

Package ‘mlr3spatiotempcv’

August 19, 2021

Title Spatiotemporal Resampling Methods for 'mlr3'

Version 1.0.0

Date 2021-08-19

Description Extends the mlr3 ML framework with spatio-temporal resampling methods to account for the presence of spatiotemporal autocorrelation (STAC) in predictor variables. STAC may cause highly biased performance estimates in cross-validation if ignored.

License LGPL-3

URL <https://mlr3spatiotempcv.mlr-org.com/>,
<https://github.com/mlr-org/mlr3spatiotempcv>,
<https://mlr3book.mlr-org.com>

BugReports <https://github.com/mlr-org/mlr3spatiotempcv/issues>

Depends R (>= 3.5.0)

Imports checkmate, data.table, ggplot2, mlr3 (>= 0.12.0), mlr3misc (>= 0.9.2), paradox, R6, utils

Suggests bbotk, blockCV (>= 2.1.4), caret, CAST, ggsci, ggtext, knitr, lgr, mlr3filters, mlr3pipelines, mlr3tuning, patchwork, plotly, raster, rgdal, rmarkdown, rpart, sf, skmeans, sperrorest, testthat (>= 3.0.0), vdiffir (>= 1.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.1.1

Author Patrick Schratz [aut, cre] (<<https://orcid.org/0000-0003-0748-6624>>),
Marc Becker [aut] (<<https://orcid.org/0000-0002-8115-0400>>),
Jannes Muenchow [ctb] (<<https://orcid.org/0000-0001-7834-4717>>),
Michel Lang [ctb] (<<https://orcid.org/0000-0001-9754-0393>>)

Maintainer Patrick Schratz <patrick.schratz@gmail.com>

Repository CRAN

Date/Publication 2021-08-19 17:00:02 UTC

R topics documented:

mlr3spatiotempcv-package	3
as_task_classif	4
as_task_classif_st	5
as_task_regr	7
as_task_regr_st	8
autoplot.ResamplingCustomCV	10
autoplot.ResamplingCV	11
autoplot.ResamplingSpCVBlock	13
autoplot.ResamplingSpCVBuffer	16
autoplot.ResamplingSpCVCoords	17
autoplot.ResamplingSpCVDisc	19
autoplot.ResamplingSpCEnv	22
autoplot.ResamplingSpCVTiles	24
autoplot.ResamplingSptCVCluto	26
autoplot.ResamplingSptCVCstf	28
mlr_tasks_cookfarm	31
mlr_tasks_diplodia	32
mlr_tasks_ecuador	33
ResamplingRepeatedSpCVBlock	34
ResamplingRepeatedSpCVCoords	37
ResamplingRepeatedSpCVDisc	39
ResamplingRepeatedSpCEnv	41
ResamplingRepeatedSpCVTiles	43
ResamplingRepeatedSptCVCluto	46
ResamplingRepeatedSptCVCstf	49
ResamplingSpCVBlock	52
ResamplingSpCVBuffer	54
ResamplingSpCVCoords	56
ResamplingSpCVDisc	58
ResamplingSpCEnv	60
ResamplingSpCVTiles	62
ResamplingSptCVCluto	64
ResamplingSptCVCstf	66
TaskClassifST	68
TaskRegrST	71

Index

73

mlr3spatiotempcv-package

mlr3spatiotempcv: Spatiotemporal Resampling Methods for 'mlr3'

Description

Extends the mlr3 ML framework with spatio-temporal resampling methods to account for the presence of spatiotemporal autocorrelation (STAC) in predictor variables. STAC may cause highly biased performance estimates in cross-validation if ignored.

Main resources

- Book on mlr3: <https://mlr3book.mlr-org.com>
- mlr3book section about spatiotemporal data: <https://mlr3book.mlr-org.com/spatiotemporal.html>
- package vignettes: <https://mlr3spatiotempcv.mlr-org.com/dev/articles/>

Miscellaneous mlr3 content:

- Use cases and examples: <https://mlr3gallery.mlr-org.com>
- More classification and regression tasks: **mlr3data**
- Connector to **OpenML**: **mlr3oml**
- More classification and regression learners: **mlr3learners**
- Even more learners: <https://github.com/mlr-org/mlr3extralearners>
- Preprocessing and machine learning pipelines: **mlr3pipelines**
- Tuning of hyperparameters: **mlr3tuning**
- Visualizations for many **mlr3** objects: **mlr3viz**
- Survival analysis and probabilistic regression: **mlr3proba**
- Cluster analysis: **mlr3cluster**
- Feature selection filters: **mlr3filters**
- Feature selection wrappers: **mlr3fselect**
- Interface to real (out-of-memory) data bases: **mlr3db**
- Performance measures as plain functions: **mlr3measures**
- Parallelization framework: **future**
- Progress bars: **progressr**

Author(s)

Maintainer: Patrick Schratz <patrick.schratz@gmail.com> (**ORCID**)

Authors:

- Marc Becker <marcbecker@posteo.de> (**ORCID**)

Other contributors:

- Jannes Muenchow <jannes.muenchow@uni-jena.de> (**ORCID**) [contributor]
- Michel Lang <michellang@gmail.com> (**ORCID**) [contributor]

References

Schratz P, Muenchow J, Iturrutxa E, Richter J, Brenning A (2019). “Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data.” *Ecological Modelling*, **406**, 109–120. doi: [10.1016/j.ecolmodel.2019.06.002](https://doi.org/10.1016/j.ecolmodel.2019.06.002).

Valavi R, Elith J, Lahoz-Monfort JJ, Guillera-Arroita G (2018). “blockCV: an R package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models.” *bioRxiv*. doi: [10.1101/357798](https://doi.org/10.1101/357798).

Meyer H, Reudenbach C, Hengl T, Katurji M, Nauss T (2018). “Improving performance of spatio-temporal machine learning models using forward feature selection and target-oriented validation.” *Environmental Modelling & Software*, **101**, 1–9. doi: [10.1016/j.envsoft.2017.12.001](https://doi.org/10.1016/j.envsoft.2017.12.001).

Zhao Y, Karypis G (2002). “Evaluation of Hierarchical Clustering Algorithms for Document Datasets.” *11th Conference of Information and Knowledge Management (CIKM)*, 51-524. <http://glaros.dtc.umn.edu/gkhome/node/167>.

See Also

Useful links:

- <https://mlr3spatiotempcv.mlr-org.com/>
- <https://github.com/mlr-org/mlr3spatiotempcv>
- <https://mlr3book.mlr-org.com>
- Report bugs at <https://github.com/mlr-org/mlr3spatiotempcv/issues>

as_task_classif

Convert to a Classification Task

Description

Convert object to a `TaskClassif`. This is a S3 generic for `TaskClassifST`.

Usage

```
as_task_classif(x, ...)
```

```
## S3 method for class 'TaskClassifST'
as_task_classif(x)
```

Arguments

x	(any) Object to convert.
...	(any) Additional arguments.

Value[TaskClassif](#).**See Also**[mlr3::as_task_classif\(\)](#)

 as_task_classif_st *Convert to a Spatiotemporal Classification Task*

Description

Convert an object to a [TaskClassifST](#). This is a S3 generic for the following objects:

1. [TaskClassifST](#): ensure the identity
2. [data.frame\(\)](#) and [DataBackend](#): provides an alternative to the constructor of [TaskClassifST](#).
3. [sf::sf](#).

Usage

```
as_task_classif_st(x, ...)

## S3 method for class 'TaskClassifST'
as_task_classif_st(x, clone = FALSE, ...)

## S3 method for class 'data.frame'
as_task_classif_st(
  x,
  target = NULL,
  id = deparse(substitute(x)),
  positive = NULL,
  crs = NA,
  coords_as_features = FALSE,
  coordinate_names = NA,
  ...
)

## S3 method for class 'DataBackend'
as_task_classif_st(
  x,
  target = NULL,
  id = deparse(substitute(x)),
  positive = NULL,
  crs = NA,
  coords_as_features = FALSE,
  coordinate_names = c("x", "y"),
```

```

    ...
  )

  ## S3 method for class 'sf'
  as_task_classif_st(
    x,
    target = NULL,
    id = deparse(substitute(x)),
    positive = NULL,
    coords_as_features = FALSE,
    ...
  )

```

Arguments

x	(any) Object to convert.
...	(any) Additional arguments.
clone	(logical(1)) If TRUE, ensures that the returned object is not the same as the input x.
target	(character(1)) Name of the target column.
id	(character(1)) Id for the new task. Defaults to the (deparsed and substituted) name of x.
positive	(character(1)) Level of the positive class. See TaskClassif .
crs	[character(1)] Coordinate reference system. Either a PROJ string or an EPSG code.
coords_as_features	[logical(1)] Whether the coordinates should also be used as features.
coordinate_names	(character()) The variables names of the coordinates in the data.

Value

[TaskClassifST](#).

