Package 'dynparam'

April 4, 2019

Type Package

Title Creating Meta-Information for Parameters

Version 1.0.0

URL https://github.com/dynverse/dynparam

BugReports https://github.com/dynverse/dynparam/issues

Description Provides tools for describing parameters of algorithms in an abstract way.
Description can include an id, a description, a domain (range or list of values),
and a default value. 'dynparam' can also convert parameter sets to a 'ParamHelpers' format,
in order to be able to use 'dynparam' in conjunction with 'mlrMBO'.

License GPL-3

LazyData TRUE

RoxygenNote 6.1.1

Encoding UTF-8

Depends R (>= 3.0.0)

Imports assertthat, carrier, dplyr, dynutils (>= 1.0.2), Hmisc,
magrittr, purrr, stringr, testthat, tibble, tidyr

Suggests ParamHelpers, lhs

NeedsCompilation no

Author Robrecht Cannoodt [aut, cre] (<https://orcid.org/0000-0003-3641-729X>,
rcannood),
Wouter Saelens [aut] (<https://orcid.org/0000-0002-7114-6248>, zouter)

Maintainer Robrecht Cannoodt <rcannood@gmail.com>

Repository CRAN

Date/Publication 2019-04-04 16:10:10 UTC

R topics documented:

`character_parameter` *Define a character / string parameter*

Description

Define a character / string parameter

Usage

```
character_parameter(id, default, values, description = NULL,
  tuneable = TRUE)
```

Arguments

`id`	The name of the parameter.
`default`	The default value of the parameter.
`values`	A set of possible values.
`description`	An optional (but recommended) description of the parameter.
`tuneable`	Whether or not a parameter is tuneable.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
character_parameter(
  id = "method",
  default = "pearson",
  values = c("pearson", "spearman", "kendall"),
  description = "Which correlation coefficient to compute."
)
```

---

collapse_set *A helper function for collapsing a set*

Description

Will surround the collapsed set with brackets if it has more than one element.

Usage

```
collapse_set(..., sep = ", ", prefix = "{", postfix = "}")
```

Arguments

...	Characters to collapse
sep	Seperator between elements
prefix	A prefix
postfix	A postfix

---

distribution *Defining, serialising and printing distributions*

Description

Distributions are used to define the domain of an integer_parameter() or a numeric_parameter().

Usage

```
distribution(lower, upper, ...)

distribution_function(dist)

quantile_function(dist)

## S3 method for class 'distribution'
as.list(x, ...)

as_distribution(li)

is_distribution(x)
```

Arguments

lower	Lower limit of the distribution.
upper	Upper limit of the distribution.
...	Fields to be saved in the distribution.
dist	A distribution object.
x	An object which might be a distribution.
li	A list to be converted into a distribution.

Details

See the sections below for more information each of the functions.

List of all currently implemented distributions

- expuniform_distribution()
- normal_distribution()
- uniform_distribution()

Serialisation

- as.list(dist): Converting a distribution to a list.
- as_distribution(li): Converting a list back to a distribution.
- is_distribution(x): Checking whether something is a distribution.

Defining a distribution

In order to create a new distribution named xxx, you need to create three functions.

- A xxx() function that calls distribution(...) %>% add_class("xxx") at the end.
- quantile_function.xxx(): The quantile function for converting between a uniform distribution and the xxx distribution.
- distribution_function.xxx(): The distribution function for converting between a uniform distribution and the xxx distribution.

Check the implementations of normal_distribution(), quantile_function.normal_distribution() and distribution_function.normal_distribution() for an example on how to do define these functions. Alternatively, check the examples below.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
di <- uniform_distribution(lower = 1, upper = 10)
print(di)

li <- as.list(di)
di2 <- as_distribution(li)
print(di2)

# Defining a custom distribution, using the pbeta and qbeta functions
beta_distribution <- function(
  shape1,
  shape2,
  ncp,
  lower = -Inf,
  upper = Inf
) {
  di <- distribution(lower = lower, upper = upper, shape1, shape2, ncp)
  add_class(di, beta_distribution)
}

distribution_function.beta_distribution <- function(dist) {
  function(q) {
    stats::pbeta(q, shape1 = dist$shape1, shape2 = dist$shape2, ncp = dist$ncp)
  }
}

quantile_function.beta_distribution <- function(dist) {
  function(p) {
    stats::qbeta(p, shape1 = dist$shape1, shape2 = dist$shape2, ncp = dist$ncp)
  }
}
```

