Package 'SangerTools'

July 21, 2025

Type Package

Title Tools for Population Health Management Analytics

Version 1.0.2

Maintainer Asif Laldin <`laldin.asif@gmail.com`>

Description Created for population health analytics and monitoring.
The functions in this package work best when working with patient level Master Patient Index-like datasets .
Built to be used by NHS bodies and other health service providers.

License AGPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports ggplot2, dplyr, scales, janitor, readr, utils, ggthemes,
magrittr, readxl, ggtext, DBI, odbc, rlang, tibble

Suggests knitr, rmarkdown, testthat (>= 3.0.0), kableExtra

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 2.10)

Language en-gb

NeedsCompilation no

Author Asif Laldin [aut, cre],
Gary Hutson [aut] (ORCID: <`https://orcid.org/0000-0003-3534-6143`>)

Repository CRAN

Date/Publication 2022-02-20 13:10:02 UTC

Contents

`age_bandizer` *Age Band Creation: Create a new column of 5 year Age Bands from an integer column*

Description

Age Band Creation: Create a new column of 5 year Age Bands from an integer column

Usage

```
age_bandizer(df, Age_col)
```

Arguments

`df`	a tidy dataframe in standard Master Patient Index format ie SangerTools::PopHealthData
`Age_col`	a integer column within @param df NAs must be removed or imputed prior to running this function

Value

A dataframe with width ncol(df)+1, new column will be named Ageband and will be a factor with levels defined

Examples

```
library(SangerTools)
library(dplyr)
health_data <- SangerTools::PopHealthData
```

`age_bandizer_2` *Create age bands from a numerical column*

Description

An alternative age banding function that allows users greater flexibility for defining band size. This function utilises Base R standard evaluation. The function currently supports band size of 2, 5, 10 & 20. The input,column, Age_col should be numeric and must not contain NAs; if either of these conditions is violated the function will terminate.

Usage

```
age_bandizer_2(df, Age_col, Age_band_size = 5)
```

Arguments

`df`	A dataframe with a numerical column denoting Age.
`Age_col`	A numerical column within ‘df‘; passed with quotation marks.
`Age_band_size`	The size of the Age band to use. Defaults to 5; will take values 2,5,10,20.

Value

A dataframe containing a new column ‘Ageband‘ which has factor levels defined.

Examples

```
## Not run:
library(SangerTools)
df <- data.frame(Age = sample(x = 0:120, size = 100, replace = TRUE))
df_agebanded <- age_bandizer_2(
  df = df,
  Age_col = "Age",
  Age_band = 5
)
print(df_agebanded)

## End(Not run)
```

`categorical_col_chart` *Plot Counts of Categorical Variables*

Description

Create a ggplot2 column chart of categorical variables with labels, in ascending order. The plot will be customised using the provided theme `theme_sanger`, y-axis labels will have a comma for every third integer value. If the column provided to ‘grouping_var‘ has more than approximately 5 values, you may need to consider rotating x axis labels using `theme`

A comprehensive explanation of ggplot2 customisation is available here

Usage

```
categorical_col_chart(df, grouping_var)
```

Arguments

`df`	A dataframe with categorical variables
`grouping_var`	a categorical variable by which to group the count by

Value

a ggplot2 object

Examples

```
library(SangerTools)
library(dplyr)
library(ggplot2)
# Group by Age Band
health_data <- SangerTools::PopHealthData
health_data %>%
  dplyr::filter(Smoker == 1) %>%
  SangerTools::categorical_col_chart(AgeBand) +
  labs(
    title = "Smoking Population by Age Band",
    subtitle = "Majority of Smokers are Working Aged ",
    x = NULL,
    y = "Patient Number"
  )
```

`cohort_processing` *Patient Cohort Re-Identification Processing*

Description

Population Health Management commonly leads practitioners to identify a cohort that will have an intervention applied. As a rule of thumb most analysts will work with pseudonymised data sets. For targeted interventions patients require re-identification; this process is generally carried out by a third party organisation. As third party organisations work with many health care providers they have a strict set of requirements. This has been based around SW CSU's required formatting.

