Package 'epiCleanr'

August 20, 2023

Type Package

Title A Tidy Solution for Epidemiological Data

Version 0.1.0

Description epiCleanr is specifically designed to meet the requirements of epidemiologists and public health data experts. It offers a comprehensive suite of functionalities for importing, cleaning, unit testing, visualizing, and preprocessing data—tasks that are frequently encountered in epidemiological analysis. By focusing on addressing the unique challenges inherent in epidemiological datasets, epiCleanr enhances both the efficiency and reproducibility of your work.

License MIT + file LICENSE

URL https://github.com/truenomad/epiCleanr

BugReports https://github.com/truenomad/epiCleanr/issues

LazyData true

Encoding UTF-8

Imports rio,
dplyr,
tidyr,
tools,
withr,
ggplot2,
stringr,
tidyselect,
rlang

Suggests testthat, knitr

RoxygenNote 7.2.3

VignetteBuilder knitr

R topics documented:

clean_names *Clean variable names*

Description

This function transforms a variable name into a standard, cleaned format. The transformation includes converting to lower case, replacing spaces and punctuation with underscores, replacing uppercase characters following lowercase characters (camel case) with lowercase characters separated by an underscore, and removing leading or trailing underscores. The function is inspired by the `janitor::clean_names()` function from the janitor package.

Usage

```
clean_names(var)
```

Arguments

`var`	Character vector representing the variable name(s) to be cleaned.

Value

The cleaned variable name(s) as a character vector.

See Also

`janitor::clean_names()` for the function that inspired this one.

Examples

```
clean_names("My variable NAME")
clean_names(c("var1", "Another Variable", "SOME_VAR"))
```

export *Export Data to Various File Formats*

Description

This function provides a unified interface for exporting data to various file formats supported by the `rio::export()` function. The format is automatically detected from the file extension to simplify the exporting process.

Usage

```
export(data, file_path, ...)
```

Arguments

data	The dataset to be exported.
file_path	Character string specifying the path to the output file.
...	Additional arguments to be passed to the underlying write functions. These arguments are specific to the file format being exported. Please refer to the documentation of each package used for more information.

Value

No return value, called for side effects.

Examples

```
## Not run:
# Export a CSV file
export(mtcars, "path/to/your/file.csv")

# Export an Excel file
export(mtcars, "path/to/your/file.xlsx")

# Export a Stata DTA file
export(mtcars, "path/to/your/file.dta")

# Export an RDS file
export(mtcars, "path/to/your/file.rds")

# Export an RData file
export(list(mtcars = mtcars, iris = iris), "path/to/your/file.RData")

## End(Not run)
```

---

import *Import Data from Various File Formats*

Description

This function provides a unified interface for importing data from various file formats supported by the ‘rio‘ package. The format is automatically detected from the file extension to simplify the importing process.

Usage

```
import(file_path, ...)
```

Arguments

file_path	Character string specifying the path to the input file or a URL pointing to the dataset.
...	Additional arguments to be passed to the underlying read functions. These arguments are specific to the file format being imported. Please refer to the documentation of each package used for more information.

Value

A data frame or appropriate R object containing the imported data.

See Also

`foo::import()`, which this function is based on.

Examples

```
## Not run:
# Import a CSV file
data_csv <- import("path/to/your/file.csv")

# Import an Excel file
data_excel <- import("path/to/your/file.xlsx")

# Import a Stata DTA file
data_dta <- import("path/to/your/file.dta")

# Import an RDS file
data_rds <- import("path/to/your/file.rds")

# Import an RData file
data_rdata <- import("path/to/your/file.RData")

# Import a JSON file
data_json <- import("path/to/your/file.json")

# Import an SPSS file
data_spss <- import("path/to/your/file.sav")

# Import a data.table file
data_dt <- import("path/to/your/file.dt")

# Import a dataset from a website (e.g., GitHub)
data_web <- import("https://github.com/your_username/your_dataset.xlsx")

## End(Not run)
```

`missing_plot` *Plot Missing data over time*

Description

This function visualizes the proportion of missing data or reporting rate for specified variables in a dataset. It creates a tile plot using `ggplot2::ggplot()`; where the x-axis can represent any categorical time such as time (e.g., year, month), and the y-axis can represents either variables or groupings (e.g., state). The output can further be manipulated to one's needs.

Usage

```
missing_plot(data, x_var, y_var = NULL, miss_vars = NULL, use_rep_rate = FALSE)
```

Arguments

data	A data frame containing the data to be visualized. Must include columns specified in 'x_var', 'y_var', and 'vars'.
x_var	A character string specifying the time variable in 'data' (e.g., "year", "month"). Must be provided.
y_var	An optional character string specifying the grouping variable in 'data' (e.g., "state"). If provided, only one variable can be specified in 'vars'.
miss_vars	An optional character vector specifying the variables to be visualized in 'data'. If NULL, all variables except 'x_var' and 'y_var' will be used.
use_rep_rate	A logical value. If TRUE, the reporting rate is visualized; otherwise, the proportion of missing data is visualized. Defaults to FALSE

Value

A ggplot2 object representing the tile plot.

Examples

```
## Not run:
# Create a data frame with all combinations of states and years
fake_data <- expand_grid(
  state = state.name,
  month = 1:12, year = 2000:2023
) |>
  mutate(
    measles = sample(0:1000, size = n(), replace = TRUE),
    polio = sample(0:500, size = n(), replace = TRUE),
    malaria = sample(0:300, size = n(), replace = TRUE),
    cholera = sample(0:700, size = n(), replace = TRUE)
  ) |>
  # Randomly set 10% of the data in each disease column to NA
  mutate(across(
    c(measles, polio, malaria, cholera),
    ~ replace(., sample(1:n(), size = n() * 0.4), NA)
  ))

# Example function usage
missing_plot(fake_data,
  x_var = "year",
  miss_vars = "polio",
  y_var = "state",
  use_rep_rate = T
)

## End(Not run)
```

Index