Package: EFSATools (via r-universe)

May 13, 2026

Type Package

Title EFSA Ensemble of Data Collections Tools

Version 1.0.0

Maintainer Luca Belmonte <luca.belmonte@efsa.europa.eu>

Description Provides tools for dataset operations and utilities designed to preserve data history within EFSA's ad hoc data collections. It also imports packages developed by EFSA that provide additional support for data collection activities.

License EUPL-1.2

URL https://openefsa.github.io/EFSATools/

BugReports https://github.com/openefsa/EFSATools/issues

Depends R (>= 4.1.0)

Imports checkmate (>= 2.3.1), dplyr (>= 1.1.4), glue (>= 1.7.0), rlang (>= 1.1.4), stringr (>= 1.5.1), tidyr (>= 1.3.1), eppoFindeR (>= 2.0.0), distilleR (>= 1.0.0)

Suggests devtools (>= 2.4.5), tibble (>= 3.3.0), covr (>= 3.6.4), knitr (>= 1.0), rmarkdown (>= 2.0), testthat (>= 3.0.0), usethis (>= 2.2.3), roxygen2 (>= 7.2.1), cli (>= 3.6.5)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Config/testthat/edition 3

VignetteBuilder knitr

Config/Needs/website pkgdown

Roxygen list(markdown = TRUE)

Repository https://openefsa.r-universe.dev

Date/Publication 2026-04-27 10:16:56 UTC

RemoteUrl https://github.com/openefsa/efsatools

RemoteRef HEAD

RemoteSha f51222ad70260a419bdf1f887763211bdaa075f2

Contents

`dropEmpty` *Drop empty lines and columns from the specified data frame.*

Description

This function drops all the empty lines and columns from the specified data frame, i.e. all the rows and columns that contain only NAs.

Usage

```
dropEmpty(dataframe)
```

Arguments

`dataframe`	`data.frame`. The data frame from which to remove the empty lines and columns.

Value

The provided data frame without empty lines and columns and all the types transformed to string.

Examples

```
# The first row is going to be dropped.
irisTest_ <- iris
irisTest_[1, ] <- NA
irisTestDropped <- dropEmpty(irisTest_)

# The Species column is going to be dropped.
irisTest_ <- iris
irisTest_$Test <- NA
irisTestDropped <- dropEmpty(irisTest_)
```

enrich *Enrich a data frame with an EFSA catalogue.*

Description

This function takes a data frame and joins it with an EFSA catalog. The EFSA catalog must be itself a data frame.

Usage

```
enrich(dataframe, catalogue, joinBy, enrichedColumnName)
```

Arguments

dataframe	data.frame. The data frame to be enriched.
catalogue	data.frame. The data frame that contains the EFSA catalogue to be used for the enrichment. It must contain at least two columns, namely: NAME and CODE.
joinBy	character (string). The variable to be used as the join key.
enrichedColumnName	character (string). The name of the column added to the original data.

Value

The specified data frame enriched with the catalogue data.

Examples

```
dataframe_ <- iris |> dplyr::rename(CODE = Species)

catalogue_ <- iris |>
  dplyr::rename(CODE = Species) |>
  dplyr::mutate(NAME = "test") |>
  dplyr::select(CODE, NAME) |>
  unique()

enriched_ <- enrich(
  dataframe = dataframe_,
  catalogue = catalogue_,
  joinBy = "CODE",
  enrichedColumnName = "enrichedColumn")
```

`removeReplicatedColumns` *Drop and merge replicated columns from the specified data frame.*

Description

This function drops and merges all the replicated columns from the specified data frame.

Usage

```
removeReplicatedColumns(dataframe, prefix)
```

Arguments

`dataframe`	`data.frame`. The data frame from which to drop the replicated columns.
`prefix`	`character` (string). The prefix with which the name of the replicated columns starts.

Details

All the occurrences of "N/A", "NA", and empty strings (case insensitive) inside the provided data frame are replaced with NAs of type character. Then, all and only the columns starting with the specified prefix are selected and united into a single column with name ending per "_deduplicated". All empty entries in the new deduplicated column are replaced with NAs. Finally, the new column is bound with the other columns of the initial dataframe.

Value

The specified data frame with an additional deduplicated column and all the types transformed to string.

Examples

```
irisTest_ <- iris
irisTest_$Species_1 <- irisTest_$Species
irisTest_$Species_2 <- irisTest_$Species
irisTest_$Species <- NULL

deduplicatedDataframe_ <- removeReplicatedColumns(
  dataframe = irisTest_,
  prefix = "Species_")
```

SCD2 *Implement a Slowly Changing Dimension Type 2.*

Description

This function implements a Slowly Changing Dimension Type 2 to merge new and current data while maintaining historical records. The function deactivates the old records and activates new ones, ensuring a history-preserving update strategy. Only the changing records are marked as not active and replaced by new active ones.

Usage

```
SCD2(newData, currentData, key = names(newData))
```

Arguments

newData	data.frame. The data frame containing new records.
currentData	data.frame. The data frame containing existing records.
key	character (vector). The columns to be used as key.

Details

The function:

1. Separates active and inactive records from the current data.
2. Gets the old records that are still present in the new data (i.e., the ones that can remain active).
3. Gets the records present in new data but not present in still active current data (i.e., the records to activate) and activates them.
4. Gets the current active records that are not present in the new data (i.e., the records to deactivate) and deactivates them.

Value

A combined data frame with old data marked as not active and new data marked as active.

Examples

```
currentData_ <- tibble::tribble(
  ~id, ~colA, ~colB, ~colC, ~IS_ACTIVE, ~START_DATE, ~END_DATE,
  1, "a1", "b1", "c1", TRUE, Sys.time(), as.Date(NA),
  2, "a2", "b2", "c2", TRUE, Sys.time(), as.Date(NA),
  3, "a3", "b3", "c3", TRUE, Sys.time(), as.Date(NA))

newData_ <- tibble::tribble(
  ~id, ~colA, ~colB, ~colC,
  1, "a1", "b1", "c1",
  2, "a2", "b2", "c20",
```

```
  3, "a4", "b4", "c4")

mergedData <- SCD2(newData = newData_, currentData = currentData_)
```

SSCD2 *Implement a "Simple" Slowly Changing Dimension Type 2.*

Description

This function implements a Simplified version of Slowly Changing Dimension Type 2 to merge new and current data while maintaining historical records. The function deactivates all the old records and activates new ones, ensuring a history-preserving update strategy. The difference between a standard SCD2 is that this simplified version applies no checks on the data, deactivating all the old records and activating the new ones, even if some of the old records are still active.

Usage

```
SSCD2(newData, currentData)
```

Arguments

newData	data.frame. The data frame containing new records.
currentData	data.frame. The data frame containing existing records.

Value

A combined data frame with all old data marked as not active and new data marked as active.

Examples

```
currentData_ <- tibble::tribble(
  ~id, ~colA, ~colB, ~colC, ~IS_ACTIVE, ~START_DATE, ~END_DATE,
  1, "a1", "b1", "c1", TRUE, Sys.time(), as.Date(NA),
  2, "a2", "b2", "c2", TRUE, Sys.time(), as.Date(NA),
  3, "a3", "b3", "c3", TRUE, Sys.time(), as.Date(NA))

newData_ <- tibble::tribble(
  ~id, ~colA, ~colB, ~colC,
  1, "a1", "b1", "c1",
  2, "a2", "b2", "c20",
  3, "a4", "b4", "c4")

mergedData <- SSCD2(newData = newData_, currentData = currentData_)
```

Index