Examples

```

library("mlr3")
data("ecuador", package = "mlr3spatiotempcv")

# data.frame
as_task_classif_st(ecuador, target = "slides", positive = "TRUE",
  coords_as_features = FALSE,

```

```
crs = "+proj=utm +zone=17 +south +datum=WGS84 +units=m +no_defs",
coordinate_names = c("x", "y"))

# sf
ecuador_sf = sf::st_as_sf(ecuador, coords = c("x", "y"), crs = 32717)
as_task_classif_st(ecuador_sf, target = "slides", positive = "TRUE")

# TaskClassifST
task = tsk("ecuador")
as_task_classif_st(task)
```

as_task_regr

Convert to a Regression Task

Description

Convert object to a [TaskRegr](#). This is a S3 generic for [TaskRegrST](#).

Usage

```
as_task_regr(x, ...)
```

S3 method for class 'TaskRegrST'
as_task_regr(x)

Arguments

x	(any) Object to convert.
...	(any) Additional arguments.

Value

[TaskRegr](#).

See Also

[mlr3::as_task_regr\(\)](#)

as_task_regr_st *Convert to a Spatiotemporal Regression Task*

Description

Convert an object to a [TaskRegrST](#). This is a S3 generic for the following objects:

1. [TaskRegrST](#): ensure the identity
2. [data.frame\(\)](#) and [DataBackend](#): provides an alternative to the constructor of [TaskRegrST](#).
3. [sf::sf](#).

Usage

```
as_task_regr_st(x, ...)
```

```
## S3 method for class 'TaskRegrST'
as_task_regr_st(x, clone = FALSE, ...)
```

```
## S3 method for class 'data.frame'
as_task_regr_st(
  x,
  target = NULL,
  id = deparse(substitute(x)),
  crs = NA,
  coords_as_features = FALSE,
  coordinate_names = NA,
  ...
)
```

```
## S3 method for class 'DataBackend'
as_task_regr_st(
  x,
  target = NULL,
  id = deparse(substitute(x)),
  crs = NA,
  coords_as_features = FALSE,
  coordinate_names = c("x", "y"),
  ...
)
```

```
## S3 method for class 'sf'
as_task_regr_st(
  x,
  target = NULL,
  id = deparse(substitute(x)),
  coords_as_features = FALSE,
```



```
    ...
  )
```

Arguments

x	(any) Object to convert.
...	(any) Additional arguments.
clone	(logical(1)) If TRUE, ensures that the returned object is not the same as the input x.
target	(character(1)) Name of the target column.
id	(character(1)) Id for the new task. Defaults to the (deparsed and substituted) name of x.
crs	[character(1)] Coordinate reference system. Either a PROJ string or an EPSG code.
coords_as_features	[logical(1)] Whether the coordinates should also be used as features.
coordinate_names	(character()) The variables names of the coordinates in the data.

Value

[TaskRegrST](#).

Examples

```
library("mlr3")
data("cookfarm_sample", package = "mlr3spatiotempcv")

# data.frame
as_task_regr_st(cookfarm_sample, target = "PHIHOX",
  coords_as_features = FALSE,
  crs = 26911,
  coordinate_names = c("x", "y"))

# sf
cookfarm_sf = sf::st_as_sf(cookfarm_sample, coords = c("x", "y"), crs = 26911)
as_task_regr_st(cookfarm_sf, target = "PHIHOX")

# TaskRegrST
task = tsk("cookfarm")
as_task_regr_st(task)
```

 autoplot.ResamplingCustomCV

Visualization Functions for Non-Spatial CV Methods.

Description

Generic S3 plot() and autoplot() (ggplot2) methods.

Usage

```
## S3 method for class 'ResamplingCustomCV'
autoplot(
  object,
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  ...
)

## S3 method for class 'ResamplingCustomCV'
plot(x, ...)
```

Arguments

object	[Resampling] mlr3 spatial resampling object of class ResamplingCustomCV .
task	[TaskClassifST]/[TaskRegrST] mlr3 task object.
fold_id	[numeric] Fold IDs to plot.
plot_as_grid	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument fold_id.
train_color	[character(1)] The color to use for the training set observations.
test_color	[character(1)] The color to use for the test set observations.
...	Passed to geom_sf(). Helpful for adjusting point sizes and shapes.
x	[Resampling] mlr3 spatial resampling object of class ResamplingCustomCV .

See Also

- [mlr3book](#) chapter on on "[Spatiotemporal Visualization](#)".
- [autoplot.ResamplingSpCVBlock\(\)](#)
- [autoplot.ResamplingSpCVBuffer\(\)](#)
- [autoplot.ResamplingSpCVCoords\(\)](#)
- [autoplot.ResamplingSpCVEnv\(\)](#)
- [autoplot.ResamplingSpCVDisc\(\)](#)
- [autoplot.ResamplingSpCVTiles\(\)](#)
- [autoplot.ResamplingCV\(\)](#)
- [autoplot.ResamplingSptCVCstf\(\)](#)
- [autoplot.ResamplingSptCVCluto\(\)](#)

Examples

```
if (mlr3misc::require_namespaces(c("sf", "patchwork"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  breaks = quantile(task$data()$dem, seq(0, 1, length = 6))
  zclass = cut(task$data()$dem, breaks, include.lowest = TRUE)

  resampling = rsm("custom_cv")
  resampling$instantiate(task, f = zclass)

  autoplot(resampling, task) +
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
  autoplot(resampling, task, fold_id = 1)
  autoplot(resampling, task, fold_id = c(1, 2)) *
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
}
```

autoplot.ResamplingCV *Visualization Functions for Non-Spatial CV Methods.*

Description

Generic S3 plot() and autoplot() (ggplot2) methods.

Usage

```
## S3 method for class 'ResamplingCV'
autoplot(
  object,
  task,
  fold_id = NULL,
```

```

    plot_as_grid = TRUE,
    train_color = "#0072B5",
    test_color = "#E18727",
    ...
)

## S3 method for class 'ResamplingRepeatedCV'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = 1,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  ...
)

## S3 method for class 'ResamplingCV'
plot(x, ...)

## S3 method for class 'ResamplingRepeatedCV'
plot(x, ...)

```

Arguments

object	[Resampling] mlr3 spatial resampling object of class ResamplingCV or ResamplingRepeatedCV .
task	[TaskClassifST]/[TaskRegrST] mlr3 task object.
fold_id	[numeric] Fold IDs to plot.
plot_as_grid	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument fold_id.
train_color	[character(1)] The color to use for the training set observations.
test_color	[character(1)] The color to use for the test set observations.
...	Passed to geom_sf(). Helpful for adjusting point sizes and shapes.
repeats_id	[numeric] Repetition ID to plot.
x	[Resampling] mlr3 spatial resampling object of class ResamplingCV or ResamplingRepeatedCV .

See Also

- mlr3book chapter on on "[Spatiotemporal Visualization](#)".
- [autoplot.ResamplingSpCVBlock\(\)](#)
- [autoplot.ResamplingSpCVBuffer\(\)](#)
- [autoplot.ResamplingSpCVCoords\(\)](#)
- [autoplot.ResamplingSpCEnv\(\)](#)
- [autoplot.ResamplingSpCVDisc\(\)](#)
- [autoplot.ResamplingSpCVTiles\(\)](#)
- [autoplot.ResamplingSptCVCstf\(\)](#)
- [autoplot.ResamplingSptCVCluto\(\)](#)

Examples

```
if (mlr3misc::require_namespaces(c("sf", "patchwork", "ggtext"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmpl("cv")
  resampling$instantiate(task)

  autoplot(resampling, task) +
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
  autoplot(resampling, task, fold_id = 1)
  autoplot(resampling, task, fold_id = c(1, 2)) *
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
}
```

autoplot.ResamplingSpCVBlock

Visualization Functions for SpCV Block Methods.

Description

Generic S3 `plot()` and `autoplot()` (ggplot2) methods to visualize mlr3 spatiotemporal resampling objects.

Usage

```
## S3 method for class 'ResamplingSpCVBlock'
autoplot(
  object,
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
```

```

    test_color = "#E18727",
    show_blocks = FALSE,
    show_labels = FALSE,
    ...
)

## S3 method for class 'ResamplingRepeatedSpCVBlock'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = 1,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  show_blocks = FALSE,
  show_labels = FALSE,
  ...
)

## S3 method for class 'ResamplingSpCVBlock'
plot(x, ...)

## S3 method for class 'ResamplingRepeatedSpCVBlock'
plot(x, ...)

```

Arguments

object	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVBlock or ResamplingRepeatedSpCVBlock .
task	[TaskClassifST]/[TaskRegrST] mlr3 task object.
fold_id	[numeric] Fold IDs to plot.
plot_as_grid	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument fold_id.
train_color	[character(1)] The color to use for the training set observations.
test_color	[character(1)] The color to use for the test set observations.
show_blocks	[logical(1)] Whether to show an overlay of the spatial blocks polygons.
show_labels	[logical(1)] Whether to show an overlay of the spatial block IDs.
...	Passed to <code>geom_sf()</code> . Helpful for adjusting point sizes and shapes.

repeats_id	[numeric] Repetition ID to plot.
x	[Resampling] mlr3 spatial resampling object. One of class ResamplingSpCVBuffer , ResamplingSpCVBlock , ResamplingSpCVCoords , ResamplingSpCVEnv .

