Package: scicalc (via r-universe)

February 10, 2025

Title Scientific Calculations for Quantitative Clinical Pharmacology and Pharmacometrics Analysis

Version 0.1.2

Description Utility functions helpful for reproducible scientific calculations.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports arrow, checkmate, digest, dplyr, fs, haven, magrittr, readr, readxl, rlang, stats, stringr

Suggests knitr, rmarkdown, testthat (>= 3.0.0), ggplot2, here, purrr, pzfx, tools, tidyr

Config/testthat/edition 3

VignetteBuilder knitr

URL https://a2-ai.github.io/scicalc

Config/pak/sysreqs cmake make libicu-dev libssl-dev libx11-dev zlib1g-dev

Repository https://a2-ai.r-universe.dev

RemoteUrl https://github.com/a2-ai/scicalc

RemoteRef HEAD

RemoteSha bc32814c8b9e9ce17ca464e1c403b274edfd546f

Contents

`bbmi` *Calculates Baseline Body Mass Index based on Weight and Height*

Description

Calculates Baseline Body Mass Index based on Weight and Height

Usage

```
bbmi(weight, height)
```

Arguments

`weight`	weight of subject (kg)
`height`	height of subject (cm)

Value

the bBMI value (kg m^(-2))

Examples

```
b <- bbmi(80.56, 167)

df <- data.frame(
"WT" = c(80.56, 71.53, 81.04, 70.17),
"HT" = c(167, 161, 163, 164)
)
df <- dplyr::mutate(df, bbmi = bbmi(WT, HT))
```

`bhfc` *Calculates hepatic function criteria*

Description

Calculates hepatic function criteria

Usage

```
bhfc(ast, ulnast, bili, ulnbili)
```

Arguments

`ast`	Aspartate aminotransferase concentration (IU/L)
`ulnast`	Upper limit of normal AST (IU/L), typically 33
`bili`	bilirubin concentration (mg/dL)
`ulnbili`	Upper limit of normal BILI (mg/dL), typically 1.2

Value

category of hepatic function

Examples

```
bhfc(15, 33, 0.6, 1.2)

df <- data.frame(
  "ID" = c(1, 1, 1, 1, 2, 2, 2, 2),
  "SEX" = c("F", "F", "F", "F", "M", "M", "M", "M"),
  "RACE" = c("WHITE", "WHITE", "WHITE", "WHITE", "BLACK", "BLACK", "BLACK", "BLACK"),
  "AGE" = c(24, 24, 24, 24, 22, 22, 22, 22),
  "CREAT" = c(1, 1, 1, 1, 4, 4, 4, 4),
  "WEIGHT" = c(70, 70, 70, 70, 65, 65, 65, 65),
  "AST" = c(15, 15, 15, 15, 23, 23, 23, 23),
  "ULNAST" = c(33, 33, 33, 33, 33, 33, 33, 33),
  "BILI" = c(1, 1, 1, 1, 0.4, 0.4, 0.4, 0.4),
  "ULNBILI" = c(1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2)
)

df <- df %>%
  dplyr::group_by(ID) %>%
  dplyr::mutate(BHFC = bhfc(AST, ULNAST, BILI, ULNBILI))
```

---

brfc *Calculates renal impairment categories based on CrCL*

Description

Calculates renal impairment categories based on CrCL

Usage

```
brfc(crcl)
```

Arguments

`crcl`	creatinine clearance rate (mL/min)

Value

integer renal impairment category

Examples

```
brfc(crcl(FALSE, 20, 10, 70))

df <- data.frame(
  "ID" = c(1, 1, 1, 1, 2, 2, 2, 2),
  "SEX" = c("F", "F", "F", "F", "M", "M", "M", "M"),
  "RACE" = c("WHITE", "WHITE", "WHITE", "WHITE", "BLACK", "BLACK", "BLACK", "BLACK"),
  "AGE" = c(24, 24, 24, 24, 22, 22, 22, 22),
  "CREAT" = c(1, 1, 1, 1, 4, 4, 4, 4),
  "WEIGHT" = c(70, 70, 70, 70, 65, 65, 65, 65)
```

```
)

df <- df %>%
  dplyr::group_by(ID) %>%
  dplyr::mutate(
    CRCL = crcl(is_female(SEX), AGE, CREAT, WEIGHT),
    BRFC = brfc(CRCL)
  )
```

`bsa` *Calculates Body Surface Area based on Weight and Height using the method specified. Default is Dubois.*

Description

Calculates Body Surface Area based on Weight and Height using the method specified. Default is Dubois.

