Package 'amp.dm'

May 7, 2026

Title Data Management Tools for Pharmacometrics

Version 0.2.1

Description Tools and functions to efficiently create datasets used in pharmacometric analysis. Additional functionality is added to create documentation and prepare files for submission and quality control purposes.

Depends R (>= 4.3.0)

License GPL (>= 3)

URL https://leidenadvancedpkpd.github.io/amp.dm/, https://github.com/LeidenAdvancedPKPD/amp.dm

BugReports https://github.com/LeidenAdvancedPKPD/amp.dm/issues

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder quarto

Imports tools, utils, stats, grDevices, cli, rlang, dplyr, vctrs, forcats, readxl, haven, fs, xtable

Suggests ggplot2, patchwork, rstudioapi, knitr, R3port, testthat, quarto, tidyr

NeedsCompilation no

Author Richard Hooijmaijers [aut, cre, cph], LAPP Consultants [fnd, cph]

Maintainer Richard Hooijmaijers <richardhooijmaijers@gmail.com>

Repository CRAN

Date/Publication 2026-03-30 09:30:31 UTC

Contents

`attr_add` *Add attributes from a list to a dataframe*

Description

This function adds data attributes available in a list to a data frame. Additional checks are done to verify if the attributes are in a valid and use-able format

Usage

```
attr_add(data, attrl, attrib = c("label", "format", "remark"), verbose = TRUE)
```

Arguments

data	the data frame for which the attributes should be set
attrl	named list with the attributes for the dataset (see details)
attrib	a vector of attributes that should be set for data (currently 'label', 'format' and 'remark' are applicable)
verbose	a logical indicating if datachecks should be printed to console

Details

This function adds attributes available in a list to a data frame. The structure of this list should be available in a specific format. The names items in the list are aligned with the variables in the data frame. For each item, the content of the 'label', 'format' and 'remark' elements will be added as attributes to the dataset. For an example of the format of list see for instance attr_xls.

Value

dataframe with the attributes added

Author(s)

Richard Hooijmaijers

See Also

attr_xls

Examples

```
xmpl   <- system.file("example/Attr.Template.xlsx",package = "amp.dm")
attrl  <- attr_xls(xmpl)
data  <- read.csv(system.file("example/NM.theoph.V1.csv",package = "amp.dm"), na.strings = ".")
attr_add(data,attrl) |> str()
```

---

attr_extract *Extracts attributes from a data frame*

Description

This function extracts attributes available in a data frame and creates a structured list

Usage

```
attr_extract(dfrm)
```

Arguments

dfrm	data frame containing the attributes

Value

named list with the attributes

Author(s)

Richard Hooijmaijers

Examples

```
attrl  <- attr_xls(system.file("example/Attr.Template.xlsx",package = "amp.dm"))
nm     <- read.csv(system.file("example/NM.theoph.V1.csv",package = "amp.dm"))
nmf    <- attr_add(nm, attrl, verbose = FALSE)
attrl2 <- attr_extract(nmf)
all.equal(attrl,attrl2)
```

`attr_factor` *Create factors for variables within a data frame using the format attribute*

Description

This function searches for the 'format' attribute within a data frame, if found it applies the format to that variable. The resulting variable will be a factor useful for plotting and reporting

Usage

```
attr_factor(data, verbose = TRUE, largestfirst = FALSE)
```

Arguments

`data`	the data.frame for which factors should be created
`verbose`	a logical indicating if datachecks should be printed to console
`largestfirst`	either a logical or a character vector indicating if the largest group should be the first level (see details)

Details

In order to make this function work the 'format' attribute should be present and should be available as a named vector (e.g. `attr(data$GENDER,'format') <- c('0' = 'Male', '1' = 'Female')`). If the attribute is found it overwrites the variable with the format applied to it. Be aware that the original levels defined in the format could be lost in this process. The 'largestfirst' argument could be set to a logical which indicates if a for all variables in the dataset, the largest group should be the first level. The argument could also be a character vector indicating for which of the variables in the dataset, the largest group should be the first level. In case you want to set a specific order, this can be done directly in the the format attribute, e.g. `attr(data$VAR,'format') <- c('2' = 'level 1', '1' = 'Level 2')`

Value

data frame with the formats assigned

Author(s)

Richard Hooijmaijers

See Also

factor, attr_add, attr_xls

Examples

```
dfrm <- data.frame(GENDER=rep(c(0,1),4),RESULT=rnorm(8))
attr(dfrm$GENDER,'format')   <- c('0' = 'Male', '1' = 'Female')
dfrm <- attr_factor(dfrm)
str(dfrm$GENDER)
```

---

attr_xls *Reads in attributes from an external excel file*

Description

This function reads in attributes available in an excel file and creates a structured list

Usage

```
attr_xls(xls, sepfor = "\n", nosort = FALSE)
```

Arguments

xls	character with the name of the excel file containing the attributes
sepfor	character of length 1 indicating what the separator for formats should be
nosort	logical indicating if sorting of variables should be omitted (otherwise sorting of no. column in excel file is applied)

Value

named list with the attributes

Author(s)

Richard Hooijmaijers

Examples

```
xmpl  <- system.file("example/Attr.Template.xlsx",package = "amp.dm")
head(attr_xls(xmpl),3)
```

check_cat *Create an overview of the available categories*

Description

This function reports information for the categories, mainly the frequencies, proportions and missing values

Usage

```
check_cat(x, missing = c(-999, NA), detail = 5, threshold = c(NA, NA))
```

Arguments

x	numeric vector with the categories
missing	vector with the values that present missing information
detail	numeric with he level of detail to print (see below for details)
threshold	numeric vector with the threshold numbers and proportions (see details)

Details

The detail argument can be used to print certain information:

- 5: All possible information is printed
- 4: Only the table with frequencies and proportions
- 3: Only information regarding missing data
- 2: Only a warning in case number of missing is above threshold (see below)
- 1: A named vector with the available categories that can be used in num_lump The threshold presents the absolute number (first number) and proportions (second number) to check. If either one of these numbers is above the threshold for missing values, a warning is given. This can be convenient to decide whether or not a category should be used during analyses.

