, c(
    "DHHGAGE_cont", "PAA_045", "PAA_050", "PAA_075", "PAA_080", "PAADVDYS",
    "PAADVVIG", "PAYDVTOA", "PAYDVADL", "PAYDVVIG", "PAYDVDYS", "energy_exp"
  )
)

head(energy_exp2015_2016)

energy_exp2017_2018 <- rec_with_table(
  cchs2017_2018_p, c(
    "DHHGAGE_cont", "PAA_045", "PAA_050", "PAA_075", "PAA_080", "PAADVDYS",
    "PAADVVIG", "PAYDVTOA", "PAYDVADL", "PAYDVVIG", "PAYDVDYS", "energy_exp"
  )
)

tail(energy_exp2015_2016)

combined_energy_exp <- suppressWarnings(merge_rec_data(energy_exp2015_2016,
 energy_exp2017_2018))

head(combined_energy_exp)
tail(combined_energy_exp)
```

`food_insecurity_der` *Food insecurity*

Description

NOTE: this is not a function.

This is a derived variable that uses the different food insecurity variables from all CCHS cycles to generate food_insecurity_der that is harmonized across all cycles. food_insecurity_der is a categorical variable with two categories:

1. no food insecurity in the last 12 months
2. food insecurity in the last 12 months

Usage

```
food_insecurity_der(FINF1, FSCDHFS, FSCDHFS2)
```

Arguments

FINF1	variable used in 2001 and 2003 survey cycles indicating food insecurity in the past 12 months
FSCDHFS	variable used in the 2005 survey cycle measuring food insecurity & hunger in the last 12 months
FSCDHFS2	variable used in 2007-2014 survey cycles measuring household food insecurity in the last 12 months

Details

Food insecurity is measured differently across CCHS cycles. In 2001 and 2003, FINF1 is used; in 2005, FSCDHFS is used; and in 2007 to 2014, FSCDHFS2 is used. Each variable examines food insecurity in the household over the past 12 months, but use different base variables to derive food insecurity.

If you are using cchsflow for CCHS survey years that use consistent food insecurity variables, it is appropriate to use FINF1, FSCDHFS, or FSCDHFS2 that are available on cchsflow. If you are using cchsflow for only the 2001 and 2003 cycles, it is appropriate to use FINF1. If you are using cchsflow for only the 2005 cycle, FSCDHFS is appropriate. If you are using cchsflow for cycles between 2007 and 2014, FSCDHFS2 is appropriate. For multiple CCHS survey years that do not use the same food insecurity variables (i.e. using cchsflow for years 2001 to 2007), food_insecurity_der is recommended.

Examples

```
library(cchsflow)
?food_insecurity_der
```

GEN_02A2 *Satisfaction with life (GEN_02A/GEN_02A2)*

Description

NOTE: this is not a function.

These are two variables asked in the CCHS that asks respondents to rate their satisfaction with their lives. The variable GEN_02A is a categorical variable with 5 categories:

1. Very satisfied
2. Satisfied
3. Neither satisfied nor unsatisfied
4. Dissatisfied
5. Very dissatisfied

The GEN_02A2 is a continuous variable from 0 to 10, where 0 represents very dissatisfied and 10 represents very satisfied.

Usage

```
GEN_02A2(GEN_02A, GEN_02A2)
```

Arguments

GEN_02A	- categorical life satisfaction variable asked from 2003-2007
GEN_02A2	- continuous life satisfaction variable asked from 2009-2014, and derived for 2003-2007

Details

GEN_02A was asked to respondents in the 2003, 2005, and 2007-2008 CCHS survey cycles; while GEN_02A2 was asked to respondents in CCHS survey cycles from 2009 to 2014. To harmonize GEN_02A2 across more cycles, GEN_02A2 was derived for earlier cycles by converting GEN_02A values to match the scale used in GEN_02A2. The very satisfied category was converted to a score of 10; the satisfied category was converted to a score of 7; the neither satisfied nor unsatisfied category was converted to a score of 5; the dissatisfied category was converted to a score of 2; and the very dissatisfied category was converted to a score of 0.

When using earlier CCHS cycles (2003-2007), it is appropriate to use GEN_02A. When using multiple CCHS cycles that include cycles from 2009-2014, GEN_02A2 is recommended.

Examples

```
library(cchsflow)
?GEN_02A2
```

---

get_data_variable_name

Get Data Variable Name

Description

Retrieves the name of the column inside data to use for calculations

Usage

```
get_data_variable_name(
  data_name,
  data,
  row_being_checked,
  variable_being_checked
)
```

Arguments

data_name	name of the database being checked
data	database being checked
row_being_checked	the row from variable details that contains information on this variable
variable_being_checked	the name of the recoded variable

Value

the data equivalent of variable_being_checked

`if_else2` *if_else2*

Description

Custom ifelse function that evaluates missing (NA) values. If the logical argument (x) compares to a value that is 'NA', it is set to 'FALSE'

Usage

```
if_else2(x, a, b)
```

Arguments

x	A logical argument
a	value if 'x' is 'TRUE'
b	value if 'x' is 'FALSE'

Details

unlike the base ifelse() function, if_else2() is able to evaluate NA as either a or b. In base ifelse(), anything compared to NA will produce NA, which can break a function. When dealing with large datasets like the CCHS, there are many missing (NA) values. That means a special ifelse function like if_else2() is needed in order for other functions to not break

Value

a or b based on the evaluation of x

Examples

```
age <- 12
status <- if_else2((age < 18), "child", "invalid age")
print(status)

age <- NA
status <- if_else2((age < 18), "child", "invalid age")
print(status)
```

`immigration_fun` *Immigration by ethnicity and settlement*

Description

This function creates a categorical variable based on immigrant status (SDCFIMM), country of birth (SDCGCBG), ethnicity (SDCGCGT), and time in Canada (SDCGRES).

Usage

```
immigration_fun(SDCFIMM, SDCGCBG, SDCGCGT, SDCGRES)
```

Arguments

SDCFIMM	Immigrant status (1-immigrant, 2-non-immigrant)
SDCGCBG	Country of birth (1-Canada, 2-Outside of Canada)
SDCGCGT	Cultural or racial origin (1-white, 2-visible minority)
SDCGRES	Length/time in Canada since immigration (1- 0-9 years, 2- 10+ years)

Details

immigration_der uses the CCHS variables that have been transformed by cchsflow. In order to generate a value for BMI across CCHS cycles, the following SDC variables must be transformed and harmonized.

Value

Categorical variable (immigration_der) with six categories:

- 1 - White Canada-born
- 2 - Non-white Canadian born
- 3 - White immigrant born outside of Canada (0-9 years in Canada)
- 4 - Non-white immigrant born outside of Canada (0-9 years in Canada)
- 5 - White immigrant born outside of Canada (10+ years in Canada)
- 6 - Non-white immigrant born outside of Canada (10+ years in Canada)

Examples

```
# Using immigration_fun() to create immigration_der values across CCHS cycles
# immigration_fun() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform immigration_der, use rec_with_table() for each CCHS cycle
# and specify immigration_der, along with the various SDC variables.
# Then by using merge_rec_data() you can combine immigration_der across cycles.
```

```
library(cchsflow)
immigration2001 <- rec_with_table(
  cchs2001_p, c(
    "SDCFIMM", "SDCGCBG", "SDCGCGT", "SDCGRES", "immigration_der"
  )
)

head(immigration2001)

immigration2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "SDCFIMM", "SDCGCBG", "SDCGCGT", "SDCGRES", "immigration_der"
  )
)

tail(immigration2009_2010)

combined_immigration <- merge_rec_data(immigration2001, immigration2009_2010)

head(combined_immigration)

tail(combined_immigration)
```

`is_equal` *is equal*

Description

Function to compare even with NA present This function returns TRUE wherever elements are the same, including NA's, and false everywhere else.

Usage

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

Arguments

v1	variable 1
v2	variable 2

Value

boolean value of whether or not v1 and v2 are equal

Examples

```
library(cchsflow)
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` *label_data*

Description

Attaches labels to the DataToLabel 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

LBFA_31A *Occupation Group (9 categories)*

Description

NOTE: this is not a function.