Details

By default a plot is returned; if `fold_id` is set, a gridded plot is created. If `plot_as_grid = FALSE`, a list of plot objects is returned. This can be used to align the plots individually.

When no single fold is selected, the `ggsci::scale_color_ucscgb()` palette is used to display all partitions. If you want to change the colors, call `<plot> + <color-palette>()`.

Value

`ggplot()` or list of `ggplot2` objects.

See Also

- [mlr3book](#) chapter on on "[Spatiotemporal Visualization](#)".
- [autoplot.ResamplingSpCVBuffer\(\)](#)
- [autoplot.ResamplingSpCVCoords\(\)](#)
- [autoplot.ResamplingSpCVEnv\(\)](#)
- [autoplot.ResamplingSpCVDisc\(\)](#)
- [autoplot.ResamplingSpCVTiles\(\)](#)
- [autoplot.ResamplingCV\(\)](#)
- [autoplot.ResamplingSptCVCstf\(\)](#)
- [autoplot.ResamplingSptCVCluto\(\)](#)

Examples

```
if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmpl("spcv_block", range = 1000L)
  resampling$instantiate(task)

  ## list of ggplot2 resamplings
  plot_list = autoplot(resampling, task,
    crs = 4326,
    fold_id = c(1, 2), plot_as_grid = FALSE)

  ## Visualize all partitions
  autoplot(resampling, task) +
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))

  ## Visualize the train/test split of a single fold
```

```

autoplot(resampling, task, fold_id = 1) +
  ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))

## Visualize train/test splits of multiple folds
autoplot(resampling, task,
  fold_id = c(1, 2),
  show_blocks = TRUE) *
  ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
}

```

autoplot.ResamplingSpCVBuffer

Visualization Functions for SpCV Buffer Methods.

Description

Generic `S3 plot()` and `autoplot()` (`ggplot2`) methods to visualize `mlr3` spatiotemporal resampling objects.

Usage

```

## S3 method for class 'ResamplingSpCVBuffer'
autoplot(
  object,
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  ...
)

## S3 method for class 'ResamplingSpCVBuffer'
plot(x, ...)

```

Arguments

<code>object</code>	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVBuffer .
<code>task</code>	[TaskClassifST]/[TaskRegrST] mlr3 task object.
<code>fold_id</code>	[numeric] Fold IDs to plot.
<code>plot_as_grid</code>	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument <code>fold_id</code> .

train_color	[character(1)] The color to use for the training set observations.
test_color	[character(1)] The color to use for the test set observations.
...	Passed to geom_sf(). Helpful for adjusting point sizes and shapes.
x	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVBuffer .

See Also

- [mlr3book](#) chapter on on "[Spatiotemporal Visualization](#)".
- [autoplot.ResamplingSpCVBlock\(\)](#)
- [autoplot.ResamplingSpCVCoords\(\)](#)
- [autoplot.ResamplingSpCEnv\(\)](#)
- [autoplot.ResamplingCV\(\)](#)
- [autoplot.ResamplingSptCVCstf\(\)](#)
- [autoplot.ResamplingSptCVCluto\(\)](#)

Examples

```
if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmpl("spcv_buffer", theRange = 1000)
  resampling$instantiate(task)

  ## single fold
  autoplot(resampling, task, fold_id = 1) +
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))

  ## multiple folds
  autoplot(resampling, task, fold_id = c(1, 2)) *
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
}
```

autoplot.ResamplingSpCVCoords

Visualization Functions for SpCV Coords Methods.

Description

Generic S3 plot() and autoplot() (ggplot2) methods.

Usage

```
## S3 method for class 'ResamplingSpCVCoords'
autoplot(
  object,
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  ...
)

## S3 method for class 'ResamplingRepeatedSpCVCoords'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = 1,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  ...
)

## S3 method for class 'ResamplingSpCVCoords'
plot(x, ...)

## S3 method for class 'ResamplingRepeatedSpCVCoords'
plot(x, ...)
```

Arguments

object	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVCoords or ResamplingRepeatedSpCVCoords .
task	[TaskClassifST]/[TaskRegrST] mlr3 task object.
fold_id	[numeric] Fold IDs to plot.
plot_as_grid	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument fold_id.
train_color	[character(1)] The color to use for the training set observations.
test_color	[character(1)] The color to use for the test set observations.

...	Passed to geom_sf(). Helpful for adjusting point sizes and shapes.
repeats_id	[numeric] Repetition ID to plot.
x	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVCoords or ResamplingRepeatedSpCVCoords .

See Also

- mlr3book chapter on on "[Spatiotemporal Visualization](#)".
- [autoplot.ResamplingSpCVBlock\(\)](#)
- [autoplot.ResamplingSpCVBuffer\(\)](#)
- [autoplot.ResamplingSpCEnv\(\)](#)
- [autoplot.ResamplingSpCVDisc\(\)](#)
- [autoplot.ResamplingSpCVTiles\(\)](#)
- [autoplot.ResamplingCV\(\)](#)
- [autoplot.ResamplingSptCVCstf\(\)](#)
- [autoplot.ResamplingSptCVCluto\(\)](#)

Examples

```
if (mlr3misc::require_namespaces(c("sf"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmpl("spcv_coords")
  resampling$instantiate(task)

  autoplot(resampling, task) +
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
  autoplot(resampling, task, fold_id = 1)
  autoplot(resampling, task, fold_id = c(1, 2)) *
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
}
```

autoplot.ResamplingSpCVDisc

Visualization Functions for SpCV Disc Method.

Description

Generic S3 plot() and autoplot() (ggplot2) methods to visualize mlr3 spatiotemporal resampling objects.

Usage

```
## S3 method for class 'ResamplingSpCVDisc'
autoplot(
  object,
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  repeats_id = NULL,
  show_omitted = FALSE,
  ...
)

## S3 method for class 'ResamplingRepeatedSpCVDisc'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = 1,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  show_omitted = FALSE,
  ...
)

## S3 method for class 'ResamplingSpCVDisc'
plot(x, ...)

## S3 method for class 'ResamplingRepeatedSpCVDisc'
plot(x, ...)
```

Arguments

object	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVBlock or ResamplingRepeatedSpCVBlock .
task	[TaskClassifST]/[TaskRegrST] mlr3 task object.
fold_id	[numeric] Fold IDs to plot.
plot_as_grid	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument fold_id.
train_color	[character(1)] The color to use for the training set observations.

test_color	[character(1)] The color to use for the test set observations.
repeats_id	[numeric] Repetition ID to plot.
show_omitted	[logical] Whether to show points not used in train or test set for the current fold.
...	Passed to geom_sf(). Helpful for adjusting point sizes and shapes.
x	[Resampling] mlr3 spatial resampling object. One of class ResamplingSpCVBuffer , ResamplingSpCVBlock , ResamplingSpCVCoords , ResamplingSpCEnv .

Details

This method requires to set argument `fold_id` and no plot containing all partitions can be created. This is because the method does not make use of all observations but only a subset of them (many observations are left out). Hence, train and test sets of one fold are not re-used in other folds as in other methods and plotting these without a train/test indicator would not make sense.

2D vs 3D plotting

This method has both a 2D and a 3D plotting method. The 2D method returns a **ggplot** with x and y axes representing the spatial coordinates. The 3D method uses **plotly** to create an interactive 3D plot. Set `plot3D = TRUE` to use the 3D method.

Note that spatiotemporal datasets usually suffer from overplotting in 2D mode.

See Also

- mlr3book chapter on on "[Spatiotemporal Visualization](#)".
- Vignette [Spatiotemporal Visualization](#).
- [autoplot.ResamplingSpCVBlock\(\)](#)
- [autoplot.ResamplingSpCVBuffer\(\)](#)
- [autoplot.ResamplingSpCVCoords\(\)](#)
- [autoplot.ResamplingSpCVTiles\(\)](#)
- [autoplot.ResamplingSpCEnv\(\)](#)
- [autoplot.ResamplingCV\(\)](#)
- [autoplot.ResamplingSptCVCluto\(\)](#)

Examples

```
if (mlr3misc::require_namespaces("sf", quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmpl("spcv_disc",
    folds = 5, radius = 200L, buffer = 200L)
```

```

resampling$instantiate(task)

autoplot(resampling, task,
  fold_id = 1,
  show_omitted = TRUE, size = 0.7) *
  ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
}

```

autoplot.ResamplingSpCEnv

Visualization Functions for SpCV Env Methods.

Description

Generic S3 plot() and autoplot() (ggplot2) methods.

Usage

```

## S3 method for class 'ResamplingSpCEnv'
autoplot(
  object,
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  ...
)

```

```

## S3 method for class 'ResamplingRepeatedSpCEnv'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = 1,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  ...
)

```

```

## S3 method for class 'ResamplingSpCEnv'
plot(x, ...)