Value

Nothing is returned information is printed on screen

Author(s)

Richard Hooijmaijers

Examples

```
dfrm   <- data.frame(cat1 = c(rep(1:5,10),-999),
                     cat2 = c(rep(letters[1:5],10),-999))
check_cat(dfrm$cat1)
check_cat(dfrm$cat2, detail=1)
check_cat(dfrm$cat1,detail=2,threshold = c(NA,0.1))
```

check_nmdata *Checks nonmem dataset for common errors/mistakes*

Description

This function checks if there are any common errors or mistakes within a NONMEM dataset, and reports the results back to console, table or dataframe

Usage

```
check_nmdata(
  x,
  type = 1,
  ret = "tbl",
  capt = NULL,
  align = NULL,
  size = "\\footnotesize",
  ...
)
```

Arguments

x	either a path to a CSV file or a data frame with the NONMEM data that should be checked
type	integer with the type of checks. Currently 1 can be used for checks that should all pass for a valid analysis and 2 for checks that trigger further investigation
ret	a character vector to define what kind of output should be returned (either "dfrm", "tbl", "file")
capt	character with the caption of the table (not used in case data frame is returned)
align	alignment of the table passed to general_tbl (not used in case data frame is returned)
size	character with font size as for the table general_tbl
...	additional arguments passed to general_tbl

Value

the checks are either printed, returned as dataframe or placed in a PDF

Author(s)

Richard Hooijmaijers

Examples

```
chkf <- system.file("example/NM.theoph.V1.csv",package = "amp.dm")
check_nmdata(chkf)
```

cmnt *Add comment to environment to present in documentation*

Description

Adds a comment regarding assumptions and special attention into package environment, which can be used in code chunks and easily printed after a code chunk

Usage

```
cmnt(string = "", bold = FALSE, verbose = TRUE)
```

Arguments

string	character of length one with the comment to add
bold	logical indicating if the string should be printed in bold to emphasize importance
verbose	logical indicating if the comment should be printed when function is called

Value

no return value, called for side effects

Author(s)

Richard Hooijmaijers

Examples

```
cmnt("Exclude time points > 12h")
cmnt("Subject 6 deviates and is excluded in the analysis", TRUE)
# Markdown syntax can be used for comments:
cmnt("We can use **bold** and *italic* or `code`")
# we can print the contents of the comments with
get_log()$cmnt_nfo
```

cmnt_print *Function that prints the comments given by cmnt*

Description

Prints the results in markdown format to be used directly in inline coding

Usage

```
cmnt_print(clean = TRUE)
```

Arguments

clean	logical indicating if the comments should be deleted after printing (see details)

Details

The function returns a text string with the comments given up to the point it was called. When clean is set to TRUE (default), the content of the comment dataset is cleaned to overcome repetition of comments each time it is called

Value

character string with the comments

Author(s)

Richard Hooijmaijers

Examples

```
cmnt("Comment to print")
cmnt_print()
```

---

contents_df *Create information for multiple data frames*

Description

This function creates a latex table or data frame with the number of records, subjects and variables of one or multiple data frames.

Usage

```
contents_df(
  dfv,
  subject = NULL,
  ret = "tbl",
  capt = "Information multiple data frames",
  align = "lllp{8cm}",
  ...
)
```

Arguments

dfv	a character vector with data frame(s) in global environment for which the overview should be created
subject	character string that identifies the subject variable within the data frame

ret	a character vector to define what kind of output should be returned (either "dfrm", "tbl", "file")
capt	character with the caption of the table (not used in case data frame is returned)
align	alignment of the table passed to general_tbl (not used in case data frame is returned)
...	additional arguments passed to general_tbl

Details

This function can be used to create a table with the most important information of a data frame for documentation. The function will list the the number of records, subjects and variables of each data frame within dfv. This function is especially usable to indicate the differences between similar data frames or an overview of all data frames within a working environment

Value

a data frame, code for table or nothing in case a PDF file is created

Author(s)

Richard Hooijmaijers

Examples

```
Theoph1 <-  subset(Theoph,Subject!=1)
Theoph2 <-  subset(Theoph,Subject!=2)
contents_df(c('Theoph1','Theoph2'),subject='Subject',ret='dfrm')
```

---

counts_df *Create counts and frequencies within in data frame*

Description

This function calculates and reports counts and frequencies stratified by one or more variables within a data frame

Usage

```
counts_df(
  data,
  by,
  id = NULL,
  style = 1,
  ret = "tbl",
  capt = "Information multiple data frames",
  align = NULL,
  size = "\\footnotesize",
  ...
)
```

Arguments

data	data frame for which the table should be created
by	character identifying by variables within the data frame for stratification
id	character identifying the ID variable within the data frame (see details)
style	numeric with the type of output to return (see details)
ret	a character vector to define what kind of output should be returned (either "dfrm", "tbl", "file")
capt	character with the caption of the table (not used in case data frame is returned)
align	alignment of the table passed to general_tbl (not used in case data frame is returned)
size	character with font size as for the table general_tbl
...	additional arguments passed to general_tbl

Details

This function generates frequency tables, by default for the number of observation per strata. In case the id argument is used the function will also report the number and frequencies of distinct IDs. By default the observations and percentages are reported in separate columns (convenient for further processing). In case style is set to a value of 2, a single column is created that holds the observations and percentages in a formatted ways (convenient for tabulating)

Value

a data frame, code for table or nothing in case a PDF file is created

Author(s)

Richard Hooijmaijers

Examples

```
data("Theoph")
Theoph$trt <- ifelse(as.numeric(Theoph$Subject)<6,1,2)
Theoph$sex <- ifelse(as.numeric(Theoph$Subject)<4,1,0)
counts_df(data=Theoph, by=c("trt","sex"),id="Subject", ret="dfrm")
counts_df(data=Theoph, by="sex",id="Subject", ret="dfrm")
counts_df(data=Theoph, by=c("trt","sex"),id="Subject", style=2, ret="dfrm")
```

`create_addl` *Create ADDL data item and deletes unnecessary amount lines*

Description

This function determines if subsequent dose items are exactly the same as the tau value. If this is the case it will count the number of times this occur and create the applicable number of additional dose levels and removes unnecessary rows

Usage

```
create_addl(data, datetime, id, dose, tau, evid = NULL)
```

Arguments

`data`	data frame to perform the action on
`datetime`	character identifying the date/time variable (POSIXct) within the data frame
`id`	character identifying the subject ID within the data frame
`dose`	character identifying the dose within the data frame (ADDL can only be set for equal doses)
`tau`	character identifying the tau (or II) within the data frame
`evid`	character identifying the event ID (EVID) within the data frame. This is used to distinguish observations from dosing records, e.g. 0 for observations

Value

a data frame with ADDL records added

Author(s)

Richard Hooijmaijers

Examples

```
dts <- c(Sys.time(),Sys.time() + (24*60*60),Sys.time() + (48*60*60),Sys.time() + (96*60*60))
data <- data.frame(id=1,dt=dts,dose=10,tau=24)
create_addl(data=data, datetime="dt", id="id", dose="dose", tau="tau")

data2 <- rbind(cbind(data,evid=1),data.frame(id=1,dt=dts[4]+60,dose=10,tau=24,evid=0))
create_addl(data=data2, datetime="dt", id="id", dose="dose", tau="tau", evid="evid")
```

`define_tbl` *Create define PDF file for submission of pharmacometric data files*

Description

This function creates the define.pdf file necessary for esubmission.