This is a 9 category variable (LBFA_31A) that is in the CCHS that asks which occupation group best describes a respondent. Occupation group is asked in the 2001 CCHS cycle and in CCHS cycles from 2007-2014.

Usage

```
LBFA_31A(LBFA_31A)
```

Arguments

LBFA_31A	cchsflow variable name for Occupation Group (9 categories)

Details

While occupation group is asked in many survey cycles, the 2001 CCHS survey cycle is the only survey that has 9 categories. The categories are as follows:

1. Management
2. Professional (including accountants)
3. Technologist, Technician or Tech Occupation
4. Administrative, Financial or Clerical
5. Sales or Service
6. Trades, Transport or Equipment Operator
7. Farming, Forestry, Fishing, Mining
8. Processing, Manufacturing, Utilities
9. Other

To harmonize the 2001 CCHS cycle with other survey cycles, LBFA_31A_a and LBFA_31A_b were created in which categories in the 2001 survey cycle were collapsed.

See Also

LBFA_31A_a, LBFA_31A_b

Examples

```
library(cchsflow)
 ?LBFA_31A
```

LBFA_31A_a *Occupation Group (5 categories)*

Description

NOTE: this is not a function.

This is a 5 category variable (LBFA_31A_a) that is in the CCHS that asks which occupation group best describes a respondent. Occupation group is asked in the 2001 CCHS cycle and in CCHS cycles from 2007-2014.

Usage

```
LBFA_31A_a(LBFA_31A_a)
```

Arguments

LBFA_31A_a	cchsflow variable name for Occupation Group (5 categories)

Details

In the 2007-2014 CCHS survey cycles, occupation group has 5 categories. The categories are as follows:

1. Management, Health, Education, Art, Culture
2. Business, Finance, Admin
3. Sales or Service
4. Trades, Transport or Equipment Operator
5. Unique to Primary Industry/Processing/Manufacturing

In this variable, categories from the 2001 CCHS survey cycle were collapsed to harmonize with the other survey cycles. "Management, Professional (including accountants), Technologist, Technician or Tech Occupation" were combined into one category "Management, Health, Education, Art, Culture". "Farming, Forestry, Fishing, Mining" and "Processing, Manufacturing, Utilities", were combined into one category "Farming, Forestry, Fishing, Mining, Processing, Manufacturing, Utilities".

The "other" category in the 2001 CCHS survey cycle was assigned to missing (NA(b)). This is consistent with other studies that group the "other" category as "missing". LBFA_31A_b is a 6 category variable that keeps the "other" category in the 2001 survey cycle as "other".

See Also

LBFA_31A, LBFA_31A_b

Examples

```
library(cchsflow)
?LBFA_31A_a
```

LBFA_31A_b *Occupation Group (6 categories)*

Description

NOTE: this is not a function.

This is a 6 category variable (LBFA_31A_b) that is in the CCHS that asks which occupation group best describes a respondent. Occupation group is asked in the 2001 CCHS cycle and in CCHS cycles from 2007-2014.

Usage

```
LBFA_31A_b(LBFA_31A_b)
```

Arguments

LBFA_31A_b	cchsflow variable name for Occupation Group (6 categories)

Details

In the 2007-2014 CCHS survey cycles, occupation group has 5 categories. This variable, however, includes a sixth category to account for the "other" category asked in the 2001 CCHS survey cycle. The categories are as follows:

1. Management, Health, Education, Art, Culture
2. Business, Finance, Admin
3. Sales or Service
4. Trades, Transport or Equipment Operator
5. Unique to Primary Industry/Processing/Manufacturing
6. Other

In this variable, categories from the 2001 CCHS survey cycle were collapsed to harmonize with the other survey cycles. "Management, Professional (including accountants), Technologist, Technician or Tech Occupation" were combined into one category "Management, Health, Education, Art, Culture". "Farming, Forestry, Fishing, Mining" and "Processing, Manufacturing, Utilities", were combined into one category "Farming, Forestry, Fishing, Mining, Processing, Manufacturing, Utilities".

See Also

LBFA_31A, LBFA_31A_a

Examples

```
library(cchsflow)
?LBFA_31A_b
```

`low_drink_long_fun` *Long term risks due to drinking*

Description

This function creates a categorical variable that flags for increased long term health risks due to their drinking habits, according to Canada's Low-Risk Alcohol Drinking Guideline.

Usage

```
low_drink_long_fun(
  DHH_SEX,
  ALWDWKY,
  ALC_1,
  ALW_1,
  ALW_2A1,
  ALW_2A2,
  ALW_2A3,
  ALW_2A4,
  ALW_2A5,
  ALW_2A6,
  ALW_2A7
)
```

Arguments

`DHH_SEX`	Sex of respondent (1 - male, 2 - female)
`ALWDWKY`	Number of drinks consumed in the past week
`ALC_1`	Drinks in the past year (1 - yes, 2 - no)
`ALW_1`	Drinks in the last week (1 - yes, 2 - no)
`ALW_2A1`	Number of drinks on Sunday
`ALW_2A2`	Number of drinks on Monday
`ALW_2A3`	Number of drinks on Tuesday
`ALW_2A4`	Number of drinks on Wednesday
`ALW_2A5`	Number of drinks on Thursday
`ALW_2A6`	Number of drinks on Friday
`ALW_2A7`	Number of drinks on Saturday

Details

The classification of drinkers according to their long term health risks comes from guidelines in Alcohol and Health in Canada: A Summary of Evidence and Guidelines for Low-risk Drinking, and is based on the alcohol consumption reported over the past week. Short-term or acute risks include injury and overdose.

Categories are based on CCHS 2015-2016's variable (ALWDVLTR) where long term health risk are increased when drinking more than 10 drinks a week for women, with no more than 2 drinks a day most days, and more than 15 drinks a week for men, with no more than 3 drinks a day most days.

See https://osf.io/ykau5/ for more details on the guideline. See https://osf.io/ycxaq/ for more details on the derivation of the function on page 8.

Value

Categorical variable (ALWDVLTR_der) with two categories:

- 1 - Increased long term health risk
- 2 - No increased long term health risk

Examples

```
# Using low_drink_long_fun() to create ALWDVLTR_der values across CCHS cycles
# low_drink_long_fun() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform ALWDVLTR_der, use rec_with_table() for each CCHS cycle
# and specify ALWDVLTR_der, along with the various alcohol and sex
# variables.
# Using merge_rec_data(), you can combine ALWDVLTR_der across cycles.

library(cchsflow)
long_low_drink2001 <- rec_with_table(
  cchs2001_p, c(
    "ALW_1", "DHH_SEX", "ALW_2A1", "ALW_2A2", "ALW_2A3", "ALW_2A4",
    "ALW_2A5", "ALW_2A6", "ALW_2A7", "ALWDWKY", "ALC_1","ALWDVLTR_der"
  )
)

head(long_low_drink2001)

long_low_drink2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "ALW_1", "DHH_SEX", "ALW_2A1", "ALW_2A2", "ALW_2A3", "ALW_2A4",
    "ALW_2A5", "ALW_2A6", "ALW_2A7", "ALWDWKY", "ALC_1","ALWDVLTR_der"
  )
)

tail(long_low_drink2009_2010)

combined_long_low_drink <- bind_rows(long_low_drink2001,
long_low_drink2009_2010)

head(combined_long_low_drink)

tail(combined_long_low_drink)
```

```
# Using low_drink_long_fun() to generate ALWDVLTR_der with user-inputted
# values.
#
# Let's say you are a male, you had drinks in the last week and in the last
# year. Let's say you had 5 drinks on Sunday, 1 drink on Monday, 6 drinks on
# Tuesday, 4 drinks on Wednesday, 4 drinks on Thursday, 8 drinks on Friday,
# and 2 drinks on Saturday with a total of 30 drinks in a week.
# Using low_drink_long_fun(), we can check if you would be classified as
# having an increased long term health risk due to drinking.

long_term_drink <- low_drink_long_fun(DHH_SEX = 1, ALWDWKY = 30, ALC_1 = 1,
 ALW_1 = 1, ALW_2A1 = 5, ALW_2A2 = 1, ALW_2A3 = 6, ALW_2A4 = 4, ALW_2A5 = 4,
 ALW_2A6 = 8, ALW_2A7 = 2)

print(long_term_drink)
```

`low_drink_score_fun` *Low drinking score (all cycles)*

Description

This function creates a derived variable based on their drinking habits and flags for health and social problems from their pattern of alcohol use according to Canada's Low-Risk Alcohol Drinking Guideline.