Usage

```
bsa(weight, height, method = "Dubois")
```

Arguments

`weight`	weight of a subject (kg)
`height`	height of a subject (cm)
`method`	String to dictate which equation to use. Dubois or Mosteller.

Value

bsa (m^2)

Examples

```
bsa(70, 170)
bsa(70, 170, method = "Mosteller")
bsa(70, 170, method = "Dubois")
```

`categorize` *Converts continuous variable into factor categories.*

Description

Converts continuous variable into factor categories.

Usage

```
categorize(continuous_var, nbins = 4, units = "", type = 7, digits = 1)
```

Arguments

`continuous_var`	continuous variable data
`nbins`	number of bins to break data into, default is 4
`units`	string, optional units string to add to labels of categorized data
`type`	type argument for stats::quantile, default is 7
`digits`	number of digits to round quantile breaks to for labels, default is 1

Value

a vector of categorized data as factor

Examples

```
x <- rnorm(1000, mean = 10, sd = 5)
xc <- categorize(x, nbins = 5)
```

`check_for_unique_units` *Gives a TRUE/FALSE for if the Parameters have only 1 associated unit*

Description

Gives a TRUE/FALSE for if the Parameters have only 1 associated unit

Usage

```
check_for_unique_units(params, units)
```

Arguments

`params`	a column from a dataset with lab parameters
`units`	a column from a dataset with units associated with those parameters

Value

a boolean

Examples

```
df <- data.frame(
  PARAM = c(
    "ALB","ALT","AST","CR","TBIL",
    "ALB","CR","TBIL","ALT","AST"),
  UNIT = c(
    "g/L","U/L","U/L","umol/L","umol/L",
    "U/L"," mol/L"," mol/L","IU/L","IU/L")
)
check_for_unique_units <- get_unique_units_df(df$PARAM, df$UNIT)
```

---

ckdepi_2009_egfr — *Calculates Estimated Glomerular Filtration Rate based on Sex, Race, Age, and Creatinine levels based on the CKDEPI 2009 equation*

Description

Calculates Estimated Glomerular Filtration Rate based on Sex, Race, Age, and Creatinine levels based on the CKDEPI 2009 equation

Usage

```
ckdepi_2009_egfr(sexf, raceb, age, creat)
```

Arguments

`sexf`	boolean value of sex Female: TRUE, Male: FALSE
`raceb`	boolean value of Race == Black: Black: TRUE, Other: FALSE
`age`	age of subject (years)
`creat`	creatinine levels of subject (mg/dL)

Value

the eGFR value (mL/min/1.73m2)

Examples

```
e <- ckdepi_2009_egfr(TRUE, TRUE, 24, 1)

df <- data.frame(
   "SEXF" = c(TRUE, FALSE, TRUE, FALSE),
   "RACEB" = c(FALSE, FALSE, TRUE, FALSE),
   "AGE" = c(24, 24, 23, 24),
```

```
   "CREAT" = c(1, 1, 2, 1)
   )
df <- dplyr::mutate(df, egfr = ckdepi_2009_egfr(SEXF, RACEB, AGE, CREAT))
```

`ckdepi_2021_egfr` *Calculates eGFR using the CKDEPI 2021 creatinine equation*

Description

Calculates eGFR using the CKDEPI 2021 creatinine equation

Usage

```
ckdepi_2021_egfr(sexf, age, creat)
```

Arguments

`sexf`	boolean value of sex Female: TRUE, Male: FALSE
`age`	age of subject (years)
`creat`	creatinine levels of subject (mg/dL)