Usage

```
define_tbl(
  attr = NULL,
  ret = "dfrm",
  capt = "Dataset define form",
  align = "lp{3cm}lp{8cm}",
  outnm = NULL,
  orientation = "portrait",
  size = "\\footnotesize",
  src = NULL,
  ...
)
```

Arguments

`attr`	list with datasets attributes
`ret`	a character vector to define what kind of output should be returned (either "dfrm", "tbl", "file")
`capt`	character with the caption of the table
`align`	alignment of the table passed to general_tbl
`outnm`	character with the name of the tex file to generate and compile (e.g. define.tex)
`orientation`	character the page orientation in case a file is to be returned (can be either 'portrait' or 'landscape')
`size`	character with font size as for the table general_tbl
`src`	object that holds information regarding the source (e.g. `get_log()$srce_nfo`), if NULL an attempt is made to get it from the environment
`...`	additional arguments passed to general_tbl

Value

a data frame, code for table or nothing in case a PDF file is created

Author(s)

Richard Hooijmaijers

Examples

```
xmpl   <- system.file("example/Attr.Template.xlsx",package = "amp.dm")
attrl  <- attr_xls(xmpl)
define_tbl(attrl)
```

`egfr` *Calculates EGFR values based on different types of formulas*

Description

This function calculates Estimated Glomerular Filtration Rate (EGFR) values based on most commonly used formulas

Usage

```
egfr(
  scr = NULL,
  sex = NULL,
  age = NULL,
  race = NULL,
  ht = NULL,
  bun = NULL,
  scys = NULL,
  prem = NULL,
  bsa = NULL,
  formula = "CKD-EPI"
)
```

Arguments

`scr`	vector with Serum creatinine values in mg/dL
`sex`	vector with SEX values (where female is defined as a value of 1)
`age`	vector with AGE values in years
`race`	vector with RACE values (where caucasian is defined as 1, black as and Japanese as > 2)
`ht`	vector with HEIGHT values in cm
`bun`	vector with Blood urea nitrogen in mg/dL
`scys`	vector with Serum cystatin C in mg/L
`prem`	vector with PREM (premature) values (where PREM is defined as value of 1)
`bsa`	vector with BSA values in m2 provide in case correction should be applied (see details)
`formula`	character with the formula to be used for the EGFR calculations (see details)

Details

Currently there are different formulas available for calculations:

- "CKD-EPI": EGFR according to the Chronic Kidney Disease Epidemiology (CKD-EPI) study formula (Levey):

$$\text{eGFR} = 141 \cdot \min\left(\frac{\text{Scr}}{\kappa}, 1\right)^{\alpha} \cdot \max\left(\frac{\text{Scr}}{\kappa}, 1\right)^{-1.209} \cdot 0.993^{\text{Age}} \cdot 1.159 \text{ [if black]} \cdot 1.018 \text{ [if female]}$$

  where $\min()$ indicates the minimum of $\frac{\text{Scr}}{\kappa}$ or 1; $\max()$ indicates the maximum of $\frac{\text{Scr}}{\kappa}$or 1. scaling parameter κ is 0.9 for males and 0.7 for females and scaling parameter α is -0.411 for males and -0.329 for females.

- "CKD-EPI-ignore-race": EGFR according to the Chronic Kidney Disease Epidemiology (CKD-EPI) refit without race study formula (Delgado):

$$\text{eGFR} = 142 \cdot \min\left(\frac{\text{Scr}}{\kappa}, 1\right)^{\alpha} \cdot \max\left(\frac{\text{Scr}}{\kappa}, 1\right)^{-1.200} \cdot 0.9938^{\text{Age}} \cdot 1.012 \text{ [if female]}$$

  where $\min()$ indicates the minimum of $\frac{\text{Scr}}{\kappa}$ or 1; $\max()$ indicates the maximum of $\frac{\text{Scr}}{\kappa}$or 1. scaling parameter κ is 0.9 for males and 0.7 for females and scaling parameter α is -0.302 for males and -0.241 for females.

- "CKD-EPI-Scys", EGFR according to the Chronic Kidney Disease Epidemiology study formula (Inker):

$$\text{eGFR} = 133 \cdot \min\left(\frac{\text{Scys}}{0.8}, 1\right)^{-0.499} \cdot \max\left(\frac{\text{Scys}}{0.8}, 1\right)^{-1.328} \cdot 0.996^{\text{Age}} \cdot 0.932 \text{ [if female]}$$

  where $\min()$ indicates the minimum of $\frac{\text{Scys}}{0.8}$ or 1; $\max()$ indicates the maximum of $\frac{\text{Scys}}{0.8}$ or 1.

- "CKD-EPI-Scr-Scys", EGFR according to the Chronic Kidney Disease Epidemiology study formula (Inker):

$$\text{eGFR} = k \cdot l \cdot 135 \cdot \min\left(\frac{\text{Scr}}{\kappa}, 1\right)^{\alpha} \cdot \max\left(\frac{\text{Scr}}{\kappa}, 1\right)^{-0.601} \cdot \min\left(\frac{\text{Scys}}{0.8}, 1\right)^{-0.375} \cdot \max\left(\frac{\text{Scys}}{0.8}, 1\right)^{-0.711} \cdot 0.995^{\text{Age}}$$

  where $\min()$ indicates the minimum of $\frac{\text{Scys}}{0.8}$ or 1; $\max()$ indicates the maximum of $\frac{\text{Scys}}{0.8}$ or 1, and where $\min()$ indicates the minimum of $\frac{\text{Scr}}{\kappa}$ or 1; $\max()$ indicates the maximum of $\frac{\text{Scr}}{\kappa}$or 1. Scaling parameter k is 1 for males and 0.969 for female, scaling parameter l is 1 if White/Caucasian and 1.08 if Black/African American, scaling parameter κ is 0.9 for males and 0.7 for females and scaling parameter α is -0.207 for males and -0.248 for females.

