

# Package ‘RLescalation’

January 20, 2025

**Type** Package

**Title** Optimal Dose Escalation Using Deep Reinforcement Learning

**Version** 1.0.1

**Description** An implementation to compute an optimal dose escalation rule using deep reinforcement learning in phase I oncology trials (Matsuura et al. (2023) <[doi:10.1080/10543406.2023.2170402](https://doi.org/10.1080/10543406.2023.2170402)>). The dose escalation rule can directly optimize the percentages of correct selection (PCS) of the maximum tolerated dose (MTD).

**URL** <https://github.com/MatsuuraKentaro/RLescalation>

**BugReports** <https://github.com/MatsuuraKentaro/RLescalation/issues>

**VignetteBuilder** knitr

**License** MIT + file LICENSE

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.3.2

**Imports** glue, R6, nleqslv, reticulate, stats, utils

**Suggests** knitr, rmarkdown

**Collate** 'timer.R' 'train\_algo.R' 'utils.R' 'escalation\_rule.R'  
'rl\_dnn\_config.R' 'rl\_config\_set.R' 'compute\_rl\_scenarios.R'  
'learn\_escalation\_rule.R' 'setup\_python.R' 'zzz.R'  
'simulate\_one\_trial.R'

**NeedsCompilation** no

**Author** Kentaro Matsuura [aut, cre, cph]  
(<<https://orcid.org/0000-0001-5262-055X>>)

**Maintainer** Kentaro Matsuura <matsuurakentaro55@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-01-09 18:10:05 UTC

## Contents

clean_python_settings . . . . .	2
compute_rl_scenarios . . . . .	2
EscalationRule . . . . .	3
learn_escalation_rule . . . . .	5
rl_config_set . . . . .	6
rl_dnn_config . . . . .	7
setup_python . . . . .	8
simulate_one_trial . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

clean\_python\_settings *Clean the Python Virtual Environment*

---

### Description

Clean the Python Virtual Environment

### Usage

```
clean_python_settings(envname = "r-RLescalation")
```

### Arguments

envname	Python virtual environment name.
---------	----------------------------------

---

compute\_rl\_scenarios *Compute DLT Probability Scenarios for Reinforcement Learning*

---

### Description

Compute the scenarios described in Sect. 2.2 of the original paper.

### Usage

```
compute_rl_scenarios(J, target, epsilon, delta)
```

### Arguments

J	A positive integer value. The number of doses.
target	A positive numeric value. The target DLT probability.
epsilon	A positive numeric value. The acceptable range of target DLT probabilities is defined as [target - epsilon, target + epsilon].
delta	A positive numeric value. The unacceptable ranges of target DLT probabilities are defined as [0, target - delta] and [target + delta, 1].

**Value**

A named list of three elements: - prob: a list of DLT probability scenarios - MTD: a list of true MTD indices (Note that -1 means "no MTD") - weight: a vector of weights for each scenario

**Examples**

```
scenarios <- compute_rl_scenarios(J = 6, target = 0.25, epsilon = 0.04, delta = 0.1)
print(scenarios)
```

---

EscalationRule	<i>EscalationRule Class</i>
----------------	-----------------------------

---

**Description**

This class represents an escalation rule that generates a next escalation.

**Public fields**

`policy` The RLib policy that is a Python object.  
`dir` Directory path of the escalation rule (policy).  
`dirpath` Full path to the directory of the escalation rule.  
`created_at` Created time of this object.  
`info` Information when learning the escalation rule.  
`input` Inputs for learning the escalation rule.  
`log` The log of scores during the learning of the escalation rule.  
`checkpoints` The integer vector of iteration counts for checkpoints.  
`checkpoints_paths` The paths to the directories where each checkpoint is stored.

**Methods****Public methods:**

- `EscalationRule$new()`
- `EscalationRule$opt_action()`
- `EscalationRule$resume_learning()`
- `EscalationRule$set_info()`
- `EscalationRule$print()`
- `EscalationRule$clone()`

**Method** `new()`: Create a new EscalationRule object.

*Usage:*

```
EscalationRule$new(dir = "latest", base_dir = "escalation_rules")
```

*Arguments:*

`dir` A character value. A directory name or path where an escalation rule is outputted. By default, the latest escalation rule is searched in `'base_dir'`.

`base_dir` A character value. A directory path that is used as the parent directory if the `'dir'` argument is a directory name and is not used otherwise.

**Method** `opt_action()`: Compute optimal action probabilities using the obtained escalation rule for data of `N` and `DLT`.

*Usage:*

```
EscalationRule$opt_action(current_dose, data_Ns, data_DLTs)
```

*Arguments:*

`current_dose` An integer value. This is the current dose index, which is within `1:J`.