Usage

```
low_drink_score_fun(DHH_SEX, ALWDWKY)
```

Arguments

DHH_SEX	Sex of respondent (1 - male, 2 - female)
ALWDWKY	Number of drinks consumed in the past week

Details

The low risk drinking score is based on the scoring system in Canada's Low-Risk Alcohol Drinking Guideline. The score is divided into two steps. Step 1 allocates points based on sex and the number of drinks that you usually have each week. In step 2, one point will be awarded for each item that is true related to drinking habits. The total score is obtained from adding the points in step 1 and step 2.

Value

Low risk drinking score (low_drink_score) with four categories:

- 1 - Low risk (0 points)
- 2 - Marginal risk (1-2 points)
- 3 - Medium risk (3-4 points)
- 4 - High risk (5-9 points)

Note

Step 2 is not included in this function because the questions in step 2 are not asked in any of the CCHS cycles. The score is only based on step 1.

See https://osf.io/eprg7/ for more details on the guideline and score.

Examples

```
# Using low_drink_score_fun() to create low_drink_score values across
# CCHS cycles low_drink_score_fun() is specified in variable_details.csv
# along with the CCHS variables and cycles included.

# To transform low_drink_score, use rec_with_table() for each CCHS cycle
# and specify low_drink_score, along with the various alcohol and sex
# variables.
# Using merge_rec_data(), you can combine low_drink_score across cycles.

library(cchsflow)
low_drink2001 <- rec_with_table(
  cchs2001_p, c(
    "DHH_SEX", "ALWDWKY", "low_risk_score"
  )
)

head(low_drink2001)

low_drink2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "DHH_SEX", "ALWDWKY", "low_risk_score"
  )
)

tail(low_drink2009_2010)

combined_low_drink <- bind_rows(low_drink2001,
low_drink2009_2010)

head(combined_low_drink)

tail(combined_low_drink)
```

---

low_drink_score_fun1 *Low drinking score (select cycles)*

Description

This function creates a derived variable based on their drinking habits and flags for health and social problems from their pattern of alcohol use according to Canada's Low-Risk Alcohol Drinking Guideline.

Usage

```
low_drink_score_fun1(DHH_SEX, ALWDWKY, ALC_005, ALC_1)
```

Arguments

DHH_SEX	Sex of respondent (1 - male, 2 - female)
ALWDWKY	Number of drinks consumed in the past week
ALC_005	In lifetime, ever had a drink? (1 - yes, 2 - no)
ALC_1	Past year, have you drank alcohol? (1 - yes, 2 - no)

Details

The low risk drinking score is based on the scoring system in Canada's Low-Risk Alcohol Drinking Guideline. The score is divided into two steps. Step 1 allocates points based on sex and the number of drinks that you usually have each week. In step 2, one point will be awarded for each item that is true related to drinking habits. The total score is obtained from adding the points in step 1 and step 2.

This score has two 0 point categories: low risk (never drank) and low risk (former drinker). The two drinking groups are derived from 'ever had a drink in lifetime'. 'Ever had a drink in lifetime' is only available in CCHS 2001-2008 and 2015-2018.

Value

Low risk drinking score (low_drink_score1) with four categories:

- 1 - Low risk - never drank (0 points)
- 2 - Low risk - former drinker (0 points)
- 3 - Marginal risk (1-2 points)
- 4 - Medium risk (3-4 points)
- 5 - High risk (5-9 points)

Note

Step 2 is not included in this function because the questions in step 2 are not asked in any of the CCHS cycles. The score is only based on step 1.

See https://osf.io/eprg7/ for more details on the guideline and score.

Examples

```
# Using low_drink_score_fun1() to create low_drink_score values across
# CCHS cycles low_drink_score_fun1() is specified in variable_details.csv
# along with the CCHS variables and cycles included.

# To transform low_drink_score1, use rec_with_table() for each CCHS cycle
# and specify low_drink_score1, along with the various alcohol and sex
# variables.
```

```
# Using merge_rec_data(), you can combine low_drink_score1 across cycles.

library(cchsflow)
low_drink2001 <- rec_with_table(
  cchs2001_p, c(
    "DHH_SEX", "ALWDWKY", "ALC_005", "ALC_1", "low_risk_score"
  )
)

head(low_drink2001)

low_drink2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "DHH_SEX", "ALWDWKY", "ALC_005", "ALC_1", "low_risk_score"
  )
)

tail(low_drink2009_2010)

combined_low_drink1 <- bind_rows(low_drink2001,
low_drink2009_2010)

head(combined_low_drink1)

tail(combined_low_drink1)
```

`low_drink_short_fun` *Short term risks due to drinking*

Description

This function creates a categorical variable that flags for increased short term health risks due to their drinking habits, according to Canada's Low-Risk Alcohol Drinking Guideline.

Usage

```
low_drink_short_fun(
  DHH_SEX,
  ALWDWKY,
  ALC_1,
  ALW_1,
  ALW_2A1,
  ALW_2A2,
  ALW_2A3,
  ALW_2A4,
  ALW_2A5,
  ALW_2A6,
  ALW_2A7
)
```

Arguments

DHH_SEX	Sex of respondent (1 - male, 2 - female)
ALWDWKY	Number of drinks consumed in the past week
ALC_1	Drinks in the past year (1 - yes, 2 - no)
ALW_1	Drinks in the last week (1 - yes, 2 - no)
ALW_2A1	Number of drinks on Sunday
ALW_2A2	Number of drinks on Monday
ALW_2A3	Number of drinks on Tuesday
ALW_2A4	Number of drinks on Wednesday
ALW_2A5	Number of drinks on Thursday
ALW_2A6	Number of drinks on Friday
ALW_2A7	Number of drinks on Saturday

Details

The classification of drinkers according to their short term health risks comes from guidelines in Alcohol and Health in Canada: A Summary of Evidence and Guidelines for Low-risk Drinking, and is based on the alcohol consumption reported over the past week. Short-term or acute risks include injury and overdose.

Categories are based on CCHS 2015-2016's variable (ALWDVSTR) where short term health risk are increased when drinking more than 3 drinks (for women) or 4 drinks (for men) on any single occasion.

See https://osf.io/ykau5/ for more details on the guideline. See https://osf.io/ycxaq/ for more details on derivation of the function on page 9.