- "CKD-EPI-Scr-Scys-ignore-race", EGFR according to the Chronic Kidney Disease Epidemiology 2021 refit without race study formula (Delgado):

$$\text{eGFR} = k \cdot 135 \cdot \min\left(\frac{\text{Scr}}{\kappa}, 1\right)^{\alpha} \cdot \max\left(\frac{\text{Scr}}{\kappa}, 1\right)^{-0.544} \cdot \min\left(\frac{\text{Scys}}{0.8}, 1\right)^{-0.323} \cdot \max\left(\frac{\text{Scys}}{0.8}, 1\right)^{-0.778} \cdot 0.9961^{\text{Age}}$$

  where $\min()$ indicates the minimum of $\frac{\text{Scys}}{0.8}$ or 1; $\max()$ indicates the maximum of $\frac{\text{Scys}}{0.8}$ or 1, and where $\min()$ indicates the minimum of $\frac{\text{Scr}}{\kappa}$ or 1; $\max()$ indicates the maximum of $\frac{\text{Scr}}{\kappa}$or 1. Scaling parameter k is 1 for males and 0.963 for female, scaling parameter κ is 0.9 for males and 0.7 for females and scaling parameter α is -0.144 for males and -0.219 for females.

- "CKD-EPI-Japan", EGFR in Japanese adults based on a Japanese coefficient-modified CKD-EPI equation (Horio):

$$\text{eGFR} = l \cdot 141 \cdot \min\left(\frac{\text{Scr}}{\kappa}, 1\right)^{\alpha} \cdot \max\left(\frac{\text{Scr}}{\kappa}, 1\right)^{-1.209} \cdot 0.993^{\text{Age}} \cdot 1.018 \text{ [if female]}$$

  where $\min()$ indicates the minimum of $\frac{\text{Scr}}{\kappa}$ or 1; $\max()$ indicates the maximum of $\frac{\text{Scr}}{\kappa}$or 1. Scaling parameter l is 1 for White/Caucasian, 1.159 for Black/African American, 0.813 for Japanese, scaling parameter κ is 0.9 for males and 0.7 for females and scaling parameter α is -0.411 for males and -0.329 for females.

- "CKD-MDRD", EGFR according to the abbreviated Modification of Diet in Renal Disease study formula (Levey):

$$\text{eGFR} = 186 \cdot \text{Scr}^{-1.154} \cdot \text{Age}^{-0.203} \cdot 1.212 \text{ [if black]} \cdot 0.742 \text{ [if female]}$$

- "CKD-MDRD2", EGFR according to the re-expressed Modification of Diet in Renal Disease (MDRD) study formula (Levey2007):

$$\text{eGFR} = 175 \cdot \text{Scr}^{-1.154} \cdot \text{Age}^{-0.203} \cdot 1.212 \text{ [if black]} \cdot 0.742 \text{ [if female]}$$

- "Schwartz-original", EGFR in children, according to the original Schwartz formula (Schwartz1987):

$$\text{eGFR} = k \cdot \frac{\text{Height}}{\text{Scr}}$$

  where k = 0.33 in pre-term infants up to 1 year, k = 0.45 in full-term infants up to 1 year, k = 0.55 in children 1 year to 13 years, k = 0.55 in girls >13 and <18 years and k = 0.70 in boys >13 and <18 years.

- "Schwartz-CKiD", EGFR in children, according to the Chronic Kidney Disease in Children (CKiD) revised Schwartz formula (Schwartz2012):

$$\text{eGFR} = 39.8 \cdot \left(\frac{\text{Height}}{\text{Scr}}\right)^{0.456} \cdot \left(\frac{1.8}{\text{Scys}}\right)^{0.418} \cdot \left(\frac{30}{\text{BUN}}\right)^{0.079} \cdot \left(\frac{\text{Height}}{1.4}\right)^{0.079}$$

  Scaling parameter k is 1 for males and 1.076 for females.

- "Schwartz-1B", EGFR in children, according to the Chronic Kidney Disease in Children (CKiD) 1B Schwartz formula (Schwartz2009):

$$\text{eGFR} = 40.7 \cdot \left(\frac{\text{Height}}{\text{Scr}}\right)^{0.64} \cdot \left(\frac{30}{\text{BUN}}\right)^{0.202}$$

- "Schwartz", EGFR in children, according to the updated ('bedside') Schwartz formula (Schwartz2009):

$$\text{eGFR} = 0.413 \cdot \frac{\text{Height}}{\text{Scr}}$$

  This equation is not meant for patients < 1 years of age.

- "Mayo-Quadratic", EGFR according to the Quadratic Mayo Clinic formula (Rule).

$$\text{eGFR} = \exp\left(1.911 + \frac{5.249}{\text{Scr}} - \frac{2.114}{\text{Scr}^2} - 0.00686 \cdot \text{Age} - 0.205 \text{ [if female]}\right)$$

  If Scr < 0.8 mg/dL, a value of 0.8 is used in the equation.

- "Matsuo-Japan", EGFR in Japanese adults, according to Matsuo:

$$\text{eGFR} = 194 \cdot \text{Scr}^{-1.094} \cdot \text{Age}^{-0.287} \cdot 0.739 \text{ [if female]}$$

For all of the calculation methods described above, the reported EGFR values are in the units "mL/minute/1.73m2". This means that the value is referenced to a body surface area (BSA) value of 1.73m2. When a value is provided for BSA, the final outcome will be corrected for the BSA value and the units become "mL/minute". This is done by multiplying the eGFR (referenced to a BSA of 1.73m2) with the individual's BSA (it is the users responsibility to proved BSA values that are calculated using the appropriate formula) and divided by 1.73. Additional information regarding this can be found in a FDA guidance document.

Value

a vector with EGFR values

Author(s)

Richard Hooijmaijers

Examples

```
# dataset with dummy numbers!
crea <- data.frame(id=c(1,1,2),Scr=runif(3),SEX=c(1,1,0),AGE=runif(3),RACE=c(1,1,2))
egfr(crea$Scr,crea$SEX,crea$AGE,crea$RACE, formula="CKD-EPI")
# example for use in dplyr
crea |> dplyr::mutate(EGFR = egfr(Scr,SEX, AGE, RACE, formula="CKD-EPI"))
```

---

expand_addl_ii *Expand rows in case ADDL and II variables are present*

Description

This function expands ADDL and II records. This is done by placing each ADDL record on a separate line. This is convenient in case of individual dose calculations

Usage

```
expand_addl_ii(data, evid = NULL, del_iiaddl = TRUE)
```

Arguments

data	data frame to perform the expansion on
evid	character identifying the event ID (EVID) within the data frame This is used to distinguish observations from dosing records, e.g. 0 for observations
del_iiaddl	logical identifying if the ADDL and II variables can be deleted from output

Details

The function expects that certain variables are present in the data (at least ID, TIME, ADDL and II)

Value

a data frame with expanded dose records

Author(s)

Richard Hooijmaijers

Examples

```
dfrm <- data.frame(ID=c(1,1), TIME=c(0,12),II=c(12,0),ADDL=c(5,0),AMT=c(10,0),EVID=c(1,0))
expand_addl_ii(dfrm,evid="EVID")
```

---

fill_dates *Fills down dates within a data frame that include a start and end date*

Description

This function can be used in case a start and end date is known for dosing. This function fills down the dates so each date between start and end is placed on a separate row. Subsequently the dataset can be used to merge with available date information and impute missing dates.

