Package 'edm1'

June 21, 2024

Title Set of tools to generate data

Version 2.0.0.0

Description
Provides functions to generate data according to tatistical laws or special algorytms, like a column name generator or a random data generator according to different statistical laws.

License GPL (==3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports stringr,
stringi

Contents

`data_gen` *data_gen*

Description

Allo to generate in a csv all kind of data you can imagine according to what you provide

Usage

```
data_gen(
  type_ = c("number", "mixed", "string"),
  strt_l = c(0, 0, 10),
  nb_r = c(50, 10, 40),
  output = NA,
  properties = c("1-5", "1-5", "1-5"),
  type_distri = c("random", "random", "random"),
```

```
  str_source = c("a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m"
    "o", "p", "q", "r", "s", "t", "u", "w", "x", "y", "z"),
  round_l = c(0, 0, 0),
  sep_ = ","
)
```

Arguments

type_	is a vector. Its arguments designates a column, a column can be made of numbers ("number"), string ("string") or both ("mixed")
strt_l	is a vector containing for each column the row from which the data will begin to be generated
nb_r	is a vector containing for each column, the number of row full from generated data
output	is the name of the output csv file, defaults to NA so no csv will be outputed by default
properties	is linked to type_distri because it is the parameters ("min_val-max_val") for "random type", ("u-x") for the poisson distribution, ("u-d") for gaussian distribution
type_distri	is a vector which, for each column, associate a type of distribution ("random", "poisson", "gaussian"), it meas that non only the number but also the length of the string will be randomly generated according to these distribution laws
str_source	is the source (vector) from which the character creating random string are (default set to the occidental alphabet)
round_l	is a vector which, for each column containing number, associate a round value, if the type of the value is numeric
sep_	is the separator used to write data in the csv

Value

new generated data in addition to saving it in the output

Examples

```
print(data_gen())

#  X1   X2    X3
#1   4    2  <NA>
#2   2    4  <NA>
#3   5    2  <NA>
#4   2 abcd  <NA>
#5   4 abcd  <NA>
#6   2    4  <NA>
#7   2  abc  <NA>
#8   4  abc  <NA>
#9   4    3  <NA>
#10  4  abc  abcd
#11  5 <NA>   abc
#12  4 <NA>   abc
#13  1 <NA>    ab
#14  1 <NA> abcde
#15  2 <NA>   abc
```

```
#16  4 <NA>     a
#17  1 <NA>  abcd
#18  4 <NA>    ab
#19  2 <NA>  abcd
#20  3 <NA>    ab
#21  3 <NA>  abcd
#22  2 <NA>     a
#23  4 <NA>   abc
#24  1 <NA>  abcd
#25  4 <NA>   abc
#26  4 <NA>    ab
#27  2 <NA>   abc
#28  5 <NA>    ab
#29  3 <NA>   abc
#30  5 <NA>  abcd
#31  2 <NA>   abc
#32  2 <NA>   abc
#33  1 <NA>    ab
#34  5 <NA>     a
#35  4 <NA>    ab
#36  1 <NA>    ab
#37  1 <NA> abcde
#38  5 <NA>   abc
#39  4 <NA>    ab
#40  5 <NA> abcde
#41  2 <NA>    ab
#42  3 <NA>    ab
#43  2 <NA>    ab
#44  4 <NA>  abcd
#45  5 <NA>  abcd
#46  3 <NA>  abcd
#47  2 <NA>  abcd
#48  3 <NA>  abcd
#49  3 <NA>  abcd
#50  4 <NA>     a

print(data_gen(strt_l=c(0, 0, 0), nb_r=c(5, 5, 5)))

#  X1    X2   X3
#1  2     a  abc
#2  3 abcde   ab
#3  4 abcde    a
#4  1     3  abc
#5  3     a abcd
```

`infinite_char_seq` *infinite_char_seq*

Description

Allow to generate an infinite sequence of unique letters

Usage

```
infinite_char_seq(n, base_char = letters)
```

Arguments

n	is how many sequence of numbers will be generated
base_char	is the vector containing the elements from which the sequence is generated

Examples

```
print(infinite_char_seq(28))
```

```
 [1] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"  "k"  "l"  "m"  "n"  "o"
[16] "p"  "q"  "r"  "s"  "t"  "u"  "v"  "w"  "x"  "y"  "a"  "aa" "ab"
```

---

pattern_generator *pattern_generator*

Description

Allow to create patterns which have a part that is varying randomly each time.

Usage

```
pattern_generator(base_, from_, nb, hmn = 1, after = 1, sep = "")
```

Arguments

base_	is the pattern that will be kept
from_	is the vector from which the elements of the random part will be generated
nb	is the number of random pattern chosen for the varying part
hmn	is how many of varying pattern from the same base will be created
after	is set to 1 by default, it means that the varying part will be after the fixed part, set to 0 if you want the varying part to be before
sep	is the separator between all patterns in the returned value

Examples

```
print(pattern_generator(base_="oui", from_=c("er", "re", "ere"), nb=1, hmn=3))
```

```
# [1] "ouier" "ouire" "ouier"
```

```
print(pattern_generator(base_="oui", from_=c("er", "re", "ere"), nb=2, hmn=3, after=0, se
```

```
# [1] "er-re-o-u-i"  "ere-re-o-u-i" "ere-er-o-u-i"
```

selected_char *selected_char*

Description

Allow to generate a char based on a conbinaison on characters from a vector and a number

Usage

```
selected_char(n, base_char = letters)
```

Arguments

n	is how many sequence of numbers will be generated
base_char	is the vector containing the elements from which the character is generated

Examples

```
print(selected_char(1222))
```

```
[1] "zta"
```

Index