Value

the eGFR value (mL/min/1.73m2)

Examples

```
e <- ckdepi_2021_egfr(TRUE, 24, 1)

df <- data.frame(
   "SEXF" = c(TRUE, FALSE, TRUE, FALSE),
   "RACEB" = c(FALSE, FALSE, TRUE, FALSE),
   "AGE" = c(24, 24, 23, 24),
   "CREAT" = c(1, 1, 2, 1)
   )
df <- dplyr::mutate(df, egfr = ckdepi_2021_egfr(SEXF, AGE, CREAT))
```

`ckdepi_2021_egfr_cystatin` *Calculates eGFR with CKDEPI 2021 cystatin equation*

Description

Calculates eGFR with CKDEPI 2021 cystatin equation

Usage

```
ckdepi_2021_egfr_cystatin(sexf, age, creat, cystc)
```

Arguments

`sexf`	a boolean representing if the patient is female.
`age`	age of patient in years
`creat`	serum creatinine levels in mg/dL.
`cystc`	serum cystatin C levels in mg/L.

Value

eGFR in mL/min/1.73 m^2

Examples

```
e <- ckdepi_2021_egfr_cystatin(TRUE, 24, 1, 2)

df <- data.frame(
   "SEXF" = c(TRUE, FALSE, TRUE, FALSE),
   "RACEB" = c(FALSE, FALSE, TRUE, FALSE),
   "AGE" = c(24, 24, 23, 24),
   "CREAT" = c(1, 1, 2, 1),
   "CYSTC" = c(0.4, 0.8, 1, 2)
   )
df <- dplyr::mutate(df, egfr = ckdepi_2021_egfr_cystatin(SEXF, AGE, CREAT, CYSTC))
```

`crcl` *Calculates Creatinine clearance with Cockcroft-Gault equation*

Description

Calculates Creatinine clearance with Cockcroft-Gault equation

Usage

```
crcl(sexf, age, creat, weight)
```

Arguments

sexf	bool of sex of subject. Female: True, Male: False
age	age of subject (years)
creat	serum creatinine levels (mg/dL)
weight	weight of subject (kg)

Value

CrCl (mL/min)

Examples

```
crcl(FALSE, 20, 10, 70)

df <- data.frame(
  "ID" = c(1, 1, 1, 1, 2, 2, 2, 2),
  "SEX" = c("F", "F", "F", "F", "M", "M", "M", "M"),
  "RACE" = c("WHITE", "WHITE", "WHITE", "WHITE", "BLACK", "BLACK", "BLACK", "BLACK"),
  "AGE" = c(24, 24, 24, 24, 22, 22, 22, 22),
  "CREAT" = c(1, 1, 1, 1, 4, 4, 4, 4),
  "WEIGHT" = c(70, 70, 70, 70, 65, 65, 65, 65)
)

df <- df %>%
  dplyr::group_by(ID) %>%
  dplyr::mutate(CRCL = crcl(is_female(SEX), AGE, CREAT, WEIGHT))
```

---

create_dir *Creates the directory if it doesn't exist*

Description

Creates the directory if it doesn't exist

Usage

```
create_dir(path)
```

Arguments

path	path of directory to be created

Value

Nothing

Examples

```
## Not run:
create_dir("derived/data/test")

## End(Not run)
```

`cv` *Computes the coefficient of variation of input vector.*

Description

Computes the coefficient of variation of input vector.