Usage

```
fill_dates(data, start, end, tau = 1, repdat = 1)
```

Arguments

data	data frame for which the dates should be filled down
start	character identifying the start date (as date format) within the data frame
end	character identifying the end date (as date format) within the data frame
tau	integer with the tau in days (e.g. 2 for dosing every other day)
repdat	integer with repeats per day (e.g. 2 in case of twice daily dosing)

Value

a data frame with filled out dates

Author(s)

Richard Hooijmaijers

Examples

```
dfrm <- data.frame(ID=1:2,first=as.Date(c("2016-10-01","2016-12-01"), "%Y-%m-%d"),
                   last=as.Date(c("2016-10-03","2016-12-02"), "%Y-%m-%d"))
fill_dates(dfrm, "first", "last")
fill_dates(dfrm, "first", "last", 2, 3)
```

`filterr` *Filter data with logging of results*

Description

This function is a wrapper around dplyr::filter. Additional actions are performed on the background to log the information of the filter action, and info regarding the step is printed.

Usage

```
filterr(.data, ..., comment = "")
```

Arguments

`.data`	the data frame for which the filter should be created
`...`	arguments passed to dplyr::filter
`comment`	character with the reason of filtering used in log file

Details

The function can be used to keep track of records that are omitted in the data management process. In general one would like to keep all records from the source database (and use flags instead to exclude data) but in cases where this is not possible it is important to know what records are omitted and for which reason. Every time the function is used it creates a records in in a log file which can be used in the documentation.

Value

a filtered data frame

Author(s)

Richard Hooijmaijers

See Also

dplyr::filter

Examples

```
# For full trace-ability of source data, no pipes
# are preferred
dat1 <- filterr(Theoph,Subject==1)
dat2 <- Theoph |> filterr(Subject==2)
# Show what is being logged
get_log()$filterr_nfo
```

`flag_outliers` *Creates a flag for outlying values*

Description

This function creates a flag identifying the outliers in a vector

Usage

```
flag_outliers(var, type = "boxstat")
```

Arguments

`var`	numeric vector that should be checked for outliers
`type`	character with the type of test to perform for outliers (currently only the "boxstats" is available that uses the boxplot method)

Value

a numeric vector the same length as `var` with either 0 (no outlier) or 1 (outlier)

Author(s)

Richard Hooijmaijers

Examples

```
dfrm <- data.frame(a = 1:10, b = c(1:9,50))
flag_outliers(dfrm$a)
flag_outliers(dfrm$b)
```

get_log *Retrieve log objects*

Description

Returns one or more dataframes with log information related to function like filterr, left_joinr, cmnt, srce and read_data

Usage

```
get_log()
```

Value

a named list of dataframes

Author(s)

Richard Hooijmaijers

Examples

```
xldat <- readxl::readxl_example("datasets.xlsx")
xlin  <- read_data(xldat, comment="read test")
get_log()
```

get_script *Get the current script name (either interactive Rstudio, markdown or batch script)*

Description

Get the current script name (either interactive Rstudio, markdown or batch script)

Usage

```
get_script(base = TRUE, noext = TRUE)
```

Arguments

base	logical indicating if the basename should be returned (without path)
noext	logical indicating if the file extension should be omitted

Value

character with the current script name

Author(s)

Richard Hooijmaijers

`impute_covar` *Impute missing covariates*

Description

The function will impute all NA values with either a given statistic (e.g. median) or with the largest group

Usage

```
impute_covar(var, uniques = NULL, type = "median", verbose = FALSE)
```

Arguments

`var`	vector with the items that should be imputed
`uniques`	vector that defines unique records to enable calculation of stats on non duplicate values
`type`	character of length one defining the type of statistics to perform for imputation (see details)
`verbose`	logical indicating if additional information should be given

Details

The function can be used to impute continuous or categorical covariates. In case continuous covariates the type argument should be a statistic like median or mean. In case a categorical covariate is used, the type should be set to 'largest' in which case the category that occurs most is used. In case multiple values occur most, the last encountered is used.

Value

a vector where missing values are imputed

Author(s)

Richard Hooijmaijers

Examples

```
dfrm  <- data.frame(num1 = c(NA,110))
impute_covar(dfrm$num1,type="median")
```

`impute_dose` *Imputes dose records using ADDL and II by looking forward and backwards*

Description

This function imputes dose records by looking at the missing doses between available records based on a given II value

Usage

```
impute_dose(data, id, datetime, amt = "AMT", ii = "II", thr = 50, back = TRUE)
```

Arguments

`data`	data frame to perform the calculations on
`id`	character identifying the id variable within the data frame
`datetime`	character identifying the date/time variable (POSIXct) within the data frame
`amt`	character identifying the AMT (amount) variable within the data frame
`ii`	character identifying the II (Interdose Interval) variable within the data frame
`thr`	numeric indicating the threshold percentage for imputation (see details)
`back`	logical indicating if the time for imputed doses should be taken from the previous record (TRUE) or the next (FALSE)

Details

The function fills in the doses by looking at the time difference and II between all records. Be aware that this can result in unexpected results in a few cases, so results should always be handled with care. The function will determine if a dose should be imputed based on the 'thr' value. For each dose, the function determines the percentage difference from the previous dose based on the II value. In case the expected difference is above the threshold value, imputation will be done.

Value

a data frame with imputed dose records

Author(s)

Richard Hooijmaijers

Examples

```
dfrm <- data.frame(id=c(1,1), dt=c(Sys.time(),Sys.time()+ 864120),
                   II=c(24,24),AMT=c(10,10))
impute_dose(dfrm,"id","dt")
```

`left_joinr` *Perform a left join of two data frames with logging of results*

Description

This function is a wrapper around dplyr::left_join. Additional actions are performed on the background to log the information of the join action, and info regarding the step is printed.

