Package 'baizer'

March 10, 2023

Title Useful Functions for Data Processing

Version 0.3.0

Description
In ancient Chinese mythology, Bai Ze is a divine creature that knows the needs of everything. 'baizer' provides data processing functions frequently used by the author. Hope this package also knows what you want!

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports dplyr (>= 1.1.0), magrittr, openxlsx, purrr, rlang, rstatix, stats, stringr, tibble, tidyselect

Suggests car (>= 0.5.1), covr, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 3.5.0)

LazyData true

URL https://github.com/william-swl/baizer

BugReports https://github.com/william-swl/baizer/issues

NeedsCompilation no

Author William Song [aut, cre]

Maintainer William Song <william_swl@163.com>

Repository CRAN

Date/Publication 2023-03-10 12:30:02 UTC

R topics documented:

`adjacent_div` *expand a number vector according to the adjacent two numbers*

Description

expand a number vector according to the adjacent two numbers

Usage

```
adjacent_div(v, n_div = 10, .unique = FALSE)
```

Arguments

`v`	number vector
`n_div`	how many divisions expanded by two numbers
`.unique`	only keep unique numbers

Value

new number vector

Examples

```
adjacent_div(10^c(1:3), n_div = 10)
```

---

c2r *wrapper of tibble::column_to_rownames*

Description

wrapper of tibble::column_to_rownames

Usage

```
c2r(df, col = "")
```

Arguments

df	tibble
col	a col name

Value

data.frame

Examples

```
mini_diamond %>% c2r("id")
```

---

cmdargs *get the command line arguments*

Description

get the command line arguments

Usage

```
cmdargs(x = NULL)
```

Arguments

x	one of 'wd, R_env, script_path, script_dir, env_configs'

Value

list of all arguments, or single value of select argument

Examples

```
cmdargs()
```

`collapse_vector` *dump a named vector into character*

Description

dump a named vector into character

Usage

```
collapse_vector(named_vector, front_name = TRUE, collapse = ",")
```

Arguments

`named_vector`	a named vector
`front_name`	if TRUE, put names to former
`collapse`	collapse separator

Value

character

Examples

```
collapse_vector(c(e = 1:4), front_name = TRUE, collapse = ";")
```

`detect_dup` *detect possible duplication in a vector, ignore case, blank and special character*

Description

detect possible duplication in a vector, ignore case, blank and special character

Usage

```
detect_dup(vector, index = FALSE)
```

Arguments

vector	vector possibly with duplication
index	return duplication index

Value

duplication sub-vector

Examples

```
detect_dup(c("a", "C_", "c -", "#A"))
```

---

diff_index *the index of different character*

Description

the index of different character

Usage

```
diff_index(s1, s2, nth = NULL, ignore_case = FALSE)
```

Arguments

s1	string1
s2	string2
nth	just return nth index
ignore_case	ignore upper or lower cases

Value

list of different character indices

Examples

```
diff_index("AAAA", "ABBA")
```

extract_kv *extract key and values for a character vector*

Description

extract key and values for a character vector

Usage

```
extract_kv(v, sep = ": ", key_loc = 1, value_loc = 2)
```

Arguments

v	character vector
sep	separator between key and value
key_loc	key location
value_loc	value location

Value

a named character vector

Examples

```
extract_kv(c("x: 1", "y: 2"))
```

fancy_count *fancy count to show an extended column*

Description

fancy count to show an extended column

Usage

```
fancy_count(df, ..., ext = NULL, ext_fmt = "count", sort = TRUE, digits = 2)
```

Arguments

df	tibble
...	other arguments from dplyr::count()
ext	extended column
ext_fmt	count|ratio|clean, output format of extended column
sort	sort by frequency or not
digits	if ext_fmt=ratio, the digits of ratio

Value

count tibble

Examples

```
fancy_count(mini_diamond, cut, ext = clarity)

fancy_count(mini_diamond, cut, ext = clarity, ext_fmt = "ratio")

fancy_count(mini_diamond, cut, ext = clarity, ext_fmt = "clean")

fancy_count(mini_diamond, cut, ext = clarity, sort = FALSE)

fancy_count(mini_diamond, cut, clarity, ext = id) %>% head(5)
```

---

fetch_char *fetch character from strings*

Description

fetch character from strings

Usage

```
fetch_char(s, index_list, na.rm = FALSE, collapse = FALSE)
```

Arguments

s	strings
index_list	index of nth character, can be output of diff_index or same_index
na.rm	remove NA values from results or not
collapse	optional string used to combine the characters from a same string