`data_Ns` A numeric vector. The cumulative number of patients assigned to each dose in your clinical trial.

`data_DLTs` A numeric vector. The cumulative number of DLTs corresponding to each dose for the `'data_Ns'` argument.

*Returns:* A character that represents the optimal action. One of the followings: `down`, `stay`, `up`, `MTD_1`, ..., `MTD_J`, `no_MTD`

**Method** `resume_learning()`: Resume learning the escalation rule. This function updates the original `EscalationRule` object.

*Usage:*

```
EscalationRule$resume_learning(iter)
```

*Arguments:*

`iter` A number of additional iterations.

*Returns:* An updated `EscalationRule` object.

**Method** `set_info()`: Set information when learning the escalation rule.

*Usage:*

```
EscalationRule$set_info(info, input, log, checkpoints)
```

*Arguments:*

`info` Information when learning the escalation rule.

`input` Inputs for learning the escalation rule.

`log` The log of scores during the learning of the escalation rule.

`checkpoints` The paths to the directories where each checkpoint is stored.

**Method** `print()`: Print function for `EscalationRule` object

*Usage:*

```
EscalationRule$print()
```

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

---

learn\_escalation\_rule *Build an Optimal Dose Escalation Rule using Reinforcement Learning*

---

### Description

Build an Optimal Dose Escalation Rule using Reinforcement Learning

### Usage

```
learn_escalation_rule(
    J,
    target,
    epsilon,
    delta,
    N_total,
    N_cohort,
    seed = NULL,
    rl_config = rl_config_set(),
    rl_scenarios = NULL,
    output_dir = format(Sys.time(), "%Y%m%d_%H%M%S"),
    output_base_dir = "escalation_rules",
    checkpoint_dir = "checkpoints"
)
```

### Arguments

J	A positive integer value. The number of doses.
target	A positive numeric value. The target DLT probability.
epsilon	A positive numeric value. The acceptable range of target DLT probabilities is defined as [target - epsilon, target + epsilon].
delta	A positive numeric value. The unacceptable ranges of target DLT probabilities are defined as [0, target - delta] and [target + delta, 1].
N_total	A positive integer value. The total number of patients.
N_cohort	A positive integer value. The number of patients for each cohort.
seed	An integer value. Random seed for reinforcement learning.
rl_config	A list. Other settings for reinforcement learning. See <a href="#">rl_config_set</a> for details.
rl_scenarios	A list. Scenarios used for reinforcement learning. Default is NULL (use scenarios in the Sect. 2.2 of the original paper). See <a href="#">compute_rl_scenarios</a> for details.
output_dir	A character value. Directory name or path to store the built escalation rule. Default is the current datetime.
output_base_dir	A character value. Parent directory path where the built escalation rule will be stored. Valid only if 'output_dir' does not contain '/'. Default is "escalation_rules".
checkpoint_dir	A character value. Parent directory path to save checkpoints. It enables you to resume learning from that point onwards. Default is "checkpoints".

**Value**

An [EscalationRule](#) object.

**Examples**

```
library(RLescalation)

# We obtain an optimal dose escalation rule by executing `learn_escalation_rule()`.
## Not run:
escalation_rule <- learn_escalation_rule(
  J = 6, target = 0.25, epsilon = 0.04, delta = 0.1,
  N_total = 36, N_cohort = 3, seed = 123,
  rl_config = rl_config_set(iter = 1000)
)
## End(Not run)
```

---

rl\_config\_set

*Configuration of Reinforcement Learning*


---

**Description**

Mainly settings for the arguments of the `training()` function. Not compatible with the new API stack introduced in Ray 2.10.0.

**Usage**

```
rl_config_set(
  iter = 1000L,
  save_start_iter = NULL,
  save_every_iter = NULL,
  cores = 4L,
  gamma = 1,
  lr = 5e-05,
  train_batch_size = 10000L,
  model = rl_dnn_config(),
  sgd_minibatch_size = 200L,
  num_sgd_iter = 20L,
  ...
)
```

**Arguments**

`iter` A positive integer value. Number of iterations.