Usage

```
cohort_processing(
  df,
  Split_by,
  path,
  prefix = "DSCRO",
  com_code = "11M",
  date_format = "%Y%m%d",
  suffix = "_REID_V01"
)
```

Arguments

`df`	a tidy dataframe in standard Master Patient Index format ie SangerTools::PopHealthData.
`Split_by`	A column within df that will be used to split the patients and will also appear in the file name. Ideally should be a health organisation code such as GP Practice Code or Hospital Trust Code. Should only have alpha-numeric values
`path`	A file path to which the CSV files will be written
`prefix`	File name prefix, default is "DSCRO" See more here: NHS DSCRO
`com_code`	Commissioner Code, default is "11M"; Gloucestershire.
`date_format`	A date format passed internally to ‘format(Sys.Date())‘; will form part of file name to denote date of generation. You can read more about date formatting in R from R lang
`suffix`	A file name suffix, default is "_REID_V01", To be left as blank use "", without spaces.

Value

n number of CSV files written to the location specified by path argument.

`crude_rates` *Crude Prevalence Calculator*

Description

Calculate the crude prevalence of a health condition from a Master Patient Index like dataset

Usage

```
crude_rates(df, Condition, ...)
```

Arguments

`df`	a tidy dataframe in standard Master Patient Index format ie SangerTools::PopHealthData
`Condition`	A Health condition flag denoted by 1 & 0; where 1 denotes the patient being positive for the health condition
`...`	Variables used to standardise by; Must always have Ageband, additional variables are optional

Value

a tibble with Crude Prevalence Rates(Rate per 1,000) for each value included in ...

Examples

```
library(SangerTools)
library(dplyr)
health_data <- SangerTools::PopHealthData
glimpse(health_data)
# Generate crude prevalene rate stats
crude_prevalence <- SangerTools::crude_rates(health_data, Diabetes, Locality)
print(crude_prevalence)
```

`df_to_sql` *Dataframe to SQL*

Description

DataFrame to SQL; Write your DataFrame or Tibble directly to SQL from R This wrapper function allows for the easy movement of your computed results in R to a SQL Database for saving. The function uses a ODBC driver to establish a connection. You will need to select a Database that your user has write-access to. The user credentials are the same as your OS login details; as such this function will most likely only work from you work computer.

Usage

```
df_to_sql(df, driver, server, database, sql_table_name, overwrite = FALSE, ...)
```

Arguments

`df`	A ‘dataFrame‘ or ‘tibble‘ ie PopHealthData.
`driver`	A driver for database ie "SQL Server"; must be passed in quotation.
`server`	The unique name of your database server; must be passed in quotation.
`database`	The name of the database to which you will write ‘df‘; must be passed in quotation.
`sql_table_name`	The name that ‘df‘ will be referred to in SQL database; must be passed in quotation.
`overwrite`	If there is a SQL table with the same name whether it will be overwritten; defaults to FALSE.
`...`	Function forwarding for additional functionality.

Value

A message confirming that a new table has been created in a SQL ‘database‘.

Examples

```
## Not run:
library(odbc)
library(DBI)
health_data <- SangerTools::PopHealthData
df_to_sql(
  df = health_data,
  driver = "SQL SERVER",
  database = "DATABASE",
  sql_table_name = "New Table Name",
  overwrite = FALSE
)

## End(Not run)
```

`excel_clip` *Dataframe or Tibble to Clipboard*

Description

This function copies a data frame or tibble to your clipboard in a format that allows for a simple paste into excel whilst maintaining column and row structure. By default row_names has been set to FALSE.

Usage

```
excel_clip(df, row_names = FALSE, col_names = TRUE, ...)
```

Arguments

`df`	A dataframe or tibble
`row_names`	Set to FALSE for row.names not to be included
`col_names`	Set to TRUE for col.names to be included
`...`	function forwarding for additional write.table functionality

Value

a data frame copied to your clipboard

`master_patient_index` *Master Patient Index*

Description

A fabricated Master Patient Index (MPI) inspired by Gloucestershire's population to be used with functions included in SangerTools

Usage

```
master_patient_index
```

Format

A tibble with 10,000 rows and 11 variables:

PseudoNHSNumber A Pseudonymised NHS Patient Identifier

Sex The identifiable sex of the patient

Smoker Health Condition Flag: 1 denotes if the patient is a smoker

Diabetes Health Condition Flag: 1 denotes if the patient has diabetes

Dementia Health Condition Flag: 1 denotes if the patient has dementia

Obesity Health Condition Flag: 1 denotes if the patient is Obese

Age Age of the patient

IMD_Decile The decile of indices of multiple deprivation: `https://www.gov.uk/government/statistics/english-indices-of-deprivation-2019`