Value

list of characters

Examples

```
fetch_char(rep("ABC", 3), list(1, 2, 3))
```

filterC *apply tbflt on dplyr filter*

Description

apply tbflt on dplyr filter

Usage

```
filterC(.data, tbflt = NULL, .by = NULL)
```

Arguments

.data	tibble
tbflt	tbflt object
.by	group by, same as .by argument in dplyr::filter

Value

tibble

Examples

```
c1 <- tbflt(cut == "Fair")

c2 <- tbflt(x > 8)

mini_diamond %>%
  filterC(c1) %>%
  head(5)

mini_diamond %>% filterC(c1 & c2)
```

fix_to_regex *trans fixed string into regular expression string*

Description

trans fixed string into regular expression string

Usage

```
fix_to_regex(p)
```

Arguments

p	raw fixed pattern

Value

regex pattern

Examples

```
fix_to_regex("ABC|?(*)")
```

---

float_to_percent *from float number to percent number*

Description

from float number to percent number

Usage

```
float_to_percent(x, digits = 2)
```

Arguments

x	number
digits	hold n digits after the decimal point

Value

percent character of x

Examples

```
float_to_percent(0.12)
```

fps_vector *farthest point sampling (FPS) for a vector*

Description

farthest point sampling (FPS) for a vector

Usage

```
fps_vector(v, n, method = "round")
```

Arguments

v	vector
n	sample size
method	round|floor|ceiling, the method used when trans to integer

Value

sampled vector

Examples

```
fps_vector(1:10, 4)
```

geom_mean *geometric mean*

Description

geometric mean

Usage

```
geom_mean(x, na.rm = TRUE)
```

Arguments

x	value
na.rm	remove NA or not

Value

geometric mean value

Examples

```
geom_mean(1, 9)
```

is.zero *if a number only have zeros*

Description

if a number only have zeros

Usage

```
is.zero(x)
```

Arguments

x	number

Value

all zero or not

Examples

```
is.zero(c("0.000", "0.102", NA))
```

mini_diamond *Minimal tibble dataset adjusted from diamond*

Description

Minimal tibble dataset adjusted from diamond

Usage

```
mini_diamond
```

Format

`mini_diamond`**:**
A data frame with 100 rows and 7 columns:

id unique id

cut, clarity 2 category variables

carat, price, x, y 4 continuous variables ...

Source

adjusted from `ggplot2`

move_row *move selected rows to target location*

Description

move selected rows to target location

Usage

```
move_row(df, rows, .after = FALSE, .before = FALSE)
```

Arguments

df	tibble
rows	selected rows indexes
.after	TRUE will move selected rows to the last row, or you can pass a target row index
.before	TRUE will move selected rows to the first row, or you can pass a target row index

Value

reordered tibble

Examples

```
move_row(mini_diamond, 3:5, .after = 8)
```

number_fun_wrapper *wrapper of the functions to process number string with prefix and suffix*

Description

wrapper of the functions to process number string with prefix and suffix

Usage

```
number_fun_wrapper(
  x,
  fun = ~.x,
  prefix_ext = NULL,
  suffix_ext = NULL,
  verbose = FALSE
)
```

Arguments

x	number string vector with prefix and suffix
fun	process function
prefix_ext	prefix extension
suffix_ext	suffix extension
verbose	print more details

Value

processed number with prefix and suffix

Examples

```
number_fun_wrapper(">=2.134%", function(x) round(x, 2))
```

---

ordered_slice *slice a tibble by an ordered vector*

Description

slice a tibble by an ordered vector

Usage

```
ordered_slice(df, by, ordered_vector, na.rm = FALSE, dup.rm = FALSE)
```

Arguments

df	tibble
by	slice by this column, this value must has no duplicated value
ordered_vector	ordered vector
na.rm	remove NA or unknown values from ordered vector
dup.rm	remove duplication values from ordered vector