```

```

## S3 method for class 'ResamplingRepeatedSpCEnv'
plot(x, ...)

```

Arguments

object	[Resampling] mlr3 spatial resampling object of class ResamplingSpCEnv or ResamplingRepeatedSpCEnv .
task	[TaskClassifST]/[TaskRegrST] mlr3 task object.
fold_id	[numeric] Fold IDs to plot.
plot_as_grid	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument fold_id.
train_color	[character(1)] The color to use for the training set observations.
test_color	[character(1)] The color to use for the test set observations.
...	Passed to geom_sf(). Helpful for adjusting point sizes and shapes.
repeats_id	[numeric] Repetition ID to plot.
x	[Resampling] mlr3 spatial resampling object of class ResamplingSpCEnv or ResamplingRepeatedSpCEnv .

See Also

- mlr3book chapter on on "[Spatiotemporal Visualization](#)".
- [autoplot.ResamplingSpCVBlock\(\)](#)
- [autoplot.ResamplingSpCVBuffer\(\)](#)
- [autoplot.ResamplingSpCVCoords\(\)](#)
- [autoplot.ResamplingSpCVDisc\(\)](#)
- [autoplot.ResamplingSpCVTiles\(\)](#)
- [autoplot.ResamplingCV\(\)](#)
- [autoplot.ResamplingSptCVCstf\(\)](#)
- [autoplot.ResamplingSptCVCluto\(\)](#)

Examples

```
if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmpl("spcv_env", folds = 4, features = "dem")
  resampling$instantiate(task)

  autoplot(resampling, task) +
```

```

    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
  autoplot(resampling, task, fold_id = 1)
  autoplot(resampling, task, fold_id = c(1, 2)) *
    ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
}

```

autoplot.ResamplingSpCVTiles

Visualization Functions for SpCV Tiles Method.

Description

Generic S3 plot() and autoplot() (ggplot2) methods to visualize mlr3 spatiotemporal resampling objects.

Usage

```

## S3 method for class 'ResamplingSpCVTiles'
autoplot(
  object,
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  repeats_id = NULL,
  show_omitted = FALSE,
  ...
)

## S3 method for class 'ResamplingRepeatedSpCVTiles'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = 1,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  show_omitted = FALSE,
  ...
)

## S3 method for class 'ResamplingSpCVTiles'
plot(x, ...)

## S3 method for class 'ResamplingRepeatedSpCVTiles'
plot(x, ...)

```


Arguments

object	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVBlock or ResamplingRepeatedSpCVBlock .
task	[TaskClassifST]/[TaskRegrST] mlr3 task object.
fold_id	[numeric] Fold IDs to plot.
plot_as_grid	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument fold_id.
train_color	[character(1)] The color to use for the training set observations.
test_color	[character(1)] The color to use for the test set observations.
repeats_id	[numeric] Repetition ID to plot.
show_omitted	[logical] Whether to show points not used in train or test set for the current fold.
...	Passed to <code>geom_sf()</code> . Helpful for adjusting point sizes and shapes.
x	[Resampling] mlr3 spatial resampling object. One of class ResamplingSpCVBuffer , ResamplingSpCVBlock , ResamplingSpCVCoords , ResamplingSpCVEnv .

Details

Specific combinations of arguments of "spcv_tiles" remove some observations, hence show_omitted has an effect in some cases.

See Also

- mlr3book chapter on on "[Spatiotemporal Visualization](#)".
- Vignette [Spatiotemporal Visualization](#).
- [autoplot.ResamplingSpCVBlock\(\)](#)
- [autoplot.ResamplingSpCVBuffer\(\)](#)
- [autoplot.ResamplingSpCVCoords\(\)](#)
- [autoplot.ResamplingSpCVDisc\(\)](#)
- [autoplot.ResamplingSpCVEnv\(\)](#)
- [autoplot.ResamplingCV\(\)](#)
- [autoplot.ResamplingSptCVCluto\(\)](#)

Examples

```

if (mlr3misc::require_namespaces("sf", quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmpl("spcv_tiles",
    nsplit = c(4L, 3L), reassign = FALSE)
  resampling$instantiate(task)

  autoplot(resampling, task,
    fold_id = 1,
    show_omitted = TRUE, size = 0.7) *
  ggplot2::scale_x_continuous(breaks = seq(-79.085, -79.055, 0.01))
}

```

autoplot.ResamplingSptCVCluto

Visualization Functions for SptCV Cluto Methods.

Description

Generic S3 plot() and autoplot() (ggplot2) methods to visualize mlr3 spatiotemporal resampling objects.

Usage

```

## S3 method for class 'ResamplingSptCVCluto'
autoplot(
  object,
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  tickformat_date = "%Y-%m",
  nticks_x = 3,
  nticks_y = 3,
  point_size = 3,
  axis_label_fontsize = 11,
  ...
)

## S3 method for class 'ResamplingRepeatedSptCVCluto'
autoplot(
  object,
  task,

```

```

    fold_id = NULL,
    repeats_id = 1,
    plot_as_grid = TRUE,
    train_color = "#0072B5",
    test_color = "#E18727",
    ...
)

## S3 method for class 'ResamplingSptCVCluto'
plot(x, ...)

## S3 method for class 'ResamplingRepeatedSptCVCluto'
plot(x, ...)

```

Arguments

object	[Resampling] mlr3 spatial resampling object of class ResamplingSptCVCluto or ResamplingRepeatedSptCVCluto .
task	[TaskClassifST]/[TaskRegrST] mlr3 task object.
fold_id	[numeric] Fold IDs to plot.
plot_as_grid	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument fold_id.
train_color	[character(1)] The color to use for the training set observations.
test_color	[character(1)] The color to use for the test set observations.
tickformat_date	[character] Date format for z-axis.
nticks_x	[integer] Number of x axis breaks. Only applies to SptCVCluto.
nticks_y	[integer] Number of y axis breaks. Only applies to SptCVCluto.
point_size	[numeric] Point size of markers.
axis_label_fontsize	[integer] Font size of axis labels.
...	Passed to <code>geom_sf()</code> . Helpful for adjusting point sizes and shapes.
repeats_id	[numeric] Repetition ID to plot.

x [Resampling]
mlr3 spatial resampling object of class [ResamplingSptCVCluto](#) or [ResamplingRepeatedSptCVCluto](#).

See Also

- mlr3book chapter on on "[Spatiotemporal Visualization](#)".
- Vignette [Spatiotemporal Visualization](#).
- [autoplot.ResamplingSpCVBlock\(\)](#)
- [autoplot.ResamplingSpCVBuffer\(\)](#)
- [autoplot.ResamplingSpCVCoords\(\)](#)
- [autoplot.ResamplingSpCEnv\(\)](#)
- [autoplot.ResamplingSpCVDisc\(\)](#)
- [autoplot.ResamplingSpCVTiles\(\)](#)
- [autoplot.ResamplingCV\(\)](#)
- [autoplot.ResamplingSptCVCstf\(\)](#)

Examples

```
## Not run:
if (mlr3misc::require_namespaces(c("sf", "skmeans", "plotly"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task_st = tsk("cookfarm")
  resampling = rsmpl("sptcv_cluto", folds = 5, time_var = "Date")
  resampling$instantiate(task_st)

  # plot
  autoplot(resampling, task_st)
  autoplot(resampling, task_st, fold_id = 1)
  autoplot(resampling, task_st, fold_id = c(1, 2))
}

## End(Not run)
```

autoplot.ResamplingSptCVCstf

Visualization Functions for SptCV Cstf Methods.

Description

Generic `S3 plot()` and `autoplot()` (ggplot2) methods to visualize mlr3 spatiotemporal resampling objects.

Usage

```

## S3 method for class 'ResamplingSptCVCstf'
autoplot(
  object,
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  tickformat_date = "%Y-%m",
  nticks_x = 3,
  nticks_y = 3,
  point_size = 3,
  axis_label_fontsize = 11,
  static_image = FALSE,
  show_omitted = FALSE,
  plot3D = NULL,
  ...
)

## S3 method for class 'ResamplingRepeatedSptCVCstf'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = 1,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  tickformat_date = "%Y-%m",
  nticks_x = 3,
  nticks_y = 3,
  point_size = 3,
  axis_label_fontsize = 11,
  plot3D = NULL,
  ...
)

## S3 method for class 'ResamplingSptCVCstf'
plot(x, ...)

## S3 method for class 'ResamplingRepeatedSptCVCstf'
plot(x, ...)