Usage

```
left_joinr(x, y, by = NULL, ..., comment = "", keepids = FALSE)
```

Arguments

`x, y`	a pair of data frames used for joining
`by`	character vector of variables to join by
`...`	additional arguments passed to dplyr::left_join
`comment`	information for the reason of merging
`keepids`	logical indicating if merge identifiers should be available in output data (for checking purposes)

Details

The function can be used to keep track of records that are available after a join in the data management process. Joining of data could lead to unexpected results, e.g. creation of cartesian product or loosing data. Therefore it is important to know what the result of a join step is. Every time the function is used it creates a records in in a log file which can be used in the documentation.

Value

a joined data frame

Author(s)

Richard Hooijmaijers

See Also

dplyr::left_join

Examples

```
dose  <- data.frame(Subject = unique(Theoph$Subject),
                    dose = sample(1:3,length(unique(Theoph$Subject)),
                                  replace = TRUE))
dose2 <- dose[dose$Subject%in%1:6,]
# Preferred to explicitly list by
dat1 <- left_joinr(Theoph, dose, by="Subject")
# The base R pipe is preferred for better logging of source data
dat2 <- Theoph |> left_joinr(dose, by="Subject")
# Avoid long pipes before function for readability in log. e.g dont:
dat3 <- Theoph |> dplyr::mutate(ID=3) |> dplyr::bind_cols(X=3) |>
  left_joinr(dose, by="Subject")
# Show what is being logged
get_log()$joinr_nfo
```

`log_df` *Create information for all functions that log actions*

Description

This function creates a table including information on functions that log information such as `filterr`, `left_joinr` and `read_data`

Usage

```
log_df(
  log,
  type,
  coding = FALSE,
  ret = "dfrm",
  capt = NULL,
  align = NULL,
  size = "\\footnotesize",
  ...
)
```

Arguments

`log`	list with logged information typically obtained with get_log
`type`	character with the type of info that should be taken from log (either "filterr_nfo","joinr_nfo" or "read_nfo")
`coding`	logical indicating if the coding (within `filterr`) should be displayed
`ret`	a character vector to define what kind of output should be returned (either "dfrm", "tbl", "file")
`capt`	character with the caption of the table (not used in case data frame is returned)
`align`	alignment of the table passed to general_tbl (not used in case data frame is returned)

size	character with font size as for the table general_tbl
...	additional arguments passed to general_tbl

Details

This function generates information for function that logs information. It attempts to find a good alignment and caption, mainly for outputting to a table. It is possible to set your own captions and alignment, take into account that the alignment differs per type and in case the coding argument is changed.

Value

function creates a PDF file or returns a data frame

Author(s)

Richard Hooijmaijers

Examples

```
dat1 <- filterr(Theoph,Subject==1)
dat2 <- Theoph |> filterr(Subject==2)
log_df(get_log(), "filterr_nfo")
```

---

make_readonly *Sets the read-only attribute for all files available within a folder*

Description

This function will change the file attributes so only read access is set

Usage

```
make_readonly(x)
```

Arguments

x	character of length 1 with the path that contains files or character vector with filenames to be set to read-only

Details

This function will attempt to set a read-only attributes on files. This is either done through system commands such as `attrib` for windows and `chmod` for linux (444). With the latter take into account possible issues with (sudo) rights on files. In case x is a directory, the function will set readonly attribute for all files in the folder (and recurse into all subfolders!).

Value

nothing is returned, only system commands are issued

Author(s)

Richard Hooijmaijers

Examples

```
## Not run:
  tmpf <- tempfile(fileext = ".txt")
  cat("test",file=tmpf)
  make_readonly(tmpf)

## End(Not run)
```

---

num_lump *Perform lumping of numerical values*

Description

This function is mainly a wrapper for forcats::fct_lump but applied on numeric variables. Furthermore there is the option to use uniques to determine small categories for instance on individual level

Usage

```
num_lump(x, lumpcat = 99, uniques = NULL, prop = NULL, min = NULL, ...)
```

Arguments

x	numeric vector with the items that should be lumped
lumpcat	the category in which the lumped levels should be added (see details)
uniques	vector that defines unique records to enable lumping on non duplicate values
prop	numeric with the threshold proportions for lumping
min	numeric with the min number of times a level should appear to not lump
...	additional arguments passed to forcats::fct_lump_min and/or forcats::fct_lump_prop

Details

The argument lumpcat is the level in which lumped values should appear and can be one of the following:

- numeric with the category number to set the levels to
- character specifying "largest" to select the largest category (selected before lumping)
- named vector to set the 'algorithm' for instance: c('5'='3', '4'='6') to set category 5 to 3 and 4 to 6 when these categories need lumping

Value

vector with the lumping applied

Author(s)

Richard Hooijmaijers

Examples

```
dfrm <- data.frame(id = 1:30, cat = c(rep(1,8),rep(2,13), rep(3,4),rep(4,5)))
num_lump(x=dfrm$cat, lumpcat=99, prop=0.15)
```

---

output_data *export R data for NONMEM modeling*

Description

This function exports data for NONMEM modeling analyses including options that are frequently necessary to adapt.

Usage

```
output_data(
  x,
  csv = NULL,
  xpt = NULL,
  attr = NULL,
  verbose = TRUE,
  maxdig = 6,
  tonum = TRUE,
  firstesc = NULL,
  readonly = FALSE,
  overwrite = TRUE,
  ...
)
```

Arguments

x	data frame to be exported.
csv	character with the name of the csv file to generate
xpt	character with the name of the xpt file to generate
attr	character with the name of the rds file to generate
verbose	logical indicating if additional information should be written to the console
maxdig	numeric with the maximum number of decimals for numeric variables to be in output (see details)

tonum	logical indicating if all variables should be made numeric (standard for NONMEM input files)
firstesc	character with escape character for first variable, used to exclude row in NONMEM
readonly	logical indicating if the output csv file should be made readonly
overwrite	logical indicating if (all) output should be overwritten or not
...	Arguments passed to write.csv

Details

In case tonum is TRUE, all variables will be made numeric and Inf values will be set to NA (all NA values will be set to a dot). The rounding set in maxdig will only be done in case tonum is set to TRUE. For xpt files, the name of the object to export is used as the name of the dataset inside the xpt file (e.g. output_data(dfrm,xpt='dataset.xpt') will result in an xpt file named 'dataset.xpt' with one dataset named 'dfrm').