`save_start_iter`, `save_every_iter` An integer value. Save checkpoints every `'save_every_iter'` iterations starting from `'save_start_iter'` or later.

cores	A positive integer value. Number of CPU cores used for learning.
gamma	A positive numeric value. Discount factor of the Markov decision process. Default is 1.0 (not discount).
lr	A positive numeric value. Learning rate (default 5e-5). You can set a learning schedule instead of a learning rate.
train_batch_size	A positive integer value. Training batch size. Deprecated on the new API stack.
model	A list. Arguments passed into the policy model. See <a href="#">rl_dnn_config</a> for details.
sgd_minibatch_size	A positive integer value. Total SGD batch size across all devices for SGD. Deprecated on the new API stack.
num_sgd_iter	A positive integer value. Number of SGD iterations in each outer loop.
...	Other settings for training(). See the arguments of the training() function in the source code of RLLib. <a href="https://github.com/ray-project/ray/blob/master/rllib/algorithms/algorithm_config.py">https://github.com/ray-project/ray/blob/master/rllib/algorithms/algorithm_config.py</a> <a href="https://github.com/ray-project/ray/blob/master/rllib/algorithms/ppo/ppo.py">https://github.com/ray-project/ray/blob/master/rllib/algorithms/ppo/ppo.py</a>

**Value**

A list of reinforcement learning configuration parameters

**Examples**

```
## Not run:
escalation_rule <- learn_escalation_rule(
  J = 6, target = 0.25, epsilon = 0.04, delta = 0.1,
  N_total = 36, N_cohort = 3, seed = 123,
  # We change `iter` to 200 and `cores` for reinforcement learning to 2
  rl_config = rl_config_set(iter = 200, cores = 2)
)
## End(Not run)
```

---

rl\_dnn\_config

*DNN Configuration for Reinforcement Learning*


---

**Description**

DNN (deep neural network) configuration for reinforcement learning. For detail, see Section 3.1 of the original paper.

**Usage**

```
rl_dnn_config(
  fcnet_hiddens = c(256L, 256L),
  fcnet_activation = c("relu", "tanh", "swish", "silu", "linear"),
  ...
)
```

**Arguments**

fcnet\_hiddens A positive integer vector. Numbers of units of the intermediate layers.

fcnet\_activation A character value specifying the activation function. Possible values are "ReLU" (default), "tanh", "Swish" (or "SiLU"), or "linear".

... Other configurations. See source code of RLLib. <https://github.com/ray-project/ray/blob/master/rllib/model>

**Value**

A list of DNN configuration parameters

**Examples**

```
## Not run:
escalation_rule <- learn_escalation_rule(
  J = 6, target = 0.25, epsilon = 0.04, delta = 0.1,
  N_total = 36, N_cohort = 3, seed = 123,
  rl_config = rl_config_set(
    iter = 1000,
    # We change the DNN model
    model = rl_dnn_config(fcnet_hiddens = c(512L, 512L), fcnet_activation = "tanh")
  )
)
## End(Not run)
```

---

 setup\_python

*Setting up a Python Virtual Environment*


---

**Description**

Setting up a Python virtual environment for the Ray package, which includes the RLLib library for reinforcement learning.

**Usage**

```
setup_python(envname = "r-RLescalation")
```

**Arguments**

envname Python virtual environment name.



---

simulate\_one\_trial      *Simulate One Trial Using an Obtained Optimal Dose Escalation Rule*

---

### Description

Simulate One Trial Using an Obtained Optimal Dose Escalation Rule

### Usage

```
simulate_one_trial(escalation_rule, prob_true, seed = NULL)
```

### Arguments

`escalation_rule`      An object of class [EscalationRule](#) specifying an obtained optimal dose escalation rule.

`prob_true`            A numeric vector specifying the true DLT probabilities.

`seed`                 An integer value. Random seed for data generation in this trial.

### Value

A data frame which contains the cohort ID, the assigned dose, the number of assigned patients, the number of DLTs, and the recommended action including down, stay, up, MTD\_1, ..., MTD\_J, no\_MTD, and fail to determine MTD.

### Examples

```
library(RLescalation)

## Not run:
escalation_rule <- learn_escalation_rule(
  J = 6, target = 0.25, epsilon = 0.04, delta = 0.1,
  N_total = 36, N_cohort = 3, seed = 123,
  rl_config = rl_config_set(iter = 1000)
)
## End(Not run)

prob_true <- c(0.03, 0.13, 0.17, 0.19, 0.26, 0.31)

# Simulate one trial using the obtained `escalation_rule`
## Not run:
sim_one <- simulate_one_trial(escalation_rule, prob_true, seed = 123)
## End(Not run)
```

# Index

`clean_python_settings`, [2](#)  
`compute_rl_scenarios`, [2](#), [5](#)  
`EscalationRule`, [3](#), [4](#), [6](#), [9](#)  
`learn_escalation_rule`, [5](#)  
`rl_config_set`, [5](#), [6](#)  
`rl_dnn_config`, [7](#), [7](#)  
`setup_python`, [8](#)  
`simulate_one_trial`, [9](#)