```

Arguments

object [Resampling]
mlr3 spatial resampling object of class [ResamplingSptCVCstf](#) or [Resamplin-](#)

	gRepeatedSptCVCstf .
task	[TaskClassifST]/[TaskRegrST] mlr3 task object.
fold_id	[numeric] Fold IDs to plot.
plot_as_grid	[logical(1)] Should a gridded plot using via patchwork be created? If FALSE a list with of ggplot2 objects is returned. Only applies if a numeric vector is passed to argument fold_id.
train_color	[character(1)] The color to use for the training set observations.
test_color	[character(1)] The color to use for the test set observations.
tickformat_date	[character] Date format for z-axis.
nticks_x	[integer] Number of x axis breaks.
nticks_y	[integer] Number of y axis breaks.
point_size	[numeric] Point size of markers.
axis_label_fontsize	[integer] Font size of axis labels.
static_image	[logical] Whether to create a static image from the plotly plot via <code>plotly::orca()</code> . This requires the orca utility to be available. See https://github.com/plotly/orca for more information. When used, by default a file named <code>plot.png</code> is created in the current working directory.
show_omitted	[logical] Whether to show points not used in train or test set for the current fold.
plot3D	[logical] Whether to create a 2D image via ggplot2 or a 3D plot via plotly .
...	Passed down to <code>plotly::orca()</code> . Only effective when <code>static_image = TRUE</code> .
repeats_id	[numeric] Repetition ID to plot.
x	[Resampling] mlr3 spatial resampling object of class ResamplingSptCVCstf or ResamplingRepeatedSptCVCstf .

Details

This method requires to set argument `fold_id` and no plot containing all partitions can be created. This is because the method does not make use of all observations but only a subset of them (many observations are left out). Hence, train and test sets of one fold are not re-used in other folds as in other methods and plotting these without a train/test indicator would not make sense.

2D vs 3D plotting

This method has both a 2D and a 3D plotting method. The 2D method returns a **ggplot** with x and y axes representing the spatial coordinates. The 3D method uses **plotly** to create an interactive 3D plot. Set `plot3D = TRUE` to use the 3D method.

Note that spatiotemporal datasets usually suffer from overplotting in 2D mode.

See Also

- mlr3book chapter on on "[Spatiotemporal Visualization](#)".
- Vignette [Spatiotemporal Visualization](#).
- `autoplot.ResamplingSpCVBlock()`
- `autoplot.ResamplingSpCVBuffer()`
- `autoplot.ResamplingSpCVCoords()`
- `autoplot.ResamplingSpCVEnv()`
- `autoplot.ResamplingCV()`
- `autoplot.ResamplingSptCVCluto()`

Examples

```
if (mlr3misc::require_namespaces(c("sf", "plotly"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task_st = tsk("cookfarm")
  resampling = rsmpl("sptcv_cstf",
    folds = 5, time_var = "Date",
    space_var = "SOURCEID")
  resampling$instantiate(task_st)

  # with both `space_var` and `time_var` (LLTO), the omitted observations per
  # fold can be shown by setting `show_omitted = TRUE`
  autoplot(resampling, task_st, fold_id = 1, show_omitted = TRUE)
}
```

mlr_tasks_cookfarm *Cookfarm Profiles Regression Task*

Description

The R.J. Cook Agronomy Farm (cookfarm) is a Long-Term Agroecosystem Research Site operated by Washington State University, located near Pullman, Washington, USA. Contains spatio-temporal (3D+T) measurements of three soil properties and a number of spatial and temporal regression covariates.

Here, only the "Profiles" dataset is used from the collection. The Date column was appended from the readings dataset. 500 random samples were drawn from the complete sample.

The dataset was borrowed and adapted from package GSIF which was on archived on CRAN in 2021-03.

Usage

```
data(cookfarm_sample)
```

Format

[R6::R6Class](#) inheriting from [TaskRegr](#).

Usage

```
mlr_tasks$get("cookfarm")
tsk("cookfarm")
```

References

Gasch, C.K., Hengl, T., Gräler, B., Meyer, H., Magney, T., Brown, D.J., 2015. Spatio-temporal interpolation of soil water, temperature, and electrical conductivity in 3D+T: the Cook Agronomy Farm data set. *Spatial Statistics*, 14, pp.70–90.

Gasch, C.K., D.J. Brown, E.S. Brooks, M. Yourek, M. Poggio, D.R. Cobos, C.S. Campbell, 2016? Retroactive calibration of soil moisture sensors using a two-step, soil-specific correction. Submitted to *Vadose Zone Journal*.

Gasch, C.K., D.J. Brown, C.S. Campbell, D.R. Cobos, E.S. Brooks, M. Chahal, M. Poggio, 2016? A field-scale sensor network data set for monitoring and modeling the spatial and temporal variation of soil moisture in a dryland agricultural field. Submitted to *Water Resources Research*.

See Also

Dictionary of Tasks: [mlr_tasks](#)

`as.data.table(mlr_tasks)` for a complete table of all (also dynamically created) [Tasks](#).

Other Task: [TaskClassifST](#), [TaskRegrST](#), [mlr_tasks_diplodia](#), [mlr_tasks_ecuador](#)

mlr_tasks_diplodia *Diplodia Classification Task*

Description

Data set created by Patrick Schratz, University of Jena (Germany) and Eugenia Iturrutxa, NEIKER, Vitoria-Gasteiz (Spain). This dataset should be cited as Schratz et al. (2019) (see reference below). The publication also contains additional information on data collection. The data set provided here shows infections of trees by the pathogen *Diplodia Sapinea* in the Basque Country in Spain. Predictors are environmental variables like temperature, precipitation, soil and more.

Usage

```
data(diplodia)
```

Format

[R6::R6Class](#) inheriting from [TaskClassif](#).

Usage

```
mlr_tasks$get("diplodia")  
tsk("diplodia")
```

References

Schratz P, Muenchow J, Iturritxa E, Richter J, Brenning A (2019). "Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data." *Ecological Modelling*, **406**, 109–120. doi: [10.1016/j.ecolmodel.2019.06.002](https://doi.org/10.1016/j.ecolmodel.2019.06.002).

See Also

[Dictionary of Tasks: mlr_tasks](#)

as `data.table(mlr_tasks)` for a complete table of all (also dynamically created) [Tasks](#).

Other Task: [TaskClassifST](#), [TaskRegrST](#), [mlr_tasks_cookfarm](#), [mlr_tasks_ecuador](#)

mlr_tasks_ecuador	<i>Ecuador Classification Task</i>
-------------------	------------------------------------

Description

Data set created by Jannes Muenchow, University of Erlangen-Nuernberg, Germany. This dataset should be cited as Muenchow et al. (2012) (see reference below). The publication also contains additional information on data collection and the geomorphology of the area. The data set provided here is (a subset of) the one from the 'natural' part of the RBSF area and corresponds to landslide distribution in the year 2000.

Usage

```
data(ecuador)
```

Format

[R6::R6Class](#) inheriting from [TaskClassif](#).

Usage

```
mlr_tasks$get("ecuador")  
tsk("ecuador")
```

References

Muenchow, J., Brenning, A., Richter, M., 2012. Geomorphic process rates of landslides along a humidity gradient in the tropical Andes. *Geomorphology*, 139-140: 271-284.

See Also

[Dictionary of Tasks: mlr_tasks](#)

`as.data.table(mlr_tasks)` for a complete table of all (also dynamically created) [Tasks](#).

Other Task: [TaskClassifST](#), [TaskRegrST](#), [mlr_tasks_cookfarm](#), [mlr_tasks_diplodia](#)

ResamplingRepeatedSpCVBlock

(blockCV) Repeated spatial block resampling

Description

This function creates spatially separated folds based on a pre-specified distance. It assigns blocks to the training and testing folds **randomly**, **systematically** or in a **checkerboard pattern**. The distance (`theRange`) should be in **metres**, regardless of the unit of the reference system of the input data (for more information see the details section). By default, the function creates blocks according to the extent and shape of the study area, assuming that the user has considered the landscape for the given species and case study. Alternatively, blocks can solely be created based on species spatial data. Blocks can also be offset so the origin is not at the outer corner of the rasters. Instead of providing a distance, the blocks can also be created by specifying a number of rows and/or columns and divide the study area into vertical or horizontal bins, as presented in Wenger & Olden (2012) and Bahn & McGill (2012). Finally, the blocks can be specified by a user-defined spatial polygon layer.

Details

To keep the consistency, all the functions use **metres** as their unit. In this function, when the input map has geographic coordinate system (decimal degrees), the block size is calculated based on dividing `theRange` by 111325 (the standard distance of a degree in metres, on the Equator) to change the unit to degree. This value is optional and can be changed by user via `degMetre` argument.

The `xOffset` and `yOffset` can be used to change the spatial position of the blocks. It can also be used to assess the sensitivity of analysis results to shifting in the blocking arrangements. These options are available when `theRange` is defined. By default the region is located in the middle of the blocks and by setting the offsets, the blocks will shift.

Roberts et. al. (2017) suggest that blocks should be substantially bigger than the range of spatial autocorrelation (in model residual) to obtain realistic error estimates, while a buffer with the size of the spatial autocorrelation range would result in a good estimation of error. This is because of the so-called edge effect (O'Sullivan & Unwin, 2014), whereby points located on the edges of the blocks of opposite sets are not separated spatially. Blocking with a buffering strategy overcomes this issue (see [buffering](#)).

mlr3spatiotempcv notes

By default `blockCV::spatialBlock()` does not allow the creation of multiple repetitions. `mlr3spatiotempcv` adds support for this when using the `range` argument for fold creation. When supplying a vector of `length(repeats)` for argument `range`, these different settings will be used to create folds which differ among the repetitions.