Value

a data frame is written to disk

Author(s)

Richard Hooijmaijers

See Also

write.csv

Examples

```
data(Theoph)
out_file <- tempfile(fileext = ".csv")
output_data(Theoph, csv = out_file, tonum = FALSE)
```

---

plot_vars *Creates different kind of plots for variables within a dataset using ggplot2*

Description

This function creates histograms for numeric values and barcharts for character or factor variables In case there are more then 10 unique values it will list the first 10 unique values in a 'text' plot

Usage

```
plot_vars(dfrm, vars = names(dfrm), ppp = 16, ...)
```

Arguments

dfrm	data frame that should be plotted
vars	character vector with the variables for which plots should be generated
ppp	number plots per page
...	additional arguments passed to patchwork::wrap_plots() (e.g. ncol/nrow)

Value

a ggplot/patchwork object with all variables plotted visualized

Author(s)

Richard Hooijmaijers

Examples

```
plot_vars(Theoph)
```

---

read_data *read external data with logging of results*

Description

This function reads external data with support for file types that are most commonly used in clinical and pre-clinical data, and provide manual functions for less common types

Usage

```
read_data(
  file,
  manfunc = NULL,
  comment = "",
  verbose = TRUE,
  ascii_check = "xls",
  ...
)
```

Arguments

file	character with the name of the file to read (see details for more information)
manfunc	character with the manual function to use to read data (can have the form "package::function")
comment	character with comment/information for the data that was read-in
verbose	logical indicating if information regarding the data that is read is printed in console

ascii_check	character of length one defining if the data that has been read in should be checked for valid ASCII characters (see details)
...	Additional arguments passed to the read data functions. This can be used to add arguments to for instance read.table or read_excel or for the function defined in manfunc

Details

The function reads in data, and uses the file extension to select the applicable function for reading. Below is a list of extensions that are recognized and the corresponding function that is used to read the data:

- sas7bdat: haven::read_sas
- xpt: haven::read_xpt
- xls/xlsx: readxl::read_excel
- prn/par: read.table
- csv: read.csv

The prn and par file formats are basically space delimited files but with some specifics for modeling software (e.g. prn is NONMEM input file with header starting with '#' and par is NONMEM output file as defined in $TABLE). This function can be used to read any type of data by using the manfunc argument. Any function available in R can be used here and even user written functions (see example section). This argument has precedence over the recognition of extensions. This means for instance that a CSV file can also be read-in using a different function (e.g. using `data.table::fread`). This flexibility is build in to ensure all possible data can be read in using this single function. This is mainly important for documentation purposes, to ensure all used data can be logged and documented. The data can be checked for valid ASCII characters using the "ascii_check" argument. By default this is done for excel files with extension xls or xlsx (ascii_check="xls") other options are "none" to never perform a check or "all" to perform a check regardless of the way it is read in. The default is chosen as it is likely that excel files are created manually and could therefore include non ASCII characters, and because it puts additional overhead on function otherwise.

Value

data frame containing a representation of the data in the file

Author(s)

Richard Hooijmaijers

Examples

```
# For a known filetype you can use:
dat <- read_data(paste0(R.home(),"/doc/CRAN_mirrors.csv"))

# We can use the arguments from the underlying package that does the reading
xmpl <- system.file("example/Attr.Template.xlsx",package = "amp.dm")
dat  <- read_data(xmpl, range="A2:B3")
```

```
# In case we get a file format not directly supported by the function
# we can use the manfunc to use another function
sav <- system.file("examples", "iris.sav", package = "haven")
dat <- read_data(sav,manfunc = "haven::read_sav")

# It is also possible to write your own function that reads data
# (as long as it returns a data.frame or tibble), e.g.:
read_nd <- function(x) read.csv(x) |> dplyr::distinct(ID, .keep_all = TRUE)
xmpl    <- system.file("example/NM.theoph.V1.csv", package = "amp.dm")
dat     <- read_data(xmpl,manfunc = "read_nd")
```

`session_tbl` *Create information for R session*

Description

This function creates a latex table or data frame with information from the R session (sessionInfo() and Sys.info())

Usage

```
session_tbl(
  ret = "tbl",
  capt = "Session info",
  align = "lp{8cm}",
  size = "\\footnotesize",
  incscript = FALSE,
  ...
)
```

Arguments

`ret`	a character vector to define what kind of output should be returned (either "dfrm", "tbl", "file")
`capt`	character with the caption of the table (not used in case data frame is returned)
`align`	alignment of the table passed to general_tbl (not used in case data frame is returned)
`size`	character with font size as for the table general_tbl
`incscript`	logical indicating if the name of the script should be included (using get_script)
`...`	additional arguments passed to general_tbl

Details

This function can be used to create a table with the most important information of a R session, the user that is running the R session and the current date/time

Value

a data frame, code for table or nothing in case a PDF file is created

Author(s)

Richard Hooijmaijers

Examples

```
session_tbl()
```

`srce` *Add source information to environment to present in documentation*

Description

Adds the source of variables into package environment, which can be used in code chunks at the applicable locations and easily added to documentation afterwards

Usage

```
srce(var, source, type = "c")
```

Arguments

`var`	unquoted string with the variable for which the source should be defined
`source`	unquoted strings with the source(s) used for var (see example)
`type`	character with the type of variable can be either 'c' (copied) or 'd' (derived)

Value

no return value, called for side effects

Author(s)

Richard Hooijmaijers

Examples

```
# variable AMT copied from Dose variable in Theoph data frame
srce(AMT,Theoph.Dose)
# variable BMI derived from WEIGHT variable in wt data frame
# and HEIGHT variable in ht data frame
srce(BMI,c(wt.WEIGHT,ht.HEIGHT),'d')
get_log()$srce_nfo
```

stats_df *Calculate basic statistics on data frame*

Description

This function creates a latex table or data frame with the basic statistics of a data frame.

Usage

```
stats_df(
  data,
  missingval = -999,
  ret = "tbl",
  capt = "Statistics data frame",
  align = "p{2cm}p{1cm}p{1cm}p{4cm}p{1.7cm}p{1.7cm}p{0.8cm}p{1.3cm}",
  size = "\\footnotesize",
  ...
)
```

Arguments

data	a data frame for which the overview should be created
missingval	numeric with the value that indicates missing values, if NULL no missings will be recorded
ret	a character vector to define what kind of output should be returned (either "dfrm", "tbl", "file")
capt	character with the caption of the table (not used in case data frame is returned)
align	alignment of the table passed to general_tbl (not used in case data frame is returned)
size	character with font size as for the table general_tbl
...	additional arguments passed to general_tbl

Details

This function can be used to create a table with basic statistics of a data frame. The function will list the min, max, number of NA/missing values, number of unique categories and type of variable of all data items within a data frame. In case a data item has less than 10 unique categories, it will list the unique values. The main reason to use this function is to create a structured table with statistics of a data frame to be included in documentation.