Usage

```
cv(x, na.rm = FALSE)
```

Arguments

`x`	Input vector to compute CV for.
`na.rm`	boolean to remove NA. default is FALSE

Value

CV of x. Standard deviation divided by mean. If you want % you'll need to multiply by 100

Examples

```
cv(c(1, 2, 1, 1, 2, 1, 2, 3))
```

`dubois_bsa` *Calculates Body Surface Area based on Weight and Height using Dubois Dubois equation*

Description

Calculates Body Surface Area based on Weight and Height using Dubois Dubois equation

Usage

```
dubois_bsa(weight, height)
```

Arguments

`weight`	weight of subject (kg)
`height`	height of subject (cm)

Value

the body surface area (m^2)

Examples

```
#' b <- dubois_bsa(80.56, 167)

df <- data.frame(
"WT" = c(80.56, 71.53, 81.04, 70.17),
"HT" = c(167, 161, 163, 164)
)
df <- dplyr::mutate(df, bsa = dubois_bsa(WT, HT))
```

---

egfr *Calculates eGFR based on the method specified*

Description

Calculates eGFR based on the method specified

Usage

```
egfr(sexf, raceb, age, creat, cystc, height, method = "CKDEPI 2009")
```

Arguments

sexf	a boolean representing if the patient is female.
raceb	a boolean representing if the patient is black.
age	the age of a patient in years.
creat	the serum creatinine levels in mg/dL.
cystc	the cystatin C levels in mg/L - only used in CKDEPI 2021 cystatin method
height	the height of a patient in cm.
method	a string specifying the method to use. Available options are "CKDEPI 2009", "MDRD", "CKDEPI 2021", "Schwartz".

Value

the eGFR calculated based on method.

Examples

```
e <- egfr(TRUE, TRUE, 24, 1, "CKDEPI 2009")

df <- data.frame(
   "SEXF" = c(TRUE, FALSE, TRUE, FALSE),
   "RACEB" = c(FALSE, FALSE, TRUE, FALSE),
   "AGE" = c(24, 24, 23, 24),
   "CREAT" = c(1, 1, 2, 1)
   )
df <- dplyr::mutate(df, egfr = egfr(SEXF, RACEB, AGE, CREAT, "CKDEPI 2009"))
```

---

ethnicn *Takes character input and returns standard yspec numeric value for Ethnic*

Description

Takes character input and returns standard yspec numeric value for Ethnic

Usage

```
ethnicn(ethnicc)
```

Arguments

ethnicc	Ethnic character

Value

the standard yspec numeric value for the inputted Ethnic character

Examples

```
ethnicn("HISPANIC OR LATINO") # 1

ethnicn("NOT HISPANIC OR LATINO") # 0

ethnicn("UNKNOWN") # -999
```

geom_cv — *Computes the geometric CV of a vector x*

Description

Computes the geometric CV of a vector x

Usage

```
geom_cv(x, na.rm = FALSE)
```

Arguments

x	vector of data you want the geometric CV of.
na.rm	boolean to remove NA from vector. Default is FALSE

Value

the geometric CV of the input vector x

Examples

```
geom_cv(c(1, 2, 3, 2, 1))
```

geom_mean — *Computes the geometric mean of a vector.*

Description

Computes the geometric mean of a vector.

Usage

```
geom_mean(x, na.rm = FALSE)
```

Arguments

x	vector to compute geometric mean of
na.rm	boolean to remove NA from vector in calcualtion. Default is False

Value

geometric mean of input vector x

Examples

```
geom_mean(c(1, 2, 3, 2, 1))
```

`geom_sd` *Computes the geometric standard deviation of a vector x.*

Description

Computes the geometric standard deviation of a vector x.

Usage

```
geom_sd(x, na.rm = FALSE)
```

Arguments

`x`	The vector of data you want the geometric sd of.
`na.rm`	a boolean to remove NA values. Default is False

Value

the geometric standard deviation of x

Examples

```
geom_sd(c(1, 2, 3, 2, 1))
```

`get_unique_units_df` *Creates a dataframe with distinct parameters and units combinations*

Description

Creates a dataframe with distinct parameters and units combinations

Usage

```
get_unique_units_df(params, units)
```

Arguments

`params`	a column from a dataset with lab parameters
`units`	a column from a dataset with units associated with those parameters

Value

a dataframe with distinct units and parameters with IU replaced to U and mu replaced with u

