Package 'ProduceR'

June 18, 2026

Title Concise and Efficient Tools for Everyday Statistical Production

Version 1.3

Description A set of concise and efficient tools for statistical production. Can also be used for data management.
In statistical production, you deal with complex data and need to control your process at each step of your work.
Concise functions are very helpful, because you do not hesitate to use them.
The following functions are included in the package.
'dup' checks duplicates.
'miss' checks missing values.
'tac' computes contingency table of all columns.
'toc' compares two tables, spotting significant deviations.
'chi2_find' compares columns within a data.frame, spotting related categories of (a more complex function).

Encoding UTF-8

Imports dplyr, rlang, tibble

RoxygenNote 7.3.2

License MIT + file LICENSE

Depends R (>= 3.5)

LazyData true

NeedsCompilation no

Author Vincent Reduron [cre, aut]

Maintainer Vincent Reduron <vincent.reduron@laposte.net>

Repository CRAN

Date/Publication 2026-06-18 09:10:02 UTC

Contents

`base_eu_2024` *base_eu_2024*

Description

Description.

Usage

```
base_eu_2024
```

Format

A data frame with 232 rows and 11 variables:

pays character

region character

annee integer

Population integer

PIB numeric

PIB_habitant numeric

Taux_Croissance numeric

Taux_Chomage numeric

Inflation numeric

Dette_Publique numeric

Balance_Commerciale numeric

Source

Source

`base_eu_2025` *base_eu_2025*

Description

Description.

Usage

```
base_eu_2025
```

Format

A data frame with 295 rows and 11 variables:

pays character

region character

annee integer

Population integer

PIB numeric

PIB_habitant numeric

Taux_Croissance numeric

Taux_Chomage numeric

Inflation numeric

Dette_Publique numeric

Balance_Commerciale numeric

Source

Source

`chi2_find` *Find modalities related to a criterion*

Description

Find modalities related to a criterion

Usage

```
chi2_find(df, criterion)
```

Arguments

df	data.frame
criterion	character string: criterion that spots target rows

Value

data.frame

---

coltypes *coltypes()*

Description

Create vector of df's column types. Similar to colnames(), but with column types instead of names.

Usage

```
coltypes(df)
```

Arguments

df	data.frame

Value

vector

---

dup *Analysis of the cardinality of a key/identifier in a table*

Description

Creates multiple result tables. The term "n-plicate" is used to generalize the notion of duplicate: a n_plicate can be a duplicate, a triplicate, etc.

Usage

```
dup(
  tab,
  keyby = NULL,
  count_what = "rows",
  partition = NULL,
  view = TRUE,
  nb_xmpl = 51
)
```

Arguments

tab	Either an R dataframe or a reference to a remote table ("remote table")
keyby	(character vector) names of the column(s) considered as keys
count_what	(character vector) defines what to count by key (by *keyby*). 'rows' to count distinct rows, otherwise the name of the columns whose distinct values are to be counted
partition	(character vector) names of the columns by which to break down the analysis
view	automatic opening of generated tables
nb_xmpl	number of duplicate examples displayed in table

Value

A set of dataframes in the global environment. * nup_r_tab: table of n-plicate counts * nup_xmpl_dupl: table of examples of n-plicates * nup_xmpl_nakey: table of examples of NA keys (n-plicates with value 0) * nup_r_tab_part: table of n-plicate counts broken down by the modalities of the 'partition' columns

Examples

```
# Check if "name" is a unique key of the starwars table (yes !)
dup(dplyr::starwars, keyby = "name", view = FALSE)

# Check if "key" is a unique key of the basic table (no !)
basic <- data.frame("key"   = c("a", "b", "c", "d", NA, "a", "e", "f"),
                    "value" = c(112, 117, 317,  NA,  0,  17, 117, 112))
dup(basic, keyby = "key", view = FALSE)
```

---

get_recursion_depth *get_recursion_depth*

Description

get recursion depth of a list

Usage

```
get_recursion_depth(x, depth = 0)
```

Arguments

x	: input list
depth	: depth of x in another list (1 if x in a list. 2 if x is in a list of lists. Etc.)