---

dynparam *Creating meta-information for parameters*

Description

Provides tools for describing parameters of algorithms in an abstract way. Description can include an id, a description, a domain (range or list of values), and a default value. 'dynparam' can also convert parameter sets to a 'ParamHelpers' format, in order to be able to use 'dynparam' in conjunction with 'mlrMBO'.

Parameter set

- Create a new `parameter_set()` by adding several parameters to it
- `as_paramhelper()`: Convert it to a ParamHelpers object
- `sip()`: Sample a parameter set

Parameters

These functions help you provide a meta description of parameters.

Implemented are the following functions:

- `character_parameter()`, `integer_parameter()`, `logical_parameter()`, `numeric_parameter()`: Creating parameters with basic R data types.
- `integer_range_parameter()`, `numeric_range_parameter()`: Create a discrete or continuous range parameter.
- `subset_parameter()`: A parameter containing a subset of a set of values.

See ?parameter for a list of helper functions converting parameters from and to other formats.

Distributions

These distributions allow to define prior distributions for numeric and integer parameters.

Implemented are the following distributions:

- `uniform_distribution()`
- `expuniform_distribution()`
- `normal_distribution()`

See ?distribution for a list of helper functions converting parameters from and to other formats.

Advanced topics

- `distribution()`: Creating a custom distribution

`expuniform_distribution` *Exponentially scaled uniform distribution.*

Description

Distributions are used for defining the domain of an `integer_parameter()` or `numeric_parameter()`.

Usage

```
expuniform_distribution(lower, upper)
```

Arguments

`lower`	Lower limit of the distribution.
`upper`	Upper limit of the distribution.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
expuniform_distribution(1, 10000)

expuniform_distribution(1e-5, 1e-2)
```

---

get_description *Get a description of the parameter*

Description

Get a description of the parameter

Usage

```
get_description(x, sep = ", ")
```

Arguments

x	The parameter
sep	A separator between different fields

---

integer_parameter *Define a integer parameter*

Description

Define a integer parameter

Usage

```
integer_parameter(id, default, distribution, description = NULL,
  tuneable = TRUE)
```

Arguments

id	The name of the parameter.
default	The default value of the parameter.
distribution	A distribution from which the parameter can be sampled.
description	An optional (but recommended) description of the parameter.
tuneable	Whether or not a parameter is tuneable.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
integer_parameter(
  id = "k",
  default = 5,
  distribution = uniform_distribution(3, 10),
  description = "The number of clusters."
)

integer_parameter(
  id = "num_iter",
  default = 100,
  distribution = expuniform_distribution(10, 10000),
  description = "The number of iterations."
)
```

---

integer_range_parameter

Define a integer range parameter

Description

Define a integer range parameter

Usage

```
integer_range_parameter(id, default, lower_distribution,
  upper_distribution, description = NULL, tuneable = TRUE)
```

Arguments

`id`	The name of the parameter.
`default`	The default value of the parameter.
`lower_distribution`	A distribution from which the lower value of the range can be sampled.
`upper_distribution`	A distribution from which the upper value fo the range can be sampled.
`description`	An optional (but recommended) description of the parameter.
`tuneable`	Whether or not a parameter is tuneable.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
integer_range_parameter(
  id = "ks",
  default = c(3L, 15L),
  lower_distribution = uniform_distribution(1L, 5L),
  upper_distribution = uniform_distribution(10L, 20L),
  description = "The numbers of clusters to be evaluated."
)
```

---

logical_parameter *Define a logical parameter*

Description

Define a logical parameter

Usage

```
logical_parameter(id, default, description = NULL, tuneable = TRUE)
```

Arguments

id	The name of the parameter.
default	The default value of the parameter.
description	An optional (but recommended) description of the parameter.
tuneable	Whether or not a parameter is tuneable.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
logical_parameter(
  id = "scale",
  default = TRUE,
  description = "Whether or not to scale the input variables"
)
```

normal_distribution *Normal distribution*

Description

Distributions are used for defining the domain of an `integer_parameter()` or `numeric_parameter()`.