Value

Categorical variable (ALWDVSTR_der) with two categories:

- 1 - Increased short term health risk
- 2 - No increased short term health risk

Examples

```
# Using low_drink_short_fun() to create ALWDVSTR_der values across CCHS cycles
# low_drink_short_fun() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform ALWDVSTR_der, use rec_with_table() for each CCHS cycle
# and specify ALWDVSTR_der, along with the various alcohol and sex
# variables.
# Using merge_rec_data(), you can combine ALWDVSTR_der across cycles.

library(cchsflow)
short_low_drink2001 <- rec_with_table(
```

```
  cchs2001_p, c(
    "ALW_1", "DHH_SEX", "ALW_2A1", "ALW_2A2", "ALW_2A3", "ALW_2A4",
    "ALW_2A5", "ALW_2A6", "ALW_2A7", "ALWDWKY", "ALC_1","ALWDVSTR_der"
  )
)

head(short_low_drink2001)

short_low_drink2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "ALW_1", "DHH_SEX", "ALW_2A1", "ALW_2A2", "ALW_2A3", "ALW_2A4",
    "ALW_2A5", "ALW_2A6", "ALW_2A7", "ALWDWKY", "ALC_1","ALWDVSTR_der"
  )
)

tail(short_low_drink2009_2010)

combined_short_low_drink <- bind_rows(short_low_drink2001,
short_low_drink2009_2010)

head(combined_short_low_drink)

tail(combined_short_low_drink)

# Using low_drink_short_fun() to generate ALWDVSTR_der with user-inputted
# values.
#
# Let's say you are a male, you had drinks in the last week and in the last
# year. Let's say you had 5 drinks on Sunday, 1 drink on Monday, 6 drinks on
# Tuesday, 4 drinks on Wednesday, 4 drinks on Thursday, 8 drinks on Friday,
# and 2 drinks on Saturday with a total of 30 drinks in a week.
# Using low_drink_short_fun(), we can check if you would be classified as
# having an increased short term health risk due to drinking.

short_term_drink <- low_drink_short_fun(DHH_SEX = 1, ALWDWKY = 30, ALC_1 = 1,
 ALW_1 = 1, ALW_2A1 = 5, ALW_2A2 = 1, ALW_2A3 = 6, ALW_2A4 = 4, ALW_2A5 = 4,
 ALW_2A6 = 8, ALW_2A7 = 2)

print(short_term_drink)
```

`merge_rec_data` *Merge recoded data*

Description

This function allows users to merge CCHS data transformed by the `rec_with_table` function. This function generates a labelled merged data frame with multiple transformed CCHS cycles.

Usage

```
merge_rec_data(...)
```

Arguments

...	recoded data frames to be merged.

Details

When merging recoded CCHS data, there are variables that are missing in certain CCHS cycles. This function tags missing variable observations as NA(c), indicating that the variable was not asked or included in the CCHS cycle of the respondent.

Click here for more details on how NA's are treated in cchsflow.

Value

a merged data frame consisting of multiple recoded CCHS cycles with labels for variable names and tags for variables not included in particular CCHS cycles.

Examples

```
# Merging two CCHS cycles with variables missing in each cycle.

# INCGHH_A is a cchsflow variable available for the 2001 CCHS cycle, while
# INCGHH_B is a cchsflow variable available for the 2003 CCHS cycle.
# Using merge_rec_data(), datasets containing INCGHH_A & INCGHH_B can be
# merged and tagged.

library(cchsflow)
income2001 <- rec_with_table(cchs2001_p, "INCGHH_A")
income2003 <- rec_with_table(cchs2001_p, "INCGHH_B")

income_merged <- merge_rec_data(income2001, income2003)
head(income_merged)
tail(income_merged)
```

---

multiple_conditions_fun1

Number of chronic conditions (5 chronic conditions)

Description

This function generates a derived variable (number_conditions) that counts the number of chronic conditions a respondent has. This function takes 5 CCHS-defined conditions (heart disease, cancer, stroke, bowel disorder, and arthritis), and well one derived variable (respiratory condition) to count the number of conditions a respondent has.

Usage

```
multiple_conditions_fun1(
  CCC_121,
  CCC_131,
  CCC_151,
  CCC_171,
  resp_condition_der,
  CCC_051
)
```

Arguments

CCC_121	variable indicating if respondent has heart disease (1 = respondent has heart disease, 2 = respondent does not have heart disease)
CCC_131	variable indicating if respondent has active cancer (1 = respondent has active cancer, 2 = respondent does not have active cancer)
CCC_151	variable indicating if respondent suffers from the effects of a stroke (1 = respondent suffers from stroke effects, 2 = respondent does not suffer from stroke effects)
CCC_171	variable indicating if respondent has a bowel disorder (1 = respondent has bowel disorder, 2 = respondent does not have a bowel disorder)
resp_condition_der	derived variable indicating if respondent has a respiratory condition (1 = respondent is over the age of 35 and has a respiratory condition, 2 = respondent is under the age of 35 and has a respiratory conditions, 3 = respondent does not have a respiratory condition). See resp_condition_fun1 for documentation on how variable was derived.
CCC_051	variable indicating if respondent has arthritis or rheumatism (1 = respondent has arthritis or rheumatism, 2 = respondent does not have arthritis or rheumatism)

Details

mood disorder (CCC_280) was not asked to respondents in the 2001 CCHS survey cycle. This mean respondents in this cycle will only be able to have a maximum of 6 chronic conditions as opposed to 7 for respondents in other cycles. multiple_conditions_fun2 is used for CCHS cycles from 2003 to 2014.

Value

A categorical variable indicating the number of chronic conditions a respondent has. Respondents with 5 or more conditions are grouped in the "5+" category.

See Also

multiple_conditions_fun2

Examples

```
 # Using rec_with_table() to generate multiple_conditions in a CCHS
 # cycle.

 # multiple_conditions_fun1() is specified in variable_details.csv along with
 # the CCHS variables and cycles included.

 # To generate multiple_conditions, use rec_with_table() and specify the
 # multiple_conditions, along with the variables that are derived from it.
 # Since resp_condition_der is also a derived variable, you will have to
 # specify the variables that are derived from it. In this example, data
 # from the 2001 CCHS will be used, so DHHGAGE_cont, CCC_091, and CCC_91A,
 # and CCC_031 will be specified along with resp_condition_der.

library(cchsflow)
 conditions_2001 <- suppressWarnings(rec_with_table(cchs2001_p,
 c("DHHGAGE_cont", "CCC_091",
 "CCC_91A", "CCC_031", "CCC_121","CCC_131","CCC_151", "CCC_171","CCC_280",
 "resp_condition_der","CCC_051", "number_conditions")))

 head(conditions_2001)

 # Generating multiple_conditions with user inputted values
 # Let's say you are an individual that has heart disease, bowel disorder,
 # and arthritis. multiple_conditions_fun1() can be used to count the number
 # of chronic conditions you have

library(cchsflow)
 num_conditions <- multiple_conditions_fun1(CCC_121 = 1, CCC_131 = 2,
 CCC_151 = 2, CCC_171 = 1, resp_condition_der = 3, CCC_051 = 1)

print(num_conditions)
```

---

multiple_conditions_fun2

Number of chronic conditions (6 chronic conditions)

Description

This function generates a derived variable (number_conditions) that counts the number of chronic conditions a respondent has. This function takes 6 CCHS-defined conditions (heart disease, cancer, stroke, bowel disorder, mood disorder and arthritis), and well one derived variable (respiratory condition) to count the number of conditions a respondent has.

Usage

```
multiple_conditions_fun2(
  CCC_121,
```

```
  CCC_131,
  CCC_151,
  CCC_171,
  CCC_280,
  resp_condition_der,
  CCC_051
)
```

Arguments

CCC_121	variable indicating if respondent has heart disease (1 = respondent has heart disease, 2 = respondent does not have heart disease)
CCC_131	variable indicating if respondent has active cancer (1 = respondent has active cancer, 2 = respondent does not have active cancer)
CCC_151	variable indicating if respondent suffers from the effects of a stroke (1 = respondent suffers from stroke effects, 2 = respondent does not suffer from stroke effects)
CCC_171	variable indicating if respondent has a bowel disorder (1 = respondent has bowel disorder, 2 = respondent does not have a bowel disorder)
CCC_280	variable indicating if respondent has a mood disorder (1 = respondent has a mood disorder, 2 = respondent does not have a mood disorder. Note, variable was not asked to respondents in the 2001 CCHS survey cycle.
resp_condition_der	derived variable indicating if respondent has a respiratory condition. (1 = respondent is over the age of 35 and has a respiratory condition, 2 = respondent is under the age of 35 and has a respiratory conditions, 3 = respondent does not have a respiratory condition). See resp_condition_fun1 for documentation on how variable was derived.
CCC_051	variable indicating if respondent has arthritis or rheumatism (1 = respondent has arthritis or rheumatism, 2 = respondent does not have arthritis or rheumatism)

Details

mood disorder (CCC_280) was not asked to respondents in the 2001 CCHS survey cycle. This mean respondents in this cycle will only be able to have a maximum of 6 chronic conditions as opposed to 7 for respondents in other cycles. multiple_conditions_fun1 is used for CCHS cycles from 2003 to 2014.

