Package ‘fixtuRes’

October 13, 2022

Type Package

Title Mock Data Generator

Version 0.1.3

Description Generate mock data in R using YAML configuration.

License MIT + file LICENSE

URL https://github.com/jakubnowicki/fixtuRes

Imports stringi, stats, checkmate, rlang, purrr, R6, glue, yaml, lubridate, dplyr

Suggests testthat, lintr, knitr, rmarkdown

Encoding UTF-8

StagedInstall yes

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Author Jakub Nowicki [aut, cre]

Maintainer Jakub Nowicki <q.nowicki@gmail.com>

Repository CRAN

Date/Publication 2022-02-16 08:20:07 UTC

R topics documented:

`distribution_vector` *vector of values that follow specified distribution*

Description

vector of values that follow specified distribution

Usage

```
distribution_vector(size, distribution_type, distribution_arguments = list())
```

Arguments

`size` integer, size of the output vector

`distribution_type`
character, type of distribution. You can use direct function name, e.g. "rnorm" or a regular name (e.g. "normal", "gaussian"). All standard distributions from stats package are covered. For a list check Distributions

`distribution_arguments`
list of arguments required by the distribution function

Examples

```
distribution_vector(10, "normal", list(mean = 2, sd = 0.5))
```

`id_vector` *id vector with sequence of integers*

Description

id vector with sequence of integers

Usage

```
id_vector(size, start = 1)
```

Arguments

`size`	integer, size of the output vector
`start`	integer, value of the first element

Examples

```
id_vector(10, 2)
```

`MockDataGenerator` *MockDataGenerator*

Description

Object that stores mock data configurations and generated datasets

Methods

Public methods:

- `MockDataGenerator$new()`
- `MockDataGenerator$get_data()`
- `MockDataGenerator$get_all_data()`
- `MockDataGenerator$clone()`

Method `new()`**:** Create a new MockDataGenerator object

Usage:

```
MockDataGenerator$new(configuration)
```

Arguments:

`configuration` list or path to YAML file with datasets configurations. Check configuration for details. For a sample YAML check examples.

Returns: A new MockDataGenerator object

Method `get_data()`**:** Get a dataset (if does not exist, generate it)

Usage:

```
MockDataGenerator$get_data(data_name, size = NULL, refresh = FALSE)
```

Arguments:

`data_name` string, data set name to retrieve

`size` integer, size of dataset (if provided, will refresh dataset)

`refresh` boolean, refresh existing data?

Returns: mock dataset

Method `get_all_data()`**:** Get all datasets

Usage:

```
MockDataGenerator$get_all_data(refresh = FALSE, sizes = NULL)
```

Arguments:

`refresh` boolean, refresh existing data?

`sizes` integer, or vector of integers with data sizes

Returns: list with all datasets

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
MockDataGenerator$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`random_boolean` *Generate random boolean*

Description

Generate random boolean

Usage

```
random_boolean()
```

Value

random boolean

Examples

```
random_boolean()
```

random_data_frame *Generate a random data frame from given configuration*

Description

Generate a random data frame from given configuration

Usage

```
random_data_frame(configuration, size)
```

Arguments

`configuration` list, a configuration of columns with all arguments required by vector generator passed as sublists of sublist "columns". Column can be also generated with custom function. Pass "custom_column" as column type and function (or function name) as custom_column_generator. Column generator has to accept argument size and return a vector of this size. Third option is to pass an expression that involves existing columns. This can be a simple one, or a call of an existing function.

`size` integer, number of rows to generate.

Value

data.frame

Examples

```
conf <- list(
 columns = list(
   first_column = list(
     type = "string",
     length = 3
   ),
   second_column = list(
     type = "integer",
     max = 10
   ),
   third_column = list(
     type = "calculated",
     formula = "second_column * 2"
   )
 )
)

random_data_frame(conf, size = 10)
```

random_date *Get random date from an interval*

Description

Get random date from an interval

Usage

```
random_date(min_date, max_date, format = NULL)
```

Arguments

min_date	character or date, beginning of the time interval to sample from
max_date	character or date, ending of the time interval to sample from
format	character, check strptime for details

Examples

```
random_date("2012-12-04", "2020-10-31")
```

random_datetime *Get random datetime*

Description

Get random datetime

Usage

```
random_datetime(
  min_date,
  max_date,
  date_format = NULL,
  min_time = "00:00:00",
  max_time = "23:59:59",
  time_resolution = "seconds",
  tz = "UTC"
)
```

Arguments

`min_date`	character or date, beginning of the dates interval to sample from
`max_date`	character or date, ending of the dates interval to sample from
`date_format`	character, check `strptime` for details
`min_time`	character, beginning of the time interval to sample from
`max_time`	character, ending of the time interval to sample from
`time_resolution`	character, one of "seconds", "minutes", "hours", time resolution
`tz`	character, time zone to use

Examples

```
random_datetime("2012-12-04", "2020-10-31", min_time = "7:00:00", max_time = "17:00:00")
```

---

random_datetime_vector

Get random datetime vector

Description

Get random datetime vector

Usage

```
random_datetime_vector(
  size,
  min_date,
  max_date,
  date_format = NULL,
  date_unique = FALSE,
  min_time = "00:00:00",
  max_time = "23:59:59",
  time_resolution = "seconds",
  time_unique = FALSE,
  tz = "UTC"
)
```

Arguments

`size`	integer, vector length
`min_date`	character or date, beginning of the dates interval to sample from
`max_date`	character or date, ending of the dates interval to sample from
`date_format`	character, check `strptime` for details
`date_unique`	boolean, should the date part of the output be unique?