Multiple repetitions are not possible when using the "row & cols" approach because the created folds will always be the same.

The 'Description' and 'Details' fields are inherited from the respective upstream function.

For a list of available arguments, please see [blockCV::spatialBlock](#).

Super class

`mlr3::Resampling` -> `ResamplingRepeatedSpCVBlock`

Public fields

`blocks` `sf` | list of `sf` objects

Polygons (`sf` objects) as returned by `blockCV` which grouped observations into partitions.

Active bindings

`iters` `integer(1)`

Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods**Public methods:**

- [ResamplingRepeatedSpCVBlock\\$new\(\)](#)
- [ResamplingRepeatedSpCVBlock\\$folds\(\)](#)
- [ResamplingRepeatedSpCVBlock\\$repeats\(\)](#)
- [ResamplingRepeatedSpCVBlock\\$instantiate\(\)](#)
- [ResamplingRepeatedSpCVBlock\\$clone\(\)](#)

Method `new()`: Create an "spatial block" repeated resampling instance.

For a list of available arguments, please see [blockCV::spatialBlock](#).

Usage:

```
ResamplingRepeatedSpCVBlock$new(id = "repeated_spcv_block")
```

Arguments:

`id` `character(1)`

Identifier for the resampling strategy.

Method `folds()`: Translates iteration numbers to fold number.

Usage:

```
ResamplingRepeatedSpCVBlock$folds(iters)
```

Arguments:

iters integer()
Iteration number.

Method repeats(): Translates iteration numbers to repetition number.

Usage:
ResamplingRepeatedSpCVBlock\$repeats(iters)
Arguments:
iters integer()
Iteration number.

Method instantiate(): Materializes fixed training and test splits for a given task.

Usage:
ResamplingRepeatedSpCVBlock\$instantiate(task)
Arguments:
task [Task](#)
A task to instantiate.

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

Usage:
ResamplingRepeatedSpCVBlock\$clone(deep = FALSE)
Arguments:
deep Whether to make a deep clone.

References

Valavi R, Elith J, Lahoz-Monfort JJ, Guillera-Arroita G (2018). “blockCV: an R package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models.” *bioRxiv*. doi: [10.1101/357798](https://doi.org/10.1101/357798).

Examples

```
if (mLr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mLr3)
  task = tsk("diplodia")

  # Instantiate Resampling
  rrcv = rsmpl("repeated_spcv_block",
    folds = 3, repeats = 2,
    range = c(5000L, 10000L))
  rrcv$instantiate(task)

  # Individual sets:
  rrcv$iters
  rrcv$folds(1:6)
  rrcv$repeats(1:6)

  # Individual sets:
  rrcv$train_set(1)
```

```

rrcv$test_set(1)
intersect(rrcv$train_set(1), rrcv$test_set(1))

# Internal storage:
rrcv$instance # table
}

```

ResamplingRepeatedSpCVCoords

(sperrorest) Repeated coordinate-based k-means clustering

Description

partition_cv creates a [resampling](#) object for length(repetition)-repeated nfold-fold cross-validation.

Details

This function does not actually perform a cross-validation or partition the data set itself; it simply creates a data structure containing the indices of training and test samples.

mlr3spatiotempcv notes

The 'Description' and 'Details' fields are inherited from the respective upstream function.

For a list of available arguments, please see [sperrorest::partition_cv](#).

Super class

[mlr3::Resampling](#) -> ResamplingRepeatedSpCVCoords

Active bindings

iters integer(1)

Returns the number of resampling iterations, depending on the values stored in the param_set.

Methods

Public methods:

- [ResamplingRepeatedSpCVCoords\\$new\(\)](#)
- [ResamplingRepeatedSpCVCoords\\$folds\(\)](#)
- [ResamplingRepeatedSpCVCoords\\$repeats\(\)](#)
- [ResamplingRepeatedSpCVCoords\\$instantiate\(\)](#)
- [ResamplingRepeatedSpCVCoords\\$clone\(\)](#)

Method new(): Create an "coordinate-based" repeated resampling instance.

For a list of available arguments, please see [sperrorest::partition_cv](#).

Usage:

```
ResamplingRepeatedSpCVCoords$new(id = "repeated_spcv_coords")
```

Arguments:

```
id character(1)
  Identifier for the resampling strategy.
```

Method folds(): Translates iteration numbers to fold number.

Usage:

```
ResamplingRepeatedSpCVCoords$folds(iters)
```

Arguments:

```
iters integer()
  Iteration number.
```

Method repeats(): Translates iteration numbers to repetition number.

Usage:

```
ResamplingRepeatedSpCVCoords$repeats(iters)
```

Arguments:

```
iters integer()
  Iteration number.
```

Method instantiate(): Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingRepeatedSpCVCoords$instantiate(task)
```

Arguments:

```
task Task
  A task to instantiate.
```

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

Usage:

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

Arguments:

```
deep Whether to make a deep clone.
```

References

Brenning A (2012). "Spatial cross-validation and bootstrap for the assessment of prediction rules in remote sensing: The R package sperrorest." In *2012 IEEE International Geoscience and Remote Sensing Symposium*. doi: [10.1109/igarss.2012.6352393](https://doi.org/10.1109/igarss.2012.6352393).

Examples

```
library(mlr3)
task = tsk("diplodia")

# Instantiate Resampling
rrcv = rsm("repeated_spcv_coords", folds = 3, repeats = 5)
```

```

rrcv$instantiate(task)

# Individual sets:
rrcv$iters
rrcv$folds(1:6)
rrcv$repeats(1:6)

# Individual sets:
rrcv$train_set(1)
rrcv$test_set(1)
intersect(rrcv$train_set(1), rrcv$test_set(1))

# Internal storage:
rrcv$instance # table

```

ResamplingRepeatedSpCVDisc

(sperrorest) Repeated spatial "disc" resampling

Description

(sperrorest) Repeated spatial "disc" resampling

(sperrorest) Repeated spatial "disc" resampling

Super class

[mlr3::Resampling](#) -> ResamplingRepeatedSpCVDisc

Active bindings

`iters` integer(1)

Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods

Public methods:

- [ResamplingRepeatedSpCVDisc\\$new\(\)](#)
- [ResamplingRepeatedSpCVDisc\\$folds\(\)](#)
- [ResamplingRepeatedSpCVDisc\\$repeats\(\)](#)
- [ResamplingRepeatedSpCVDisc\\$instantiate\(\)](#)
- [ResamplingRepeatedSpCVDisc\\$clone\(\)](#)

Method `new()`: Create a "Spatial 'Disc' resampling" resampling instance.

For a list of available arguments, please see [sperrorest::partition_disc](#).

Usage:

```
ResamplingRepeatedSpCVDisc$new(id = "repeated_spcv_disc")
```

Arguments:

id character(1)
Identifier for the resampling strategy.

Method folds(): Translates iteration numbers to fold number.

Usage:

```
ResamplingRepeatedSpCVDisc$folds(iters)
```

Arguments:

iters integer()
Iteration number.

Method repeats(): Translates iteration numbers to repetition number.

Usage:

```
ResamplingRepeatedSpCVDisc$repeats(iters)
```

Arguments:

iters integer()
Iteration number.

Method instantiate(): Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingRepeatedSpCVDisc$instantiate(task)
```

Arguments:

task [Task](#)
A task to instantiate.

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Brenning A (2012). "Spatial cross-validation and bootstrap for the assessment of prediction rules in remote sensing: The R package sperrorest." In *2012 IEEE International Geoscience and Remote Sensing Symposium*. doi: [10.1109/igarss.2012.6352393](https://doi.org/10.1109/igarss.2012.6352393).

Examples

```
library(mlr3)
task = tsk("ecuador")

# Instantiate Resampling
rrcv = rsmp("repeated_spcv_disc",
  folds = 3L, repeats = 2,
  radius = 200L, buffer = 200L)
```



```

rrcv$instantiate(task)

# Individual sets:
rrcv$iters
rrcv$folds(1:6)
rrcv$repeats(1:6)

# Individual sets:
rrcv$train_set(1)
rrcv$test_set(1)
intersect(rrcv$train_set(1), rrcv$test_set(1))

# Internal storage:
rrcv$instance # table

```

ResamplingRepeatedSpCEnv

(blockCV) Repeated "environmental blocking" resampling

Description

Environmental blocking for cross-validation. This function uses clustering methods to specify sets of similar environmental conditions based on the input covariates. Species data corresponding to any of these groups or clusters are assigned to a fold. This function does the clustering in raster space and species data. Clustering is done using [kmeans](#) for both approaches. This function works on single or multiple raster files; multiple rasters need to be in a raster brick or stack format.