Value

A categorical variable indicating the number of chronic conditions a respondent has. Respondents with 5 or more conditions are grouped in the "5+" category.

See Also

multiple_conditions_fun1

Examples

```
 # Using rec_with_table() to generate multiple_conditions in a CCHS
 # cycle.

 # multiple_conditions_fun2() is specified in variable_details.csv along with
 # the CCHS variables and cycles included.

 # To generate multiple_conditions, use rec_with_table() and specify the
 # multiple_conditions, along with the variables that are derived from it.
 # Since resp_condition_der is also a derived variable, you will have to
 # specify the variables that are derived from it. In this example, data
 # from the 2010 CCHS will be used, so DHHGAGE_cont, CCC_091, and CCC_031
 # will be specified along with resp_condition_der.

library(cchsflow)
 conditions_2009_2010 <- suppressWarnings(rec_with_table(cchs2009_2010_p,
 c("DHHGAGE_cont", "CCC_091",
 "CCC_031", "CCC_121","CCC_131","CCC_151", "CCC_171","CCC_280",
 "resp_condition_der","CCC_051", "number_conditions")))

 head(conditions_2009_2010)

 # Generating multiple_conditions with user inputted values
 # Let's say you are an individual that has heart disease, bowel disorder,
 # and arthritis. multiple_conditions_fun2() can be used to count the number
 # of chronic conditions you have

library(cchsflow)
 num_conditions <- multiple_conditions_fun2(CCC_121 = 1, CCC_131 = 2,
 CCC_151 = 2, CCC_171 = 1, CCC_280 = 2, resp_condition_der = 3, CCC_051 = 1)

print(num_conditions)
```

---

pack_years_fun *Smoking pack-years*

Description

This function creates a derived variable (pack_years_der) that measures an individual's smoking pack-years based on various CCHS smoking variables. This is a popular variable used by researchers to quantify lifetime exposure to cigarette use.

Usage

```
pack_years_fun(
  SMKDSTY_A,
  DHHGAGE_cont,
  time_quit_smoking,
```

```
  SMKG203_cont,
  SMKG207_cont,
  SMK_204,
  SMK_05B,
  SMK_208,
  SMK_05C,
  SMKG01C_cont,
  SMK_01A
)
```

Arguments

SMKDSTY_A	variable used in CCHS cycles 2001-2014 that classifies an individual's smoking status.
DHHGAGE_cont	continuous age variable.
time_quit_smoking	derived variable that calculates the approximate time a former smoker has quit smoking. See time_quit_smoking_fun for documentation on how variable was derived
SMKG203_cont	age started smoking daily. Variable asked to daily smokers.
SMKG207_cont	age started smoking daily. Variable asked to former daily smokers.
SMK_204	number of cigarettes smoked per day. Variable asked to daily smokers.
SMK_05B	number of cigarettes smoked per day. Variable asked to occasional smokers
SMK_208	number of cigarettes smoked per day. Variable asked to former daily smokers
SMK_05C	number of days smoked at least one cigarette
SMKG01C_cont	age smoked first cigarette
SMK_01A	smoked 100 cigarettes in lifetime (y/n)

Details

pack-years is calculated by multiplying the number of cigarette packs per day (20 cigarettes per pack) by the number of years. Example 1: a respondent who is a current smoker who smokes 1 package of cigarettes for the last 10 years has smoked 10 pack-years. Pack-years is also calculated for former smokers. Example 2: a respondent who started smoking at age 20 years and smoked half a pack of cigarettes until age 40 years smoked for 10 pack-years.

Value

value for smoking pack-years in the pack_years_der variable

Examples

```
# Using pack_years_fun() to create pack-years values across CCHS cycles
# pack_years_fun() is specified in variable_details.csv along with the CCHS
# variables and cycles included.

# To transform pack_years_der across cycles, use rec_with_table() for each
```

```
# CCHS cycle and specify pack_years_der, along with each smoking variable.
# Since time_quit_smoking_der is also a derived
# variable, you will have to specify the variables that are derived from it.
# Then by using merge_rec_data(), you can combine pack_years_der across
# cycles

library(cchsflow)

pack_years2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "SMKDSTY_A", "DHHGAGE_cont", "SMK_09A_B", "SMKG09C", "time_quit_smoking",
    "SMKG203_cont", "SMKG207_cont", "SMK_204", "SMK_05B", "SMK_208",
    "SMK_05C", "SMK_01A", "SMKG01C_cont", "pack_years_der"
  )
)

head(pack_years2009_2010)

pack_years2011_2012 <- rec_with_table(
  cchs2011_2012_p,c(
    "SMKDSTY_A", "DHHGAGE_cont", "SMK_09A_B", "SMKG09C", "time_quit_smoking",
    "SMKG203_cont", "SMKG207_cont", "SMK_204", "SMK_05B", "SMK_208",
    "SMK_05C", "SMK_01A", "SMKG01C_cont", "pack_years_der"
  )
)

tail(pack_years2011_2012)

combined_pack_years <- suppressWarnings(merge_rec_data(pack_years2009_2010,
 pack_years2011_2012))

head(combined_pack_years)
tail(combined_pack_years)
```

`pack_years_fun_cat` *Categorical smoking pack-years*

Description

This function creates a categorical derived variable (pack_years_cat) that categorizes smoking pack-years (pack_years_der).

Usage

```
pack_years_fun_cat(pack_years_der)
```

Arguments

`pack_years_der`	derived variable that calculates smoking pack-years See `pack_years_fun` for documentation on how variable was derived.

Details

pack-years is calculated by multiplying the number of cigarette packs per day (20 cigarettes per pack) by the number of years.The categories were based on the Cardiovascular Disease Population Risk Tool (Douglas Manuel et al. 2018).

pack_years_cat uses the derived variable pack_years_der. Pack_years_der uses age and various smoking variables that have been transformed by cchsflow (see documentation on pack_year_der). In order to categorize pack years across CCHS cycles, age and smoking variables must be transformed and harmonized.

Value

value for pack year categories in the pack_years_cat variable.

Examples

```
# Using pack_years_fun_cat() to categorize pack year values across CCHS cycles
# pack_years_fun_cat() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform pack_years_cat across cycles, use rec_with_table() for each
# CCHS cycle and specify pack_years_cat.
# Since pack_year_der is also also derived variable, you will have to specify
# the variables that are derived from it.
# Since time_quit_smoking_der is also a derived variable in pack_year_der,
# you will have to specify the variables that are derived from it.
# Then by using merge_rec_data(), you can combine pack_years_cat across
# cycles.

library(cchsflow)

pack_years_cat_2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "SMKDSTY_A", "DHHGAGE_cont", "SMK_09A_B", "SMKG09C", "time_quit_smoking",
    "SMKG203_cont", "SMKG207_cont", "SMK_204", "SMK_05B", "SMK_208",
    "SMK_05C", "SMK_01A", "SMKG01C_cont", "pack_years_der", "pack_years_cat"
  )
)

head(pack_years_cat_2009_2010)

pack_years_cat_2011_2012 <- rec_with_table(
  cchs2011_2012_p,c(
    "SMKDSTY_A", "DHHGAGE_cont", "SMK_09A_B", "SMKG09C", "time_quit_smoking",
    "SMKG203_cont", "SMKG207_cont", "SMK_204", "SMK_05B", "SMK_208",
    "SMK_05C", "SMK_01A", "SMKG01C_cont", "pack_years_der", "pack_years_cat"
  )
)

tail(pack_years_cat_2011_2012)
```

```
combined_pack_years_cat <- suppressWarnings(merge_rec_data
(pack_years_cat_2009_2010,pack_years_cat_2011_2012))

head(combined_pack_years_cat)
tail(combined_pack_years_cat)
```

---

pct_time_fun *Percent time in Canada*

Description

This function creates a derived variable (pct_time_der) that provides an estimated percentage of the time a person's life was spent in Canada.