Value

sliced tibble

Examples

```
ordered_slice(mini_diamond, id, c("id-3", "id-2"))
```

percent_to_float *from percent number to float number*

Description

from percent number to float number

Usage

```
percent_to_float(x, digits = 2)
```

Arguments

x	percent number character
digits	hold n digits after the decimal point

Value

float character of x

Examples

```
percent_to_float("12%")
```

pkglib *load packages as a batch*

Description

load packages as a batch

Usage

```
pkglib(...)
```

Arguments

...	pkgs

Value

nothing

Examples

```
baizer::pkglib(dplyr, purrr, tidyr)
```

r2c *wrapper of tibble::rownames_to_column*

Description

wrapper of tibble::rownames_to_column

Usage

```
r2c(df, col = "")
```

Arguments

df	tibble
col	a col name

Value

tibble

Examples

```
mini_diamond %>%
  c2r("id") %>%
  r2c("id")
```

round_string *from float number to fixed digits character*

Description

from float number to fixed digits character

Usage

```
round_string(x, digits = 2)
```

Arguments

x	number
digits	hold n digits after the decimal point

Value

character

Examples

```
round_string(1.1, 2)
```

---

same_index *the index of identical character*

Description

the index of identical character

Usage

```
same_index(s1, s2, nth = NULL, ignore_case = FALSE)
```

Arguments

s1	string1
s2	string2
nth	just return nth index
ignore_case	ignore upper or lower cases

Value

list of identical character indices

Examples

```
same_index("AAAA", "ABBA")
```

---

signif_round_string *signif or round string depend on the character length*

Description

signif or round string depend on the character length

Usage

```
signif_round_string(x, digits = 2, format = "short")
```

Arguments

x	number
digits	signif or round digits
format	short or long

Value

signif or round strings

Examples

```
signif_round_string(0.03851)
```

---

signif_string *from float number to fixed significant digits character*

Description

from float number to fixed significant digits character

Usage

```
signif_string(x, digits = 2)
```

Arguments

x	number
digits	hold n significant digits

Value

character

Examples

```
signif_string(1.1, 2)
```

---

split_column *split a column and return a longer tibble*

Description

split a column and return a longer tibble

Usage

```
split_column(df, name_col, value_col, sep = ",")
```

Arguments

df	tibble
name_col	repeat this as name column
value_col	expand by this value column
sep	separator in the string

Value

expanded tibble

Examples

```
fancy_count(mini_diamond, cut, ext = clarity) %>%
  split_column(name_col = cut, value_col = clarity)
```

---

stat_fc *fold change calculation which returns a extensible tibble*

Description

fold change calculation which returns a extensible tibble

Usage

```
stat_fc(df, y, x, method = "mean", .by = NULL, signif_digits = 2)
```

Arguments

df	tibble
y	value
x	sample test group
method	'mean'|'median'|'geom_mean', the summary method
.by	super-group
signif_digits	fold change signif digits

Value

fold change result tibble

Examples

```
stat_fc(mini_diamond, y = price, x = cut, .by = clarity)
```

`stat_test` *statistical test which returns a extensible tibble*

Description

statistical test which returns a extensible tibble

Usage

```
stat_test(
  df,
  y,
  x,
  paired = FALSE,
  alternative = "two.sided",
  method = "wilcoxon",
  .by = NULL,
  ...
)
```

Arguments

`df`	tibble
`y`	value
`x`	sample test group
`paired`	paired samples or not
`alternative`	one of "two.sided" (default), "greater" or "less"
`method`	test method, 'wilcoxon' as default
`.by`	super-group
`...`	other arguments to be passed to the function `wilcox.test`.

Value

test result tibble

Examples

```
stat_test(mini_diamond, y = price, x = cut, .by = clarity)
```

tbflt *create a tbflt object to save filter conditions*

Description

tbflt() can save a series of filter conditions, and support logical operating among conditions

Usage

```
tbflt(x = expression())
```

Arguments

x	any expression

Value

tbflt

Examples

```
c1 <- tbflt(cut == "Fair")

c2 <- tbflt(x > 8)

!c1

c1 | c2

c1 & c2
```

write_excel *write a tibble into an excel file*

Description

write a tibble into an excel file

Usage

```
write_excel(df, filename, sheetname = "sheet1", creator = "")
```

Arguments

df	tibble or a list of tibbles
filename	the output filename
sheetname	the name of sheets, sheet1 as default
creator	creator

Value

return status

Examples

```
# write_excel(mini_diamond, "mini_diamond.xlsx")
```

%neq% *not equal calculation operator, support NA*

Description

not equal calculation operator, support NA

Usage

```
x %neq% y
```

Arguments

x	value x
y	value y

Value

logical value, TRUE if x and y are not equal

Examples

```
1 %neq% NA
```

`%nin%` *not in calculation operator*

Description

not in calculation operator

Usage

```
left %nin% right
```

Arguments

`left`	left element
`right`	right element

Value

logical value, TRUE if left is not in right

Examples

```
0 %nin% 1:4
```

Index