Usage

```
normal_distribution(mean, sd, lower = -Inf, upper = Inf)
```

Arguments

`mean`	Mean of the distribution
`sd`	Standard deviation of the distribution.
`lower`	An optional lower limit.
`upper`	An optional upper limit.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
normal_distribution(mean = 0, sd = 1)

normal_distribution(mean = 5, sd = 1, lower = 1, upper = 10)
```

numeric_parameter *Define a numeric parameter*

Description

Define a numeric parameter

Usage

```
numeric_parameter(id, default, distribution, description = NULL,
  tuneable = TRUE)
```

Arguments

`id`	The name of the parameter.
`default`	The default value of the parameter.
`distribution`	A distribution from which the parameter can be sampled.
`description`	An optional (but recommended) description of the parameter.
`tuneable`	Whether or not a parameter is tuneable.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
numeric_parameter(
  id = "alpha",
  default = 0.5,
  distribution = uniform_distribution(0.0, 1.0),
  description = "Weighting parameter for distance function."
)

numeric_parameter(
  id = "beta",
  default = 0.001,
  distribution = expuniform_distribution(1e-4, 1e-1),
  description = "Percentage decrease in age per iteration"
)
```

---

numeric_range_parameter

Define a numeric range parameter

Description

Define a numeric range parameter

Usage

```
numeric_range_parameter(id, default, lower_distribution,
  upper_distribution, description = NULL, tuneable = TRUE)
```

Arguments

id	The name of the parameter.
default	The default value of the parameter.
lower_distribution	A distribution from which the lower value of the range can be sampled.
upper_distribution	A distribution from which the upper value fo the range can be sampled.
description	An optional (but recommended) description of the parameter.
tuneable	Whether or not a parameter is tuneable.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
numeric_range_parameter(
  id = "quantiles",
  default = c(0.1, 0.99),
  lower_distribution = uniform_distribution(0, 0.25),
  upper_distribution = uniform_distribution(0.9, 1),
  description = "The lower and upper quantile thresholds."
)
```

`parameter` *Defining, serialising and printing parameters*

Description

Multiple parameters can be combined in a parameter set. The sections below contain information on how to create, serialise and process a parameter.

Usage

```
parameter(id, default, ..., description = NULL, tuneable = TRUE)

## S3 method for class 'parameter'
as.list(x, ...)

as_parameter(li)

is_parameter(x)

as_descriptive_tibble(x)
```

Arguments

`id`	The name of the parameter.
`default`	The default value of the parameter.
`...`	Extra fields to be saved in the parameter.
`description`	An optional (but recommended) description of the parameter.
`tuneable`	Whether or not a parameter is tuneable.
`x`	An object (parameter or distribution) to be converted.
`li`	A list to be converted into a parameter.

Creating a parameter

- character_parameter(), integer_parameter(), logical_parameter(), numeric_parameter(): Creating parameters with basic R data types.
- integer_range_parameter(), numeric_range_parameter(): Create a discrete or continuous range parameter.
- subset_parameter(): A parameter containing a subset of a set of values.
- parameter(): An abstract function to be used by other parameter functions.

Serialisation

- as.list(param): Converting a parameter to a list.
- as_parameter(li): Converting a list back to a parameter.
- is_parameter(x): Checking whether something is a parameter.
- as_descriptive_tibble(param): Convert to a tibble containing meta information.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
int_param <- integer_parameter(
  id = "num_iter",
  default = 100L,
  distribution = expuniform_distribution(lower = 1L, upper = 10000L),
  description = "Number of iterations"
)

print(int_param)
li <- as.list(int_param)
print(as_parameter(li))

subset_param <- subset_parameter(
  id = "dimreds",
  default = c("pca", "mds"),
  values = c("pca", "mds", "tsne", "umap", "ica"),
  description = "Which dimensionality reduction methods to apply (can be multiple)"
)

int_range_param <- integer_range_parameter(
  id = "ks",
  default = c(3L, 15L),
  lower_distribution = uniform_distribution(1L, 5L),
  upper_distribution = uniform_distribution(10L, 20L),
  description = "The numbers of clusters to be evaluated"
)

parameter_set(
  int_param,
```

```
  subset_param,
  int_range_param
)
```