Value

integer

get_tac_column *Contingency table for column ‘col_name‘ in data.frame ‘df‘*

Description

Contingency table for column ‘col_name‘ in data.frame ‘df‘

Usage

```
get_tac_column(df, col_name, values, strates)
```

Arguments

df	Input data.frame
col_name	string : name of column to which generate the contingency table
values	Vector of columns that serve as measures (amounts, counts, etc.)
strates	Vector of column names by which to stratify the contingency tables

is.Date *is.Date*

Description

Returns TRUE or FALSE depending on whether its argument is of Date type or not

Usage

```
is.Date(x)
```

Arguments

x	object

Value

TRUE/FALSE

`is.POSIXct` *is.POSIXct*

Description

Returns TRUE or FALSE depending on whether its argument is of POSIXct type or not

Usage

```
is.POSIXct(x)
```

Arguments

x	object

Value

TRUE/FALSE

`is.POSIXlt` *is.POSIXlt*

Description

Returns TRUE or FALSE depending on whether its argument is of POSIXlt type or not

Usage

```
is.POSIXlt(x)
```

Arguments

x	object

Value

TRUE/FALSE

is.POSIXt *is.POSIXt*

Description

Returns TRUE or FALSE depending on whether its argument is of POSIXxt type or not

Usage

```
is.POSIXt(x)
```

Arguments

x	object

Value

TRUE/FALSE

miss *Missing: Generate a synthetic table of missing values for all columns of a data.frame*

Description

Get a synthetic table of missing values for all columns of a data.frame

Usage

```
miss(df, values = NULL, view = FALSE)
```

Arguments

df	data.frame: Input data.frame
values	column: Variable (~weight) to measure the number of missing values (otherwise, count of rows)
view	boolean: Display a glimpse of cases with NA values

Value

data.frame

Examples

```
miss(mtcars)  # Checking NA values for all columns of mtcars (none)
```

tac *Computes a contingency table (tac) of all columns in a dataframe for control purposes*

Description

Contingency table (tac) of all columns in a dataframe for control purposes

Usage

```
tac(
  df,
  values = NULL,
  sample_rate = 0.01,
  force_identifier = "NULL",
  num_but_discrete = "NULL",
  strates = NULL
)
```

Arguments

df	Input data.frame
values	Vector of columns that serve as measures (amounts, counts, etc.)
sample_rate	Sampling rate, if df is a remote table
force_identifier	list of columns what user wants to be considred as identifiers
num_but_discrete	Vector of names of numeric columns with discrete modalities (not continuous)
strates	Vector of column names by which to stratify the contingency tables

Value

data.frame

Examples

```
tab <- tac(iris) # calculate column frequencies
```

toc *TAC-based Outlier Control (TOC)*

Description

Generalized detection of outlier values in a database, based on contingency tables (tac)

Usage

```
toc(
  df1,
  df2,
  values = NULL,
  a = 10,
  r = 0.34,
  sample_rate = 0.01,
  num_but_discrete = "NULL"
)
```

Arguments

df1	Input data.frame (to compare with df2)
df2	Input data.frame (to compare with df1)
values	Vector of columns that serve as measures (amounts, counts, etc.)
a	Allowed absolute variation
r	Allowed relative variation
sample_rate	Sampling rate, if df is a remote table
num_but_discrete	Numeric variables to be treated as discrete modal variables. If 'all', all numeric variables are treated as discrete modal variables.

Value

data.frame

toc_score *Scoring significativity of difference between two values x and y*

Description

Difference score between x and y (0 = no significant difference, >0 = presence of significant difference)

Usage

```
toc_score(x, y, a)
```

Arguments

x	(num) First value to compare
y	(num) Second value to compare
a	(num) Absolute difference threshold below which all differences are considered normal

Value

numeric

Examples

```
toc_score(15, 1500, a = 500) # 1.91: significant difference
toc_score(1432, 1501, a = 100) # 0: non-significant difference
```

%+% *short for 'paste0()'*

Description

short for 'paste0()'

Usage

```
a %+% b
```

Arguments

a	string
b	string

Value

string

Index