Usage

```
pct_time_fun(DHHGAGE_cont, SDCGCBG, SDCGRES)
```

Arguments

DHHGAGE_cont	continuous age variable.
SDCGCBG	whether or not someone was born in Canada (1 - born in Canada, 2 - born outside Canada)
SDCGRES	how long someone has lived in Canada. Note: in the PUMF CCHS datasets, this is a categorical variable with two categories (1 - 0-9 years; 2 - 10+ years).

Value

Numeric value between 0 and 100 that represents percentage of a respondent's time in Canada

Note

Since SDCGRES is a categorical variable measuring length of time, we've set midpoints in the function. A respondent identified as being in Canada for 0-9 years is assigned a value of 4.5 years, and someone who has been in Canada for over 10 years is assigned a value of 15 years.

Examples

```
# Using pct_time_fun() to create percent time values between CCHS cycles
# pct_time_fun() is specified in variable_details.csv along with the CCHS
# variables and cycles included.

# To transform pct_time_der across cycles, use rec_with_table() for each CCHS
# cycle and specify pct_time_der, along with age (DHHGAGE_cont), whether or
# not someone was born in Canada (SDCGCBG), how long someone has lived in
# Canada (SDCGRES). Then by using merge_rec_data(), you can combine
# pct_time_der across cycles
```

```
library(cchsflow)
pct_time2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "DHHGAGE_cont", "SDCGCBG",
    "SDCGRES", "pct_time_der"
  )
)
head(pct_time2009_2010)

pct_time2011_2012 <- rec_with_table(
  cchs2011_2012_p,  c(
    "DHHGAGE_cont", "SDCGCBG",
    "SDCGRES", "pct_time_der"
  )
)
tail(pct_time2011_2012)

combined_pct_time <- merge_rec_data(pct_time2009_2010, pct_time2011_2012)
head(combined_pct_time)
tail(combined_pct_time)

# Using pct_time_fun() to generate a value for percent time spent in Canada
# with user inputted values Let's say you are 27 years old who was born
# outside of Canada and have been living in Canada for less than 10 years.
# Your estimated percent time spent in Canada can be calculated as follows:

pct_time <- pct_time_fun(DHHGAGE_cont = 27, SDCGCBG = 2, SDCGRES = 1)

print(pct_time)
```

`pct_time_fun_cat` *Categorical percent time in Canada*

Description

This function creates a categorical derived variable (pct_time_der_cat10) that categorizes the derived percent time in Canada variable (pct_time_der).

Usage

```
pct_time_fun_cat(pct_time_der)
```

Arguments

`pct_time_der`	derived continuous percent time in Canada. See `pct_time_fun` for documentation on how variable was derived.

Details

The percent time in Canada provides an estimated percentage of the time a person's life was spent in Canada.The categorical percent time in Canada divides the continuous value into 10 percent intervals.

pct_time_der_cat10 uses the derived variable pct_time_der. pct_time_der uses various variables that have been transformed by cchsflow (see documentation on pct_time_der). In order to categorize percent time in Canada across CCHS cycles, the variables must be transformed and harmonized.

Value

value for categorical percent time in Canada using pct_time_der variable.

Examples

```
# Using pct_time_fun_cat() to create categorical percent time values
# between CCHS cycles.
# pct_time_fun_cat() is specified in variable_details.csv along with the CCHS
# variables and cycles included.

# To transform pct_time_der_cat10 across cycles, use rec_with_table() for
# each CCHS cycle.
# Since pct_time_der is a derived variable, you will have to specify the
# variables that are derived from it.
# Then by using merge_rec_data(), you can combine pct_time_der_cat10 across
# cycles.

library(cchsflow)
pct_time_cat2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "DHHGAGE_cont", "SDCGCBG",
    "SDCGRES", "pct_time_der", "pct_time_der_cat10"
  )
)
head(pct_time_cat2009_2010)

pct_time_cat2011_2012 <- rec_with_table(
  cchs2011_2012_p,  c(
    "DHHGAGE_cont", "SDCGCBG",
    "SDCGRES", "pct_time_der", "pct_time_der_cat10"
  )
)
tail(pct_time_cat2011_2012)

combined_pct_time_cat <- merge_rec_data(pct_time_cat2009_2010,
pct_time_cat2011_2012)
head(combined_pct_time_cat)
tail(combined_pct_time_cat)
```

RACDPAL_fun *Participation and Activity Limitation*

Description

This is a derived variable used in the CCHS (RACDPAL) to classify respondents according to the frequency with which they experience activity limitations due to disability.

Usage

```
RACDPAL_fun(RAC_1, RAC_2A, RAC_2B, RAC_2C)
```

Arguments

RAC_1	Has difficulty with activities due to disability
RAC_2A	Reduction in activities at home due to disability
RAC_2B	Reduction in activities at school or work due to disability
RAC_2C	Reduction in other activities

Details

This derived variable is generated in CCHS cycles 2003-2014. The 2001 CCHS cycle, however, contains the same base variables used to derive this variable. To include respondents in the 2001 CCHS cycle, this custom function was created using the same derivation conditions used in later cycles.

Value

the CCHS derived variable RACDPAL with 3 categories:

1. Sometimes
2. Often
3. Never

Examples

```
# Using RACDPAL_fun() to transform RACDPAL in 2001.
# RACDPAL_fun() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform RACDPAL, use rec_with_table() for each the 2001 cycle
# and specify RACDPAL, along with the various ADL variables.

library(cchsflow)

RACDPAL_2001 <- rec_with_table(
  cchs2001_p, c(
```

```
    "RAC_1", "RAC_2A", "RAC_2B", "RAC_2C", "RACDPAL"
  )
)

head(RACDPAL_2001)

# Note: In other CCHS cycles you only need to specify RACDPAL as the variable
# was included in those survey cycles.

# Using RACDPAL_fun() with user inputted data.

# Let's say you're an individual that sometimes has difficulties with
# activities due to disability, sometimes has a reduction in activities at
# home, often has a reduction at school or work, and never has a reduction
# in other activities. Your participation and activity limitation can be
# determined as follows:

library(cchsflow)
RACDPAL <- RACDPAL_fun(1, 1, 2, 3)
print(RACDPAL)
```

`recode_columns` *recode_columns*

Description

Recodes columns from passed row and returns just table with those columns and same rows as the data

Usage

```
recode_columns(
  data,
  variables_to_process,
  data_name,
  log,
  print_note,
  else_default
)
```

Arguments

`data`	The source database
`variables_to_process`	rows from variable details that are applicable to this DB
`data_name`	Name of the database being passed
`log`	The option of printing log
`print_note`	the option of printing the note columns
`else_default`	default else value to use if no else is present

Value

Returns recoded and labeled data

`recode_variable_NA_formating` *Recode NA formatting*

Description

Recodes the NA depending on the var type

Usage

```
recode_variable_NA_formating(cell_value, var_type)
```

Arguments

`cell_value`	The value inside the recTo column
`var_type`	the toType of a variable

Value

an appropriately coded tagged NA

`rec_with_table` *Recode with Table*

Description

Recode with Table is responsible for recoding values of a dataset based on the specifications in variable_details.