Examples

```
df <- data.frame(
  PARAM = c(
    "ALB","ALT","AST","CR","TBIL",
    "ALB","CR","TBIL","ALT","AST"),
  UNIT = c(
    "g/L","U/L","U/L","umol/L","umol/L",
    "U/L"," mol/L"," mol/L","IU/L","IU/L")
)
unique_df <- get_unique_units_df(df$PARAM, df$UNIT)
```

---

is_asian *Takes character input and returns TRUE/FALSE if asian/other*

Description

Takes character input and returns TRUE/FALSE if asian/other

Usage

```
is_asian(x)
```

Arguments

x	input character representing race

Value

boolean representing Race == Asian

Examples

```
is_asian("ASIAN")

is_asian("BLACK")
```

---

is_black *Takes character input and returns TRUE/FALSE if black/other also checks for "African American" and "Black or African American"*

Description

Takes character input and returns TRUE/FALSE if black/other also checks for "African American" and "Black or African American"

Usage

```
is_black(x)
```

Arguments

`x`	input character representing race

Value

boolean representing Race == Black

Examples

```
is_black("WHITE")

is_black(c("AFRICAN AMERICAN", "BLACK"))
```

`is_female` *Takes character input and returns TRUE/FALSE if female/male*

Description

Takes character input and returns TRUE/FALSE if female/male

Usage

```
is_female(x)
```

Arguments

`x`	input character representing female or male

Value

boolean representing female

Examples

```
is_female("F")

is_female(c("MALE", "FEMALE"))
```

`is_hispanic_or_latino`

Takes character input and returns TRUE/FALSE if "Hispanic or Latino" or other

---

Description

Takes character input and returns TRUE/FALSE if "Hispanic or Latino" or other

Usage

```
is_hispanic_or_latino(x)
```

Arguments

`x`	input character representing ethnicity

Value

boolean representing Ethnic == "Hispanic or Latino"

Examples

```
is_hispanic_or_latino("HISPANIC OR LATINO")

is_hispanic_or_latino("NOT HISPANIC OR LATINO")

is_hispanic_or_latino("UNKNOWN")
```

`is_not_hispanic_or_latino`

Takes character input and returns TRUE/FALSE if "Not Hispanic or Latino" or other

---

Description

Takes character input and returns TRUE/FALSE if "Not Hispanic or Latino" or other

Usage

```
is_not_hispanic_or_latino(x)
```

Arguments

`x`	input character representing ethnicity

Value

boolean representing Ethnic == "Not Hispanic or Latino"

Examples

```
is_not_hispanic_or_latino("HISPANIC OR LATINO")

is_not_hispanic_or_latino("NOT HISPANIC OR LATINO")

is_not_hispanic_or_latino("UNKNOWN")
```

---

is_other — *Takes character input and returns TRUE/FALSE if other/explicit race*

Description

Takes character input and returns TRUE/FALSE if other/explicit race

Usage

```
is_other(x)
```

Arguments

x	input character representing race

Value

boolean representing Race == Other

Examples

```
is_other("OTHER")

is_other("BLACK")
```

is_white *Takes character input and returns TRUE/FALSE if white/other*

Description

Takes character input and returns TRUE/FALSE if white/other

Usage

```
is_white(x)
```

Arguments

x	input character representing race

Value

boolean representing Race == White

Examples

```
is_white("WHITE")

is_white("BLACK")
```

mdrd_egfr *Modification of Diet in Renal Disease eGFR calculation*

Description

Modification of Diet in Renal Disease eGFR calculation

Usage

```
mdrd_egfr(sexf, raceb, age, creat)
```

Arguments

sexf	a boolean representing if the patient is female.
raceb	a boolean representing if the patient is black.
age	the age of the patient in years
creat	the serum creatinine levels in mg/dL