Details

As k-means algorithms use Euclidean distance to estimate clusters, the input covariates should be quantitative variables. Since variables with wider ranges of values might dominate the clusters and bias the environmental clustering (Hastie et al., 2009), all the input rasters are first standardized within the function. This is done either by normalizing based on subtracting the mean and dividing by the standard deviation of each raster (the default) or optionally by standardizing using linear scaling to constrain all raster values between 0 and 1.

By default, the clustering is done in the raster space. In this approach the clusters will be consistent throughout the region and across species (in the same region). However, this may result in a cluster(s) that covers none of the species records (the spatial location of response samples), especially when species data is not dispersed throughout the region or the number of clusters (k or folds) is high. In this case, the number of folds is less than specified k. If `rasterBlock = FALSE`, the clustering will be done in species points and the number of the folds will be the same as k.

Note that the input raster layer should cover all the species points, otherwise an error will rise. The records with no raster value should be deleted prior to the analysis or another raster layer would be provided.

mlr3spatiotempcv notes

The 'Description' and 'Details' fields are inherited from the respective upstream function.

For a list of available arguments, please see [blockCV::envBlock](#).

Super class

`mlr3::Resampling` -> ResamplingRepeatedSpCEnv

Active bindings

`iters` integer(1)

Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods**Public methods:**

- `ResamplingRepeatedSpCEnv$new()`
- `ResamplingRepeatedSpCEnv$folds()`
- `ResamplingRepeatedSpCEnv$repeats()`
- `ResamplingRepeatedSpCEnv$instantiate()`
- `ResamplingRepeatedSpCEnv$clone()`

Method `new()`: Create an "Environmental Block" repeated resampling instance.

For a list of available arguments, please see `blockCV::envBlock`.

Usage:

```
ResamplingRepeatedSpCEnv$new(id = "repeated_spcv_env")
```

Arguments:

`id` character(1)

Identifier for the resampling strategy.

Method `folds()`: Translates iteration numbers to fold number.

Usage:

```
ResamplingRepeatedSpCEnv$folds(iters)
```

Arguments:

`iters` integer()

Iteration number.

Method `repeats()`: Translates iteration numbers to repetition number.

Usage:

```
ResamplingRepeatedSpCEnv$repeats(iters)
```

Arguments:

`iters` integer()

Iteration number.

Method `instantiate()`: Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingRepeatedSpCEnv$instantiate(task)
```

Arguments:

task [Task](#)

A task to instantiate.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Valavi R, Elith J, Lahoz-Monfort JJ, Guillera-Arroita G (2018). “blockCV: an R package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models.” *bioRxiv*. doi: [10.1101/357798](https://doi.org/10.1101/357798).

Examples

```
if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mlr3)
  task = tsk("ecuador")

  # Instantiate Resampling
  rrcv = rsmp("repeated_spcv_env", folds = 4, repeats = 2)
  rrcv$instantiate(task)

  # Individual sets:
  rrcv$train_set(1)
  rrcv$test_set(1)
  intersect(rrcv$train_set(1), rrcv$test_set(1))

  # Internal storage:
  rrcv$instance
}
```

ResamplingRepeatedSpCVTiles

(sperrorest) Repeated spatial "tiles" resampling

Description

`partition_tiles` divides the study area into a specified number of rectangular tiles. Optionally small partitions can be merged with adjacent tiles to achieve a minimum number or percentage of samples in each tile.

mlr3spatiotempcv notes

The 'Description' and 'Note' fields are inherited from the respective upstream function.

For a list of available arguments, please see [sperrorest::partition_tiles](#).

This method is similar to [ResamplingSpCVBlock](#).

Super class

`mlr3::Resampling` -> `ResamplingRepeatedSpCVTiles`

Active bindings

`iters` integer(1)

Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods**Public methods:**

- `ResamplingRepeatedSpCVTiles$new()`
- `ResamplingRepeatedSpCVTiles$folds()`
- `ResamplingRepeatedSpCVTiles$repeats()`
- `ResamplingRepeatedSpCVTiles$instantiate()`
- `ResamplingRepeatedSpCVTiles$clone()`

Method `new()`: Create a "Spatial 'Tiles' resampling" resampling instance.

For a list of available arguments, please see [sperrorest::partition_tiles](#).

Usage:

```
ResamplingRepeatedSpCVTiles$new(id = "repeated_spcv_tiles")
```

Arguments:

`id` character(1)

Identifier for the resampling strategy.

Method `folds()`: Translates iteration numbers to fold number.

Usage:

```
ResamplingRepeatedSpCVTiles$folds(iters)
```

Arguments:

`iters` integer()

Iteration number.

Method `repeats()`: Translates iteration numbers to repetition number.

Usage:

```
ResamplingRepeatedSpCVTiles$repeats(iters)
```

Arguments:

`iters` integer()

Iteration number.

Method `instantiate()`: Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingRepeatedSpCVTiles$instantiate(task)
```

Arguments:

task [Task](#)

A task to instantiate.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Note

Default parameter settings may change in future releases. This function, especially the rotation and shifting part of it and the algorithm for cleaning up small tiles is still a bit experimental. Use with caution. For non-zero offsets (`offset!=none`), the number of tiles may actually be greater than `nsplit[1]*nsplit[2]` because of fractional tiles lurking into the study region. `reassign=TRUE` with suitable thresholds is therefore recommended for non-zero (including random) offsets.

References

Brenning A (2012). "Spatial cross-validation and bootstrap for the assessment of prediction rules in remote sensing: The R package `sperrorest`." In *2012 IEEE International Geoscience and Remote Sensing Symposium*. doi: [10.1109/igarss.2012.6352393](https://doi.org/10.1109/igarss.2012.6352393).

See Also

`ResamplingSpCVBlock`

Examples

```
if (mlr3misc::require_namespaces("sperrorest", quietly = TRUE)) {
  library(mlr3)
  task = tsk("ecuador")

  # Instantiate Resampling
  rrcv = rsmpl("repeated_spcv_tiles",
    repeats = 2,
    nsplit = c(4L, 3L), reassign = FALSE)
  rrcv$instantiate(task)

  # Individual sets:
  rrcv$iters
  rrcv$folds(10:12)
  rrcv$repeats(10:12)

  # Individual sets:
  rrcv$train_set(1)
  rrcv$test_set(1)
  intersect(rrcv$train_set(1), rrcv$test_set(1))

  # Internal storage:
```

```

rrcv$instance # table
}

```

ResamplingRepeatedSptCVCluto

(skmeans) Repeated spatiotemporal clustering resampling

Description

Spatiotemporal cluster partitioning via the `vcluster` executable of the CLUTO clustering application.

This partitioning method relies on the external CLUTO library. To use it, CLUTO's executables need to be downloaded and installed into this package.

See <https://gist.github.com/pat-s/6430470cf817050e27d26c43c0e9be72> for an installation approach that should work on Windows and Linux. macOS is not supported by CLUTO.

Before using this method, please check the restrictive copyright shown below.

Details

By default, `-clmethod='direct'` is passed to the `vcluster` executable in contrast to the upstream default `-clmethod='rb'`. There is no evidence or research that this method is the best among the available ones ("rb", "rbr", "direct", "agglo", "graph", "bagglo"). Also, various other parameters can be set via argument `cluto_parameters` to achieve different clustering results.

Parameter `-clusterfile` is handled by **skmeans** and cannot be changed.

Copyright

CLUTO's copyright is as follows:

The CLUTO package is copyrighted by the Regents of the University of Minnesota. It can be freely used for educational and research purposes by non-profit institutions and US government agencies only. Other organizations are allowed to use CLUTO only for evaluation purposes, and any further uses will require prior approval. The software may not be sold or redistributed without prior approval. One may make copies of the software for their use provided that the copies, are not sold or distributed, are used under the same terms and conditions. As unestablished research software, this code is provided on an "as is" basis without warranty of any kind, either expressed or implied. The downloading, or executing any part of this software constitutes an implicit agreement to these terms. These terms and conditions are subject to change at any time without prior notice.

Super class

`mlr3::Resampling` -> `ResamplingRepeatedSptCVCluto`

Public fields

time_var [character](#)

The name of the variable which represents the time dimension. Must be of type numeric.

clmethod [character](#)

Name of the clustering method to use within vcluster. See Details for more information.

cluto_parameters [character](#)

Additional parameters to pass to vcluster. Must be given as a single character string, e.g. "param1='value1'param2='value2'". See the CLUTO documentation for a full list of supported parameters.

verbose [logical](#)

Whether to show vcluster progress and summary output.

Active bindings

iters integer(1)

Returns the number of resampling iterations, depending on the values stored in the param_set.

Methods**Public methods:**

- [ResamplingRepeatedSptCVCluto\\$new\(\)](#)
- [ResamplingRepeatedSptCVCluto\\$folds\(\)](#)
- [ResamplingRepeatedSptCVCluto\\$repeats\(\)](#)
- [ResamplingRepeatedSptCVCluto\\$instantiate\(\)](#)
- [ResamplingRepeatedSptCVCluto\\$clone\(\)](#)

Method new(): Create an repeated resampling instance using the CLUTO algorithm.