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
)
```

Arguments

`data`	A dataframe containing the variables to be recoded. Can also be a list of dataframes
`variables`	character vector containing variable names to recode or a variables csv containing additional variable info
`database_name`	String, the name of the dataset containing the variables to be recoded. Can also be a vector of strings if data is a list
`variable_details`	A dataframe containing the specifications (rules) for recoding.
`else_value`	Value (string, number, integer, logical or NA) that is used to replace any values that are outside the specified ranges (no rules for recoding).
`append_to_data`	Logical, if `TRUE` (default), recoded variables will be appended to the data.
`log`	Logical, if `FALSE` (default), a log of recoding will not be printed.
`notes`	Logical, if `FALSE` (default), will not print the content inside the 'Note" column of the variable being recoded.
`var_labels`	labels vector to attach to variables in variables
`custom_function_path`	path to location of the function to load
`attach_data_name`	to attach name of database to end table

Details

The variable_details dataframe needs the following variables to function:

variable name of new (mutated) variable that is recoded

toType type the variable is being recoded to *cat = categorical, cont = continuous*

databaseStart name of dataframe with original variables to be recoded

variableStart name of variable to be recoded

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 *variable_details* comprises one category in a newly transformed variable. The rules for each category the new variable are a string in *recFrom* and value in *recTo*. These recode pairs are the same syntax as *sjmisc::rec()*, except in *sjmisc::rec()* the pairs are a string for the function attribute *rec =*, separated by '='. For example in *rec_w_table variable_details$recFrom = 2; variable_details$recTo = 4* is the same as *sjmisc::rec(rec = "2=4")*. the pairs are obtained from the RecFrom and RecTo columns

recode pairs each recode pair is row. see above example or *PBC-variableDetails.csv*

multiple values multiple old values that should be recoded into a new single value may be separated with comma, e.g. *recFrom = "1,2"; recTo = 1*

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

value range for doubles for double vectors (with fractional part), all values within the specified range are recoded; e.g. *recFrom = "1:2.5'; recTo = 1* recodes 1 to 2.5 into 1, but 2.55 would not be recoded (since it's not included in the specified range)

"min"* and *"max" minimum and maximum values are indicates by *min* (or *lo*) and *max* (or *hi*), e.g. *recFrom = "min:4"; recTo = 1* (recodes all values from minimum values of *x* to 4 into 1)

"else" all other values, which have not been specified yet, are indicated by *else*, e.g. *recFrom = "else"; recTo = NA* (recode all other values (not specified in other rows) 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 as old and new value, 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

```
library(cchsflow)
bmi2001 <- rec_with_table(
  data = cchs2001_p, c(
    "HWTGHTM",
    "HWTGWTK", "HWTGBMI_der"
  )
)

head(bmi2001)

bmi2011_2012 <- rec_with_table(
  data = cchs2011_2012_p,  c(
    "HWTGHTM",
    "HWTGWTK", "HWTGBMI_der"
  )
)

tail(bmi2011_2012)

combined_bmi <- bind_rows(bmi2001, bmi2011_2012)
head(combined_bmi)
tail(combined_bmi)
```

resp_condition_fun1 *resp_condition_fun1*

Description

This is one of 3 functions used to create a derived variable (resp_condition_der) that determines if a respondents has a respiratory condition. 3 different functions have been created to account for the fact that different respiratory variables are used across CCHS cycles. This function is for CCHS cycles (2009-2014) that only use COPD and Emphysema as a combined variable. Asthma is used across CCHS cycles as a separate variable.

Usage

```
resp_condition_fun1(DHHGAGE_cont, CCC_091, CCC_031)
```

Arguments

DHHGAGE_cont	continuous age variable.
CCC_091	variable indicating if respondent has either COPD or Emphysema
CCC_031	variable indicating if respondent has asthma

Value

a categorical variable (resp_condition_der) with 3 levels:

1. respondent is over the age of 35 and has a respiratory condition
2. respondent is under the age of 35 and has a respiratory condition
3. respondent does not have a respiratory condition

See Also

resp_condition_fun2, resp_condition_fun3

Examples

```
# Using resp_condition_fun1() to create values across CCHS cycles
# (2009-2014) resp_condition_fun1() is specified in
# variable_details.csv along with the CCHS variables and cycles included.

# To transform resp_condition_der, use rec_with_table() for each CCHS cycle
# and specify resp_condition_der, along with the various respiratory
# variables. Then by using merge_rec_data() you can combine
# resp_condition_der across cycles.

library(cchsflow)

resp2009_2010 <- suppressWarnings(rec_with_table(
  cchs2009_2010_p,  c(
```

```
    "DHHGAGE_cont", "CCC_091", "CCC_031",
    "resp_condition_der"
  )
))

head(resp2009_2010)

resp2011_2012 <- suppressWarnings(rec_with_table(
  cchs2011_2012_p, c(
    "DHHGAGE_cont", "CCC_091", "CCC_031",
    "resp_condition_der"
  )
))

tail(resp2011_2012)

combined_resp <-
 suppressWarnings(merge_rec_data(resp2009_2010, resp2011_2012))

head(combined_resp)
tail(combined_resp)
```

`resp_condition_fun2` *resp_condition_fun2*

Description

This is one of 3 functions used to create a derived variable (resp_condition_der) that determines if a respondents has a respiratory condition. This function is for CCHS cycles (2005-2007) that use COPD & Emphysema as separate variables, as well as Bronchitis. Asthma is used across CCHS cycles as a separate variable.

Usage

```
resp_condition_fun2(DHHGAGE_cont, CCC_91E, CCC_91F, CCC_91A, CCC_031)
```

Arguments

`DHHGAGE_cont`	continuous age variable.
`CCC_91E`	variable indicating if respondent has emphysema
`CCC_91F`	variable indicating if respondent has COPD
`CCC_91A`	variable indicating if respondent has chronic bronchitis
`CCC_031`	variable indicating if respondent has asthma

Value

a categorical variable (resp_condition_der) with 3 levels:

1. respondent is over the age of 35 and has a respiratory condition
2. respondent is under the age of 35 and has a respiratory condition
3. respondent does not have a respiratory condition

See Also

resp_condition_fun1, resp_condition_fun3

Examples

```
# Using resp_condition_fun2() to create values across CCHS cycles
# (2005-2007) resp_condition_fun2() is specified in
# variable_details.csv along with the CCHS variables and cycles included.

# To transform resp_condition_der, use rec_with_table() for each CCHS cycle
# and specify resp_condition_der, along with the various respiratory
# variables. Then by using merge_rec_data() you can combine
# resp_condition_der across cycles.

library(cchsflow)

resp2005 <- suppressWarnings(rec_with_table(
  cchs2005_p, c(
    "DHHGAGE_cont", "CCC_91E", "CCC_91F", "CCC_91A", "CCC_031",
    "resp_condition_der"
  )
))

head(resp2005)

resp2007_2008 <- suppressWarnings(rec_with_table(
  cchs2007_2008_p,  c(
    "DHHGAGE_cont", "CCC_91E", "CCC_91F", "CCC_91A", "CCC_031",
    "resp_condition_der"
  )
))

tail(resp2007_2008)

combined_resp <- suppressWarnings(merge_rec_data(resp2005, resp2007_2008))

head(combined_resp)
tail(combined_resp)
```

resp_condition_fun3 *resp_condition_fun3*

Description

This is one of 3 functions used to create a derived variable (resp_condition_der) that determines if a respondents has a respiratory condition. This function for CCHS cycles (2001-2003) that use COPD and Emphysema as a combined variable, as well as Bronchitis. Asthma is used across CCHS cycles as a separate variable.

Usage

```
resp_condition_fun3(DHHGAGE_cont, CCC_091, CCC_91A, CCC_031)
```

Arguments

DHHGAGE_cont	continuous age variable.
CCC_091	variable indicating if respondent has either COPD or Emphysema
CCC_91A	variable indicating if respondent has chronic bronchitis
CCC_031	variable indicating if respondent has asthma

Value

a categorical variable (resp_condition_der) with 3 levels:

1. respondent is over the age of 35 and has a respiratory condition
2. respondent is under the age of 35 and has a respiratory condition
3. respondent does not have a respiratory condition

See Also

resp_condition_fun1, resp_condition_fun2

Examples

```
# Using resp_condition_fun3() to create values across CCHS cycles
# (2001-2003) resp_condition_fun3() is specified in
# variable_details.csv along with the CCHS variables and cycles included.