`min_time`	character, beginning of the time interval to sample from
`max_time`	character, ending of the time interval to sample from
`time_resolution`	character, one of "seconds", "minutes", "hours", time resolution
`time_unique`	boolean, should the time part of the output be unique?
`tz`	character, time zone to use

Examples

```
random_datetime_vector(12, "2012-12-04", "2020-10-31", min_time = "7:00:00", max_time = "17:00:00")
```

`random_date_vector` *Get random date vector from an interval*

Description

Get random date vector from an interval

Usage

```
random_date_vector(size, min_date, max_date, format = NULL, unique = FALSE)
```

Arguments

`size`	integer, vector length
`min_date`	character or date, beginning of the time interval to sample from
`max_date`	character or date, ending of the time interval to sample from
`format`	character, check `strptime` for details
`unique`	boolean, should the output be unique?

Examples

```
random_date_vector(12, "2012-12-04", "2020-10-31")
```

`random_from_set` *Choose random element from set*

Description

Choose random element from set

Usage

```
random_from_set(set)
```

Arguments

`set`	vector, set of values to choose from

Value

a single element from a given set

Examples

```
random_from_set(c("a", "b", "c"))
```

`random_integer` *Generate random integer*

Description

Generate random integer

Usage

```
random_integer(min = 0, max = 999999)
```

Arguments

`min`	integer, minimum
`max`	integer, maximum

Value

random integer

Examples

```
random_integer(min = 2, max = 10)
```

`random_numeric` *Generate random numeric*

Description

Generate random numeric

Usage

```
random_numeric(min = 0, max = 999999)
```

Arguments

`min`	numeric, minimum
`max`	numeric, maximum

Value

random numeric

Examples

```
random_numeric(min = 1.5, max = 4.45)
```

`random_string` *Generate random string*

Description

Generate random string

Usage

```
random_string(
  length = NULL,
  min_length = 1,
  max_length = 15,
  pattern = "[A-Za-z0-9]"
)
```

Arguments

`length`	integer or NULL (default), output string length. If NULL, length will be random
`min_length`	integer, minimum length if length is random. Default: 1.
`max_length`	integer, maximum length if length is random. Default: 15.
`pattern`	string, pattern for string to follow. Check `stringi-search-charclass` for details.

Value

random string

Examples

```
random_string(length = 5)
```

`random_time` *Get random time from an interval*

Description

Get random time from an interval

Usage

```
random_time(
  min_time = "00:00:00",
  max_time = "23:59:59",
  resolution = "seconds"
)
```

Arguments

`min_time`	character, beginning of the time interval to sample from
`max_time`	character, ending of the time interval to sample from
`resolution`	character, one of "seconds", "minutes", "hours", time resolution

Examples

```
random_time("12:23:00", "15:48:32")
```

`random_time_vector` *Get random time vector from an interval*

Description

Get random time vector from an interval

Usage

```
random_time_vector(
  size,
  min_time = "00:00:00",
  max_time = "23:59:59",
  resolution = "seconds",
  unique = FALSE
)
```

Arguments

`size` integer, vector length

`min_time` character, beginning of the time interval to sample from

`max_time` character, ending of the time interval to sample from

`resolution` character, one of "seconds", "minutes", "hours", time resolution

`unique` boolean, should the output be unique?

Examples

```
random_time_vector(12, "12:23:00", "15:48:32")
```

`random_vector` *Generate a random vector of desired type*

Description

Generate a random vector of desired type

Usage

```
random_vector(size, type, custom_generator = NULL, unique = FALSE, ...)
```

Arguments

`size` integer, vector length

`type` "integer", "string", "boolean", "date", "time", "datetime" or "numeric" type of vector values. If custom generator provided, should be set to "custom".

`custom_generator` function or string, custom value generator. Can be a function or a string with function name. Default: NULL

`unique` boolean, should the output contain only unique values. Default: FALSE.

`...` arguments passed to function responsible for generating values. Check `random_integer`, `random_string`, `random_boolean` and `random_numeric` for details

Value

vector of random values of chosen type

Examples

```
random_vector(5, "boolean")
random_vector(10, "numeric", min = 1.5, max = 5)
random_vector(4, "string", length = 4, pattern = "[ACGT]")
random_vector(2, "integer", max = 10)

# custom generator
custom_generator <- function() sample(c("A", "B"), 1)
random_vector(3, type = "custom", custom_generator = custom_generator)
```

`set_vector` *Generate a vector of a values from a set*

Description

Generate a vector of a values from a set

Usage

```
set_vector(size, set = NULL, set_type = NULL, set_size = NULL, ...)
```

Arguments

`size`	integer, vector length
`set`	vector a set of values to pick from; default: NULL
`set_type`	string if set is NULL generate a random set of type ("integer", "string", "boolean", "numeric"); default: NULL
`set_size`	integer, number of elements in random set; default: NULL
`...`	additional arguments for random set generator. For details check `random_vector`

Note

When using a random set, be aware, that set has to be unique, thus if arguments passed to generator do not allow this, the function can end up in an infinite loop.

Examples

```
set_vector(10, set = c("a", "b", "c"))
set_vector(size = 5, set_type = "string", set_size = 3)
```

`special_vector` *Wrapper that allows generating a special type vectors*

Description

Wrapper that allows generating a special type vectors

Usage

```
special_vector(size, type, configuration)
```

Arguments

`size`	integer, vector length
`type`	type of vector, one of: "id", "distribution"
`configuration`	list of arguments required by vector function

Examples

```
special_vector(10, "id", list(start = 3))
```

Index