

Package ‘ProduceR’

October 7, 2026

Title Concise and Efficient Tools for Everyday Statistical Production

Version 1.4

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.
'sums' computes sums of all numeric columns.
'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, glue

License MIT + file LICENSE

Depends R (>= 3.5)

LazyData true

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Vincent Reduron [cre, aut]

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

Repository CRAN

Date/Publication 2026-10-07 14:10:02 UTC

Contents

base_eu_2024	2
base_eu_2025	3

chi2_find 3

coltypes 4

dup 4

get_recursion_depth 5

get_tac_column 6

is.Date 6

is.POSIXct 7

is.POSIXlt 7

is.POSIXt 8

miss 8

sums 9

tac 9

toc 10

toc_score 11

%+ % 12

Index 13

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	<i>base_eu_2025</i>
--------------	---------------------

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	<i>Find modalities related to a criterion</i>
-----------	---

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	<i>coltypes()</i>
----------	-------------------

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	<i>Analysis of the nb_appar_clefsity of a key/identifier in a table</i>
-----	---

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	<i>get_recursion_depth</i>
---------------------	----------------------------

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	<i>is.Date</i>
---------	----------------

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	<i>is.POSIXct</i>
------------	-------------------

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	<i>is.POSIXlt</i>
------------	-------------------

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	<i>Missing: Generate a synthetic table of missing values for all columns of a data.frame</i>

Description

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

Usage

miss(df, values = NULL, view = T)

Arguments

- df data.frame: Input data.frame
- values column: Variable (~weight) to measure the number of missing values (otherwise, count of rows)
- view boolean: automatic opening of generated tables

Value

data.frame

Examples

miss(base_eu_2025, view = FALSE) # Checking NA values for all columns of base_eu_2025

sums

sums: Generate a synthetic table of sums of numeric columns

Description

Get a synthetic table of sums of all numeric columns of a data.frame

Usage

```
sums(tab)
```

Arguments

tab data.frame: Input data.frame

Value

data.frame

Examples

```
sums(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",
  force_levels = "NULL",
  force_quanti = "NULL",
  strates = NULL
)
```

Arguments

<code>df</code>	Input data.frame
<code>values</code>	Vector of columns that serve as measures (amounts, counts, etc.)
<code>sample_rate</code>	Sampling rate, if df is a remote table
<code>force_identifier</code>	Vector of names of columns that user wants to be considered as identifiers
<code>force_levels</code>	Vector of names of numeric columns with discrete modalities (not continuous)
<code>force_quanti</code>	Vector of names of numeric columns that are continuous (not discrete)
<code>strates</code>	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,
  force_identifier = "NULL",
  force_levels = "NULL",
  force_quanti = "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
force_identifier	Vector of names of columns that user wants to be considred as identifiers
force_levels	Vector of names of numeric columns with discrete modalities (not continuous)
force_quantile	Vector of names of numeric columns that are continuous (not discrete) #'

Value

data.frame

Examples

```
verif <- toc(base_eu_2024, base_eu_2025[base_eu_2025$annee != 2025, ])  
# ecart significatif sur la modalite "Catalogne" (disparition dans la base 2024)
```

toc_score	<i>Scoring significativity of difference between two values x and y</i>
-----------	---

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
```

%%	<i>short for 'paste0()'</i>
----	-----------------------------

Description

short for 'paste0()'

Usage

```
a %% b
```

Arguments

a	string
b	string

Value

string

Index

* **datasets**

- base_eu_2024, [2](#)
- base_eu_2025, [3](#)
- %+%, [12](#)

- base_eu_2024, [2](#)
- base_eu_2025, [3](#)

- chi2_find, [3](#)
- coltypes, [4](#)

- dup, [4](#)

- get_recursion_depth, [5](#)
- get_tac_column, [6](#)

- is.Date, [6](#)
- is.POSIXct, [7](#)
- is.POSIXlt, [7](#)
- is.POSIXt, [8](#)

- miss, [8](#)

- sums, [9](#)

- tac, [9](#)
- toc, [10](#)
- toc_score, [11](#)