Package ‘edm1.numb’

June 18, 2024

Title Provides set of functions to test and modify the characteristics of a number

Version 2.0.0.0

Description This packages provides functions to get if a character can be converted to a number, generate column name from a number or the opposite (according to the spreadsheet column name nomenclature), convert a scientific number to a normal number as a character...

License GPL (==3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports stringr,
stringi

Contents

`can_be_num` *can_be_num*

Description

Return TRUE if a variable can be converted to a number and FALSE if not (supports float)

Usage

```
can_be_num(x)
```

Arguments

`x`	is the input value

Examples

```
print(can_be_num("34.677"))

#[1] TRUE

print(can_be_num("34"))

#[1] TRUE

print(can_be_num("3rt4"))

#[1] FALSE

print(can_be_num(34))

#[1] TRUE
```

`dcr_untl` *dcr_untl*

Description

Allow to get the final value of a incremental or decremental loop.

Usage

```
dcr_untl(strt_val, cr_val, stop_val = 0)
```

Arguments

`strt_val`	is the start value
`cr_val`	is the incremental (or decremental value)
`stop_val`	is the value where the loop has to stop

Examples

```
print(dcr_untl(strt_val=50, cr_val=-5, stop_val=5))

#[1] 9

print(dcr_untl(strt_val=50, cr_val=5, stop_val=450))

#[1] 80
```

`dcr_val` *dcr_val*

Description

Allow to get the end value after an incremental (or decremental loop)

Usage

```
dcr_val(strt_val, cr_val, stop_val = 0)
```

Arguments

`strt_val`	is the start value
`cr_val`	is the incremental or decremental value
`stop_val`	is the value the loop has to stop

Examples

```
print(dcr_val(strt_val=50, cr_val=-5, stop_val=5))

#[1] 5

print(dcr_val(strt_val=47, cr_val=-5, stop_val=5))

#[1] 7

print(dcr_val(strt_val=50, cr_val=5, stop_val=450))

#[1] 450

print(dcr_val(strt_val=53, cr_val=5, stop_val=450))

#[1] 448
```

`isnt_divisible` *isnt_divisible*

Description

Takes a vector as an input and returns all the elements that are not divisible by all choosen numbers from another vector.

Usage

```
isnt_divisible(inpt_v = c(), divisible_v = c())
```

Arguments

`inpt_v`	is the input vector
`divisible_v`	is the vector containing all the numbers that will try to divide those contained in inpt_v

Examples

```
 print(isnt_divisible(inpt_v=c(1:111), divisible_v=c(2, 4, 5)))

# [1]   1   3   7   9  11  13  17  19  21  23  27  29  31  33  37  39  41  43  47
#[20]  49  51  53  57  59  61  63  67  69  71  73  77  79  81  83  87  89  91  93
#[39]  97  99 101 103 107 109 111
```

`is_divisible` *is_divisible*

Description

Takes a vector as an input and returns all the elements that are divisible by all choosen numbers from another vector.

Usage

```
is_divisible(inpt_v = c(), divisible_v = c())
```

Arguments

`inpt_v`	is the input vector
`divisible_v`	is the vector containing all the numbers that will try to divide those contained in inpt_v

Examples

```
 print(is_divisible(inpt_v=c(1:111), divisible_v=c(2, 4, 5)))

 #[1]  20  40  60  80 100
```

`letter_to_nb` *letter_to_nb*

Description

Allow to get the number of a spreadsheet based column by the letter ex: AAA = 703

Usage

```
letter_to_nb(letter)
```

Arguments

`letter`	is the letter (name of the column)

Examples

```
print(letter_to_nb("rty"))

#[1] 12713
```

`nb_to_letter` *nb_to_letter*

Description

Allow to get the letter of a spreadsheet based column by the number ex: 703 = AAA

Usage

```
nb_to_letter(x)
```

Arguments

`x`	is the number of the column

Examples

```
print(nb_to_letter(5))

[1] "e"

print(nb_to_letter(27))

[1] "aa"

print(nb_to_letter(51))

[1] "ay"

print(nb_to_letter(52))

[1] "az"

print(nb_to_letter(53))

[1] "ba"

print(nb_to_letter(675))

[1] "yy"

print(nb_to_letter(676))

[1] "yz"
```

```
print(nb_to_letter(677))

[1] "za"

print(nb_to_letter(702))

[1] "zz"

print(nb_to_letter(703))

[1] "aaa"

print(nb_to_letter(18211))

[1] "zxk"

print(nb_to_letter(18277))

[1] "zzy"

print(nb_to_letter(18278))

[1] "zzz"

print(nb_to_letter(18279))

[1] "aaaa"
```

power_to_char *power_to_char*

Description

Convert a scientific number to a string representing normally the number.

Usage

```
power_to_char(inpt_v = c())
```

Arguments

`inpt_v`	is the input vector containing scientific number, but also other elements that won't be taken in count

Examples

```
print(power_to_char(inpt_v = c(22 * 10000000, 12, 9 * 0.0000002)))

[1] "2200000000" "12"         "0.0000018"
```

Index