Ethnicity The identifiable ethnicity of the patient

Locality The region where the patient lives - sampled from Gloucestershire Clinical Commissioning Group

PrimaryCareNetwork The network of General Practioners that the patient is registerd with - sampled from Gloucestershire Clinical Commissioning Group

Source

Generated by Asif Laldin <`a.laldin@nhs.net`>, Feb-2022

Examples

```
library(dplyr)
data(master_patient_index)
# Convert diabetes data to factor'
master_patient_index %>%
 glimpse()
```

`multiple_csv_reader` *Read Multiple CSV files into R*

Description

This function reads multiple CSVs in a directory must be same structure. This function reads multiple excel files into R after which all files are aggregated into a single data frame.

There are assumptions about they underlying files:

- All files must have column names for each column (The function will fail without this; later versions will amend this)
- All files have the same number of columns
- All files have the same column names
- All files should have data starting from the same row number
- All relevant data is stored in the same sheet in each of the files

Usage

```
multiple_csv_reader(file_path, sheet = 1, rows_to_skip = 0, col_names = TRUE)
```

Arguments

`file_path`	The Directory in which the files are located
`sheet`	Sheet to read. Either a string (the name of a sheet), or an integer (the position of the sheet). Defaults to the first sheet
`rows_to_skip`	The number of rows from the top to be excluded
`col_names`	If columns are named; defaults to TRUE

Value

a data frame object full of file paths

Examples

```
library(SangerTools)
file_path <- "my_file_path_where_csvs_are_stored"
if (length(SangerTools::multiple_csv_reader(file_path)) == 0) {
 message("This won't work without changing the variable input to a local file path with CSVs in")
}
```

`multiple_excel_reader` *Read Multiple Excel files into R*

Description

This function reads multiple excel files into R after which all files are aggregated into a single data frame.

There are assumptions about they underlying files:

- All files must have column names for each column (The function will fail without this; later versions will amend this)
- All files have the same number of columns
- All files have the same column names
- All files should have data starting from the same row number
- All relevant data is stored in the same sheet in each of the files

To understand more about the underlying function that ‘multiple_excel_reader‘ wraps around Click Here

Usage

```
multiple_excel_reader(
  file_path,
  pattern = "*.xlsx",
  sheet = 1,
  rows_to_skip = 0,
  col_names = TRUE
)
```

Arguments

`file_path`	The Directory in which the files are located
`pattern`	The file extension of the files of which you are going to read. Defaults to "*.xlsx"
`sheet`	Sheet to read. Either a string (the name of a sheet), or an integer (the position of the sheet). Defaults to the first sheet
`rows_to_skip`	The number of rows from the top to be excluded
`col_names`	A boolean value to determine if column headers name are present in files. Currently only accepts TRUE

Value

a data frame object full of file paths

Examples

```
## Not run:
combined_excel_files <- multiple_excel_reader("Inputs/", 1, TRUE)

## End(Not run)
```

`PopHealthData` *PopHealthData - Population health data for testing functions*

Description

Population Health NHS data to use with the package and allows the calculation of the various metrics.

Usage

```
PopHealthData
```

Format

A small dataset with 1000 observations (rows) and 8 columns, as described hereunder:

Sex The identifiable sex of the patient

Smoker Indicates if the patient is a smoker

Diabetes Flag to indicate if patient has a type of diabetes

AgeBand The age of the patient when they came into contact with the service

IMD_Decile The decile of indices of multiple deprivation: https://www.gov.uk/government/statistics/english-indices-of-deprivation-2019

Ethnicity The identifiable ethnicity of the patient

Locality The region where the patient lives - sampled from Gloucestershire Clinical Commissioning Group

PrimaryCareNetwork The primary care network of the patient

`scale_fill_sanger` *Branded discrete colour scale*

Description

This anonymous function allows you to apply the Sanger Theme colours to your ggplot2 plot