Value

the eGFR in mL/min/1.73 m^2

Examples

```
e <- mdrd_egfr(TRUE, TRUE, 24, 1)

df <- data.frame(
   "SEXF" = c(TRUE, FALSE, TRUE, FALSE),
   "RACEB" = c(FALSE, FALSE, TRUE, FALSE),
   "AGE" = c(24, 24, 23, 24),
   "CREAT" = c(1, 1, 2, 1)
   )
df <- dplyr::mutate(df, egfr = mdrd_egfr(SEXF, RACEB, AGE, CREAT))
```

---

mosteller_bsa — *Calculates Body Surface Area based on Weight and Height using Mosteller equation*

Description

Calculates Body Surface Area based on Weight and Height using Mosteller equation

Usage

```
mosteller_bsa(weight, height)
```

Arguments

`weight`	weight of subject (kg)
`height`	height of subject (cm)

Value

the body surface area (m^2)

Examples

```
mosteller_bsa(70, 170)
```

---

racen — *Takes character input and returns standard yspec numeric value for Race*

Description

Takes character input and returns standard yspec numeric value for Race

Usage

```
racen(racec)
```

Arguments

racec	Race character

Value

the standard yspec numeric value for the inputted Race character

Examples

```
racen("WHITE") # 1

racen("BLACK") # 2

racen("ASIAN") # 3

racen("OTHER") # 4

racen("UNKNOWN") # -999
```

---

read_csv_with_hash *Reads data from csv file and prints hash of contents.*

Description

Reads data from csv file and prints hash of contents.

Usage

```
read_csv_with_hash(csv_file_path, ...)
```

Arguments

csv_file_path	path to csv file to ingest
...	additional arguments for digest or read_csv

Value

dataframe of data within file

Examples

```
## Not run:
read_csv_with_hash("data/derived/example_data.csv")

## End(Not run)
```

`read_excel_with_hash` *Reads data from xlsx/xls file and prints hash of contents.*

Description

Reads data from xlsx/xls file and prints hash of contents.

Usage

```
read_excel_with_hash(xlsx_file_path, ...)
```

Arguments

`xlsx_file_path`	an xlsx/xls file to ingest
`...`	additional arguments to digest or read_excel

Value

a dataframe(?) of data within file

Examples

```
## Not run:
read_excel_with_hash("data/source/example.xpt")

## End(Not run)
```

`read_file_with_hash` *Reads the data from a file (csv or parquet) and prints the hash*

Description

Reads the data from a file (csv or parquet) and prints the hash

Usage

```
read_file_with_hash(file_path, ...)
```

Arguments

`file_path`	path to data file
`...`	additional arguments to digest, read_csv, read_parquet, read_sas, read_pzfx, read_xpt

Value

data within the supplied file

Examples

```
## Not run:
dat <- read_file_with_hash("data/derived/PK_data.parquet")
dat2 <- read_file_with_hash("data/source/data.csv")

## End(Not run)
```

---

read_hashed_file *Reads a file if the supplied hash matches the file's hash*

Description

Reads a file if the supplied hash matches the file's hash

Usage

```
read_hashed_file(file_path, hash, ...)
```

Arguments

file_path	path to file with data you want to read
hash	hash you expect the file to have
...	additional arguments for digest or read_csv, parquet, sas

Value

data object of contents of file_path

Examples

```
## Not run:
file_path <- "data/derived/example_pk.parquet"

hash <- 0cfd6da55e6c1e198effe1e584c26d79
read_hashed_file(file_path, hash)

## End(Not run)
```

`read_parquet_with_hash` *Reads data from parquet file and prints hash of contents.*

Description

Reads data from parquet file and prints hash of contents.

Usage

```
read_parquet_with_hash(parquet_file_path, ...)
```

Arguments

`parquet_file_path`	path to parquet file to ingest
`...`	additional arguments to digest or read_parquet

Value

a tibble of data within file

Examples

```
## Not run:
read_parquet_with_hash("data/derived/example_data.parquet")

## End(Not run)
```

`read_pzfx_with_hash` *Reads in table from a prism pzfx file.*

Description

Reads in table from a prism pzfx file.