`parameter_set` *Parameter set helper functions*

Description

Parameter set helper functions

Usage

```
parameter_set(..., parameters = NULL, forbidden = NULL)

is_parameter_set(x)

## S3 method for class 'parameter_set'
as.list(x, ...)

as_parameter_set(li)

get_defaults(x)

sip(x, n = 1, as_tibble = TRUE)

as_paramhelper(x)
```

Arguments

`...`	Parameters to wrap in a parameter set.
`parameters`	A list of parameters to wrap in a parameter set.
`forbidden`	States forbidden region of parameter via a character vector, which will be turned into an expression.
`x`	An object for which to check whether it is a parameter set.
`li`	A list to be converted into a parameter set.
`n`	Number of objects to return.
`as_tibble`	Whether or not to return as a tibble.

Parameter set instatiations

- `get_defaults()`: Get all default parameters.
- `sip()`: It's like `sample()`, but for parameter sets.
- `as_paramhelper()`: Convert a parameter set to a ParamHelpers object.

Serialisation

- `as.list()`: Converting a parameter set to a list.
- `as_parameter_set()`: Converting a list back to a parameter set.
- `is_parameter_set(x)`: Checking whether something is a parameter set.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
parameters <- parameter_set(
  integer_parameter(
    id = "num_iter",
    default = 100L,
    distribution = expuniform_distribution(lower = 1L, upper = 10000L),
    description = "Number of iterations"
  ),
  subset_parameter(
    id = "dimreds",
    default = c("pca", "mds"),
    values = c("pca", "mds", "tsne", "umap", "ica"),
    description = "Which dimensionality reduction methods to apply (can be multiple)"
  ),
  integer_range_parameter(
    id = "ks",
    default = c(3L, 15L),
    lower_distribution = uniform_distribution(1L, 5L),
    upper_distribution = uniform_distribution(10L, 20L),
    description = "The numbers of clusters to be evaluated"
  )
)

get_defaults(parameters)

sip(parameters, n = 1)
```

---

range_parameter *Define a range parameter*

Description

Define a range parameter

Usage

```
range_parameter(id, default, lower_distribution, upper_distribution,
  description = NULL, tuneable = TRUE)
```

Arguments

id	The name of the parameter.
default	The default value of the parameter.
lower_distribution	A distribution from which the lower value of the range can be sampled.
upper_distribution	A distribution from which the upper value fo the range can be sampled.
description	An optional (but recommended) description of the parameter.
tuneable	Whether or not a parameter is tuneable.

---

subset_parameter *Define a subset parameter*

Description

Define a subset parameter

Usage

```
subset_parameter(id, default, values, description = NULL,
  tuneable = TRUE)
```

Arguments

id	The name of the parameter.
default	The default value of the parameter.
values	A set of possible values.
description	An optional (but recommended) description of the parameter.
tuneable	Whether or not a parameter is tuneable.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
subset_parameter(
  id = "dimreds",
  default = c("pca", "mds"),
  values = c("pca", "mds", "tsne", "umap", "ica"),
  description = "Which dimensionality reduction methods to apply (can be multiple)"
)
```

`uniform_distribution` *Uniform distribution*

Description

Distributions are used for defining the domain of an `integer_parameter()` or `numeric_parameter()`.

Usage

```
uniform_distribution(lower, upper)
```

Arguments

`lower`	Lower limit of the distribution.
`upper`	Upper limit of the distribution.

See Also

dynparam for an overview of all dynparam functionality.

Examples

```
uniform_distribution(1, 10)
```

Index