Usage

```
scale_fill_sanger()
```

Value

A custom colour filled ggplot2 plot

Examples

```
library(SangerTools)
library(dplyr)
library(ggplot2)
# Group by Age Band
health_data <- SangerTools::PopHealthData
health_data %>%
  dplyr::filter(Smoker == 1) %>%
  SangerTools::categorical_col_chart(AgeBand) +
  labs(
    title = "Smoking Population by Age Band",
    subtitle = "Majority of Smokers are Working Aged ",
    x = NULL,
    y = "Patient Number"
  )+
  scale_fill_sanger()
```

`show_brand_palette` *Brand Colour Palette*

Description

Displays a brand colour palette for showing the hex codes associated with brand

Usage

```
show_brand_palette()
```

Value

a Base R plot object

Examples

```
library(scales)
library(SangerTools)
show_brand_palette()
```

`show_extended_palette` *Extended Brand Colour Palette*

Description

Displays extended brand colour palette for charting

Usage

```
show_extended_palette()
```

Value

a Base R plot object

Examples

```
library(scales)
library(SangerTools)
show_extended_palette()
```

`split_and_save` *Split & Save*

Description

A simpler alternative to `cohort_processing`. Will split a data frame and save as a csv

Usage

```
split_and_save(df, Split_by, path, prefix = NULL)
```

Arguments

`df`	A ‘dataFrame‘ or ‘tibble‘ ie PopHealthData.
`Split_by`	A column within df that will be used to split the patients and will also appear in the file name. Ideally should be a health organisation code such as GP Practice Code or Hospital Trust Code. Should only have alpha-numeric values
`path`	A file path to which the CSV files will be written
`prefix`	File name prefix

Value

n number of CSV files written to the location specified by path argument.

Examples

```
 ## Not run:
split_and_save(
 df = pseudo_data,
 Split_by = "Locality",
 file_path = "Inputs/",
 prefix = NULL
)

## End(Not run)
```

`standardised_rates_df` *Standardised Prevalence Rates.*

Description

Standardisation will be performed for all unique values in the column passed to ‘split_by‘. If input data frame does not contain age bands or age bands are not of class factor, it is recommended to use `age_bandizer` or `age_bandizer_2`. After the function has run, the output can be copied using `excel_clip` or written to a database using `df_to_sql`. Alternatively, if you are interested in seeing the effects of age confounding; consider joining the outputs of this function with the output from `crude_rates` using a `left_join`

Usage

```
standardised_rates_df(
  df,
  Split_by,
  Condition,
  Population_Standard,
  Granular = FALSE,
  ...
)
```

Arguments

`df`	a tidy data frame in standard Master Patient Index format ie SangerTools::PopHealthData.
`Split_by`	A column name within df for which the standardised rates will be calculated for.
`Condition`	A Health condition flag denoted by 1 & 0; where 1 denotes the patient being positive for the health condition.

`Population_Standard`	Population Standard Weight used for Standardising; default set to NULL; which denotes use of Age Structure of df.
`Granular`	Takes a boolean value. If set to TRUE will output a tibble with Standardised Rates using values provided in ‘Split_col‘ and ‘...‘By default is set to FALSE.
`...`	Variables used to standardise by; Must always have Age band for age standardisation, additional variables are optional and should be passed separated by commas.

Value

A tibble containing standardised Prevalence Rates by specified group.

Examples

```
library(SangerTools)
health_data <- SangerTools::age_bandizer(df = SangerTools::master_patient_index,
                                         Age_col=Age)
df_rates <- standardised_rates_df(
  df = health_data,
  Split_by = Locality,
  Condition = Diabetes,
  Population_Standard = NULL,
  Granular = TRUE,
  Ageband
)
print(df_rates)
```

`theme_sanger` *Customised ggplot2 Theme*

Description

A customised ggplot2 theme for the SangerTools package

Usage

```
theme_sanger()
```

Value

A customised ggplot2 plot

Examples

```
library(SangerTools)
library(ggthemes)
library(ggplot2)
library(ggtext)
categorical_col_chart(SangerTools::PopHealthData, Locality) +
  theme_sanger()+
  labs(title = "Categorical Column Chart",
  x = "Locality",
  y = "Number of Patients")+
  scale_fill_sanger()
```

`uk_pop_standard` *Data set of 2018 UK Population*

Description

Data is taken from ONS and is split into 5 year age band

Usage

```
uk_pop_standard
```

Format

A tibble with 29 rows and 2 variables:

UK_Population dbl Year price was recorded

Ageband 5 Year age band for population

Source

`https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates`

Index