Usage

```
read_pzfx_with_hash(pzfx_file_path, ...)
```

Arguments

`pzfx_file_path`	path to pzfx file
`...`	additional arguments to digest or read_pzfx

Value

data within the table of the pzfx file

Examples

```
## Not run:
read_pzfx_with_hash("mydata.pzfx", table = "experiment1")

## End(Not run)
```

`read_sas_with_hash` *Reads data from sas file and prints hash of contents.*

Description

Reads data from sas file and prints hash of contents.

Usage

```
read_sas_with_hash(sas_file_path, ...)
```

Arguments

`sas_file_path`	path to sas file to ingest
`...`	additional arguments to digest or read_sas

Value

a dataframe(?) of data within file

Examples

```
## Not run:
read_sas_with_hash("data/source/example.sas7bdat")

## End(Not run)
```

read_xpt_with_hash *Reads data from xpt file and prints hash of contents.*

Description

Reads data from xpt file and prints hash of contents.

Usage

```
read_xpt_with_hash(xpt_file_path, ...)
```

Arguments

xpt_file_path	an xpt file to ingest
...	additional arguments to digest or read_xpt

Value

a dataframe(?) of data within file

Examples

```
## Not run:
read_xpt_with_hash("data/source/example.xpt")

## End(Not run)
```

schwartz_egfr *Calculates eGFR based on Schwartz' equation*

Description

Calculates eGFR based on Schwartz' equation

Usage

```
schwartz_egfr(height, creat)
```

Arguments

height	height of patients in cm.
creat	Serum creatinine levels in mg/dL

Value

eGFR in mL/min/1.73m^2

Examples

```
schwartz_egfr(100, 1)
```

`sexf` *Takes character input and returns standard yspec numeric value for Sex.*

Description

Also returns numeric for single character Sex characters "F" and "M"

Usage

```
sexf(sex)
```

Arguments

`sex`	Sex character

Value

the standard yspec numeric value for the inputted Sex character

Examples

```
sexf("FEMALE") # 1
sexf("female") # 1
sexf("f") # 1

sexf("MALE") # 0

sexf("NOT SPECIFIED") # 0
```

`write_csv_with_hash` *Writes data to csv_path with na_value replacing NA values.*

Description

Writes data to csv_path with na_value replacing NA values.

Usage

```
write_csv_with_hash(data, csv_path, ...)
```

Arguments

data	a data object to write to file
csv_path	the file path to save the csv
...	additional arguments to digest or write_csv

Value

Nothing, creates csv_path file and prints hash of the file

Examples

```
## Not run:
df <- data.frame(
  "a" = c(1, 2, 3, 4)
  "b" = c("A", "B", "C", "D")
)
write_csv_with_hash(df, "test/test.csv")

## End(Not run)
```

---

write_file_with_hash *Writes data to path, if directory doesn't exist it is created before file is written*

Description

Writes data to path, if directory doesn't exist it is created before file is written

Usage

```
write_file_with_hash(data, path, overwrite = FALSE, ...)
```

Arguments

data	the data object to write to file
path	the destination of the file (csv or parquet)
overwrite	boolean of whether to overwrite or not.
...	additional arguments for digest or write_file.

Value

Nothing, File is created and hash of created file is printed

Examples

```
## Not run:
df <- data.frame(
  "a" = c(1, 2, 3, 4)
  "b" = c("A", "B", "C", "D")
)
write_data_with_hash(df, "data.csv")

## End(Not run)
```

`write_parquet_with_hash` *Writes data to parquet_path and prints hash*

Description

Writes data to parquet_path and prints hash

Usage

```
write_parquet_with_hash(data, parquet_path, ...)
```

Arguments

`data`	the data object to save to parquet_path
`parquet_path`	the path to the desired parquet destination
`...`	additional arguments to digest and write_parquet

Value

Nothing. creates parquet_path file and prints hash

Examples

```
## Not run:
df <- data.frame(
  "a" = c(1, 2, 3, 4)
  "b" = c("A", "B", "C", "D")
)
write_parquet_with_hash(df, "test/test.parquet")

## End(Not run)
```

Index