# To transform resp_condition_der, use rec_with_table() for each CCHS cycle
# and specify resp_condition_der, along with the various respiratory
# variables. Then by using merge_rec_data() you can combine
# resp_condition_der across cycles.

library(cchsflow)

resp2001 <- suppressWarnings(rec_with_table(
```

```
  cchs2001_p, c(
    "DHHGAGE_cont", "CCC_091", "CCC_91A", "CCC_031",
    "resp_condition_der"
  )
))

head(resp2001)

resp2003 <- suppressWarnings(rec_with_table(
  cchs2003_p,c(
    "DHHGAGE_cont", "CCC_091", "CCC_91A", "CCC_031",
    "resp_condition_der"
  )
))

tail(resp2003)

combined_resp <- suppressWarnings(merge_rec_data(resp2001, resp2003))

head(combined_resp)
tail(combined_resp)
```

`set_data_labels` *Set Data Labels*

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

Examples

```
library(cchsflow)
library(sjlabelled)
bmi2001 <- rec_with_table(
 cchs2001_p, c(
    "HWTGHTM",
    "HWTGWTK", "HWTGBMI_der"
  )
)

bmi2003 <- rec_with_table(
  cchs2003_p, c(
    "HWTGHTM",
    "HWTGWTK", "HWTGBMI_der"
  )
)

combined_bmi <- bind_rows(bmi2001, bmi2003)

get_label(combined_bmi)

labeled_combined_data <- set_data_labels(combined_bmi,
 variable_details,
 variables)

get_label(labeled_combined_data)
```

SMKG040_fun *Age started smoking daily - daily/former daily smokers*

Description

This function creates a continuous derived variable (SMKG040_fun) that calculates the approximate age that a daily or former daily smoker began smoking daily.

Usage

```
SMKG040_fun(SMKG203_cont, SMKG207_cont)
```

Arguments

SMKG203_cont	age started smoking daily. Variable asked to daily smokers.
SMKG207_cont	age started smoking daily. Variable asked to former daily smokers.

Details

SMKG203 (daily smoker) and SMKG207 (former daily) are present in CCHS 2001-2014, and are separate variables. For CCHS 2015 and onward, SMKG040 (daily/former daily) combines the two previous variables. SMKG040_fun takes the continuous functions (SMKG203_cont and SMKG207_cont) to create SMKG040 for 2001-2014.

Value

value for age started smoking daily for daily/former daily smokers in the SMKG040_cont

Note

In previous cycles, both SMKG203 and SMKG207 included respondents who did not state their smoking status. From CCHS 2015 and onward, SMKG040 only included respondents who specified daily smoker or former daily smoker. As a result, SMKG040 has a large number of missing respondents for CCHS 2015 survey cycles and onward.

Examples

```
# Using SMKG040_fun() to create age values across CCHS cycles
# SMKG040_fun() is specified in variable_details.csv under SMKG040_cont.

# To create a continuous harmonized variable for SMKG040, use rec_with_table()
# for each CCHS cycle and specify SMKG040_cont.

library(cchsflow)

age_smoke_dfd_2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "SMKG203_cont", "SMKG207_cont","SMKG040_cont"
  )
)

head(age_smoke_dfd_2009_2010)

age_smoke_dfd_2011_2012 <- rec_with_table(
  cchs2011_2012_p,c(
    "SMKG203_cont", "SMKG207_cont","SMKG040_cont"
  )
)

tail(age_smoke_dfd_2011_2012)

combined_age_smoke_dfd <- suppressWarnings(merge_rec_data
(age_smoke_dfd_2009_2010,age_smoke_dfd_2011_2012))

head(combined_age_smoke_dfd)
tail(combined_age_smoke_dfd)
```

---

smoke_simple_fun *Simple smoking status*

Description

This function creates a derived smoking variable (smoke_simple) with four categories:

- non-smoker (never smoked)
- current smoker (daily and occasional?)
- former daily smoker quit =<5 years or former occasional smoker
- former daily smoker quit >5 years

Usage

```
smoke_simple_fun(SMKDSTY_cat5, time_quit_smoking)
```

Arguments

SMKDSTY_cat5	derived variable that classifies an individual's smoking status. This variable captures cycles 2001-2018.
time_quit_smoking	derived variable that calculates the approximate time a former smoker has quit smoking. See time_quit_smoking_fun for documentation on how variable was derived.

Examples

```
# Using the 'smoke_simple_fun' function to create the derived smoking
# variable across CCHS cycles.
# smoke_simple_fun() is specified in the variable_details.csv

# To create a harmonized smoke_simple variable across CCHS cycles, use
# rec_with_table() for each CCHS cycle and specify smoke_simple_fun and
# the required base variables. Since time_quit_smoking_der is also a derived
# variable, you will have to specify the variables that are derived from it.
# Using merge_rec_data(), you can combine smoke_simple across cycles.

library(cchsflow)

smoke_simple2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
    "SMKDSTY", "SMK_09A_B", "SMKG09C", "time_quit_smoking",
    "smoke_simple"
  )
)

head(smoke_simple2009_2010)

smoke_simple2011_2012 <- rec_with_table(
  cchs2011_2012_p,c(
   "SMKDSTY", "SMK_09A_B", "SMKG09C", "time_quit_smoking",
   "smoke_simple"
  )
)
```

```
tail(smoke_simple2011_2012)

combined_smoke_simple <-
suppressWarnings(merge_rec_data(smoke_simple2009_2010,smoke_simple2011_2012))

head(combined_smoke_simple)
tail(combined_smoke_simple)
```

`time_quit_smoking_fun` *Time since quit smoking*

Description

This function creates a derived variable (time_quit_smoking_der) that calculates the approximate time a former smoker has quit smoking based on various CCHS smoking variables. This variable is for CCHS respondents in CCHS surveys 2003-2014.

Usage

```
time_quit_smoking_fun(SMK_09A_B, SMKG09C)
```

Arguments

SMK_09A_B	number of years since quitting smoking. Variable asked to former daily smokers who quit <3 years ago.
SMKG09C	number of years since quitting smoking. Variable asked to former daily smokers who quit >=3 years ago.

Value

value for time since quit smoking in time_quit_smoking_der.

Examples

```
# Using time_quit_smoking_fun() to create pack-years values across CCHS
# cycles.
# time_quit_smoking_fun() is specified in variable_details.csv along with the
# CCHS variables and cycles included.

# To transform time_quit_smoking across cycles, use rec_with_table() for each
# CCHS cycle and specify time_quit_smoking, along with each smoking variable.
# Then by using merge_rec_data(), you can combine time_quit_smoking across
# cycles.

library(cchsflow)

time_quit2009_2010 <- rec_with_table(
  cchs2009_2010_p, c(
```

```
    "SMK_09A_B", "SMKG09C", "time_quit_smoking"
  )
)

head(time_quit2009_2010)

time_quit2011_2012 <- rec_with_table(
  cchs2011_2012_p, c(
    "SMK_09A_B", "SMKG09C", "time_quit_smoking"
  )
)

tail(time_quit2011_2012)

combined_time_quit <- suppressWarnings(merge_rec_data(time_quit2009_2010,
 time_quit2011_2012))

head(combined_time_quit)
tail(combined_time_quit)
```

`variables` *variables.csv*

Description

This dataset lists all the variables that are present in cchsflow.

Details

See the below link for more details about how the worksheet is structured https://big-life-lab.github.io/cchsflow/articles/variables_sheet.html

Value

`variables` a data frame

Examples

```
data(variables)
str(variables)
```

variable_details *variable_details.csv*

Description

This dataset provides details on how variables are recoded in cchsflow.

Details

See the below link for more details about how the worksheet is structured https://big-life-lab.github.io/cchsflow/articles/variable_details.html

Value

variable_details
: a data frame

Examples

```
data(variable_details)
str(variable_details)
```

Index