Value

either tex code for table a data frame

Author(s)

Richard Hooijmaijers

Examples

```
stats_df(Theoph)
```

`time_calc` *Create time variables for usage in NONMEM analyses*

Description

This function calculates the most important time variables based on multiple variables in a data frame.

Usage

```
time_calc(
  data,
  datetime,
  evid = NULL,
  addl = NULL,
  ii = NULL,
  amt = "AMT",
  id = "ID",
  dig = 2
)
```

Arguments

`data`	data frame to perform the calculations on
`datetime`	character identifying the date/time variable (POSIXct) within the data frame
`evid`	character identifying the event ID (EVID) within the data frame
`addl`	character identifying the additional dose levels (ADDL) within the data frame
`ii`	character identifying the interdose interval (II) within the data frame
`amt`	character identifying the amount variable (only needed if `evid` is not provided)
`id`	character identifying the ID or subject variable
`dig`	numeric indicating with how many digits the resulting times should be available

Details

The function calculates the TIME, TALD (time after last dose) and TAFD (time after first dose). The different time variables are calculated in hours, where a POSIXct for the datetime variable is expected. The function takes into account ADDL and II records when provided, to correctly identify the TALD. One must be cautious however when having ADDL/II and a complex dosing schedule (e.g. alternating dose schedules, more than 1 dose per day, multiple compounds administration). In general it is good practice to double check the output for multiple subjects in case of complex designs including ADDL/II.

Value

a data frame with the calculated times

Author(s)

Richard Hooijmaijers

Examples

```
dfrm <- data.frame(ID=rep(1,5), dt=Sys.time() + c(0,8e+5,1e+6,2e+6,3e+6),
                   AMT=c(NA,10,NA,0,NA), II=rep(24,5),EVID=c(0,1,0,1,0))
time_calc(dfrm,"dt","EVID")
```

---

weight_height *Calculates different weight variables*

Description

This function calculates different variables based on weight and height and conversion from or to kilograms

Usage

```
weight_height(wt = NULL, ht = NULL, sex = NULL, bmi = NULL, type = "bmi")
```

Arguments

wt	vector with weight values, in either kg or lb depending on the type (see details)
ht	vector with height values in cm (see details)
sex	vector with SEX values (Where female is defined as a value of 1)
bmi	vector with BMI values (see details)
type	character with the type to be used for the calculations (see details)

Details

Currently the following types are defined within the function:

Convert units:

- "kg-lb" : Convert units from kg to lb using the formula

$$\text{Weight (kg)} = \text{Weight (lb)} \cdot 2.20462262$$

- "lb-kg" : Convert units from lb to kg using the formula

$$\text{Weight (lb)} = \frac{\text{Weight (kg)}}{2.20462262}$$

Body mass index:

- "bmi" : Calculates body mass index (BMI) using the standard formula (Quetelet1842,

$$\text{BMI} = \frac{\text{Weight (kg)}}{\text{Height (m)}^2}$$

Body Surface Area:

- "bsa": Body Surface Area, according to Gehan and Georg,

$$\text{BSA} = \exp -3.751 \cdot \text{Height (cm)}^{0.422} \cdot \text{Weight (kg)}^{0.515}$$

- "bsa2": Body Surface Area, according to DuBois and DuBois,

$$\text{BSA} = 0.007184 \cdot \text{Height (cm)}^{0.725} \cdot \text{Weight (kg)}^{0.425}$$

- "bsam": Body Surface Area, according to Mosteller,

$$\text{BSA} = \sqrt{\frac{\text{Weight (kg)} \cdot \text{Height (cm)}}{3600}}$$

- "bsah": Body Surface Area, according to Haycock,

$$\text{BSA} = 0.024265 \cdot \text{Height (cm)}^{0.3964} \cdot \text{Weight (kg)}^{0.5378}$$

- "bsal": Body Surface Area in normal-weight and obese adults up to 250 kg, according to Livingston,

$$\text{BSA} = 0.1173 \cdot \text{Weight (kg)}^{0.6466}$$

Fat free mass:

- "ffmj": Fat free mass, according to Janmahasatian:

$$\text{FFM} = \frac{9270 \cdot \text{Weight (kg)}}{k + (l \cdot \text{BMI})}$$

, where k is 6680 for males and 8780 for females and l is 216 for males and 244 for females.

- "ffms": Fat free mass in Indian patients, according to Sinha:

$$\text{FFM} = \frac{9270 \cdot \text{Weight (kg)}}{k \cdot l \cdot \text{BMI}^{0.28}}$$

, where k is 6680 for males and 8780 for females and l is 0.77 for males and 0.70 for females.

Lean body mass:

- "lbmb" : Calculates lean body mass (LBM), according to Boers:

$$\text{LBM} = k \cdot \text{Weight (kg)} + l \cdot \text{Height (cm)} - m$$

, where k is 0.407 for males and 0.252 for females, and l is 0.267 for males and 0.473 for females, and m is 19.2 for males and 48.3 for females.

- "lbmj" : Calculates lean body mass (LBM), according to James:

$$\text{LBM} = k \cdot \text{Weight (kg)} - l \cdot \left(\frac{\text{Weight (kg)}}{\text{Height (cm)}}\right)^2$$

, where k is 1.10 for males and 1.07 for females, and l is 128 in males and 148 in females.

- "lbmp" : Calculates lean body mass (LBM) for children up to 14 years, according to Peters:

$$\text{LBM} = 3.8 \cdot 0.0215 \cdot \text{Weight (kg)}^{0.6469} \cdot \text{Height (cm)}^{0.7236}$$

Predicted Normal Weight:

- "pnw" : Calculates the Predicted Normal Weight for obese patient, according to Duffull:

$$\text{PNWT} = k \cdot \text{Weight (kg)} - l \cdot \text{Height (cm)} \cdot \text{BMI} - m$$

, where k is 1.57 for males and 1.75 for females, and l is 0.0183 for males and 0.0242 for females, and m is 10.5 for males and 12.6 for females.

Value

a vector with calculated values

Author(s)

Richard Hooijmaijers

Examples

```
tmp <- data.frame(id=1,WT=runif(3,70,120),HT=runif(3,160,220))
weight_height(wt=tmp$WT,ht=tmp$HT,type="bmi")
# example for use in dplyr
tmp |> dplyr::mutate(BMI = weight_height(wt=WT,ht=HT,type="bmi"))
```

Index