Usage:

```
ResamplingRepeatedSptCVCluto$new(
  id = "repeated_sptcv_cluto",
  time_var = NULL,
  clmethod = "direct",
  cluto_parameters = NULL,
  verbose = TRUE
)
```

Arguments:

id [character](#)(1)

Identifier for the resampling strategy.

time_var [character](#)

The name of the variable which represents the time dimension. Must be of type numeric.

clmethod [character](#)

Name of the clustering method to use within vcluster. See Details for more information.

cluto_parameters [character](#)

Additional parameters to pass to vcluster. Must be given as a single character string, e.g. "param1='value1'param2='value2'". See the CLUTO documentation for a full list of supported parameters.

verbose [logical](#)

Whether to show vcluster progress and summary output.

Method folds(): Translates iteration numbers to fold number.

Usage:

```
ResamplingRepeatedSptCVCluto$folds(iters)
```

Arguments:

iters integer()
Iteration number.

Method repeats(): Translates iteration numbers to repetition number.

Usage:

```
ResamplingRepeatedSptCVCluto$repeats(iters)
```

Arguments:

iters integer()
Iteration number.

Method instantiate(): Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingRepeatedSptCVCluto$instantiate(task)
```

Arguments:

task [Task](#)
A task to instantiate.

time_var [character](#)
The name of the variable which represents the time dimension. Must be of type numeric.

clmethod [character](#)
Name of the clustering method to use within vcluster. See Details for more information.

cluto_parameters [character](#)
Additional parameters to pass to vcluster. Must be given as a single character string, e.g. "param1='value1' param2='value2'". See the CLUTO documentation for a full list of supported parameters.

verbose [logical](#)

Whether to show vcluster progress and summary output.

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Zhao Y, Karypis G (2002). "Evaluation of Hierarchical Clustering Algorithms for Document Datasets." *11th Conference of Information and Knowledge Management (CIKM)*, 51-524. <http://glaros.dtc.umn.edu/gkhome/node/167>.

Examples

```
## Not run:
if (mlr3misc::require_namespaces("skmeans", quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("cookfarm")

  # Instantiate Resampling
  rrcv = rsmpl("repeated_sptcv_cluto", folds = 3, repeats = 5)
  rrcv$instantiate(task, time_var = "Date")

  # Individual sets:
  rrcv$iters
  rrcv$folds(1:6)
  rrcv$repeats(1:6)

  # Individual sets:
  rrcv$train_set(1)
  rrcv$test_set(1)
  intersect(rrcv$train_set(1), rrcv$test_set(1))

  # Internal storage:
  rrcv$instance # table
}

## End(Not run)
```

 ResamplingRepeatedSptCVCstf

(CAST) Repeated "leave-location-and-time-out" resampling

Description

Create spatial, temporal or spatio-temporal Folds for cross validation

Details

Using "class" is helpful in the case that data are clustered in space and are categorical. E.g This is the case for land cover classifications when training data come as training polygons. In this case the data should be split in a way that entire polygons are held back (spacevar="polygonID") but at the same time the distribution of classes should be similar in each fold (class="LUC").

mlr3spatiotempcv notes

The 'Description', 'Details' and 'Note' fields are inherited from the respective upstream function.

For a list of available arguments, please see [CAST::CreateSpacetimeFolds](#).

Super class

`mlr3::Resampling` -> `ResamplingRepeatedSptCVCstf`

Active bindings

`iters` integer(1)

Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods**Public methods:**

- `ResamplingRepeatedSptCVCstf$new()`
- `ResamplingRepeatedSptCVCstf$folds()`
- `ResamplingRepeatedSptCVCstf$repeats()`
- `ResamplingRepeatedSptCVCstf$instantiate()`
- `ResamplingRepeatedSptCVCstf$clone()`

Method `new()`: Create a "Spacetime Folds" resampling instance.

For a list of available arguments, please see [CAST::CreateSpacetimeFolds](#).

Usage:

```
ResamplingRepeatedSptCVCstf$new(id = "repeated_sptcv_cstf")
```

Arguments:

`id` character(1)

Identifier for the resampling strategy.

Method `folds()`: Translates iteration numbers to fold number.

Usage:

```
ResamplingRepeatedSptCVCstf$folds(iters)
```

Arguments:

`iters` integer()

Iteration number.

Method `repeats()`: Translates iteration numbers to repetition number.

Usage:

```
ResamplingRepeatedSptCVCstf$repeats(iters)
```

Arguments:

`iters` integer()

Iteration number.

Method `instantiate()`: Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingRepeatedSptCVCstf$instantiate(task)
```

Arguments:

task `Task`

A task to instantiate.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Note

Standard k-fold cross-validation can lead to considerable misinterpretation in spatial-temporal modelling tasks. This function can be used to prepare a Leave-Location-Out, Leave-Time-Out or Leave-Location-and-Time-Out cross-validation as target-oriented validation strategies for spatial-temporal prediction tasks. See Meyer et al. (2018) for further information.

References

Zhao Y, Karypis G (2002). “Evaluation of Hierarchical Clustering Algorithms for Document Datasets.” *11th Conference of Information and Knowledge Management (CIKM)*, 51-524. <http://glaros.dtc.umn.edu/gkhome/node/167>.

Examples

```
library(mlr3)
library(mlr3spatiotempcv)
task = tsk("cookfarm")

# Instantiate Resampling
rrcv = rsmpl("repeated_sptcv_cstf", folds = 3, repeats = 5, time_var = "Date")
rrcv$instantiate(task)
# Individual sets:
rrcv$iters
rrcv$folds(1:6)
rrcv$repeats(1:6)

# Individual sets:
rrcv$train_set(1)
rrcv$test_set(1)
intersect(rrcv$train_set(1), rrcv$test_set(1))

# Internal storage:
rrcv$instance # table
```

 ResamplingSpCVBlock *(blockCV) Spatial block resampling*

Description

This function creates spatially separated folds based on a pre-specified distance. It assigns blocks to the training and testing folds **randomly**, **systematically** or in a **checkerboard pattern**. The distance (`theRange`) should be in **metres**, regardless of the unit of the reference system of the input data (for more information see the details section). By default, the function creates blocks according to the extent and shape of the study area, assuming that the user has considered the landscape for the given species and case study. Alternatively, blocks can solely be created based on species spatial data. Blocks can also be offset so the origin is not at the outer corner of the rasters. Instead of providing a distance, the blocks can also be created by specifying a number of rows and/or columns and divide the study area into vertical or horizontal bins, as presented in Wenger & Olden (2012) and Bahn & McGill (2012). Finally, the blocks can be specified by a user-defined spatial polygon layer.

Details

To keep the consistency, all the functions use **metres** as their unit. In this function, when the input map has geographic coordinate system (decimal degrees), the block size is calculated based on dividing `theRange` by 111325 (the standard distance of a degree in metres, on the Equator) to change the unit to degree. This value is optional and can be changed by user via `degMetre` argument.

The `xOffset` and `yOffset` can be used to change the spatial position of the blocks. It can also be used to assess the sensitivity of analysis results to shifting in the blocking arrangements. These options are available when `theRange` is defined. By default the region is located in the middle of the blocks and by setting the offsets, the blocks will shift.

Roberts et. al. (2017) suggest that blocks should be substantially bigger than the range of spatial autocorrelation (in model residual) to obtain realistic error estimates, while a buffer with the size of the spatial autocorrelation range would result in a good estimation of error. This is because of the so-called edge effect (O'Sullivan & Unwin, 2014), whereby points located on the edges of the blocks of opposite sets are not separated spatially. Blocking with a buffering strategy overcomes this issue (see [buffering](#)).

mlr3spatiotempcv notes

By default `blockCV::spatialBlock()` does not allow the creation of multiple repetitions. `mlr3spatiotempcv` adds support for this when using the `range` argument for fold creation. When supplying a vector of `length(repeats)` for argument `range`, these different settings will be used to create folds which differ among the repetitions.

Multiple repetitions are not possible when using the "row & cols" approach because the created folds will always be the same.

The 'Description' and 'Details' fields are inherited from the respective upstream function.

For a list of available arguments, please see `blockCV::spatialBlock`.

Super class

`mlr3::Resampling` -> ResamplingSpCVBlock

Public fields

`blocks` `sf` | list of `sf` objects

Polygons (`sf` objects) as returned by **blockCV** which grouped observations into partitions.

Active bindings

`iters` `integer`(1)

Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods**Public methods:**

- `ResamplingSpCVBlock$new()`
- `ResamplingSpCVBlock$instantiate()`
- `ResamplingSpCVBlock$clone()`

Method `new()`: Create an "spatial block" resampling instance.

For a list of available arguments, please see `blockCV::spatialBlock()`.

Usage:

```
ResamplingSpCVBlock$new(id = "spcv_block")
```

Arguments:

`id` `character`(1)

Identifier for the resampling strategy.

Method `instantiate()`: Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingSpCVBlock$instantiate(task)
```

Arguments:

`task` `Task`

A task to instantiate.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Valavi R, Elith J, Lahoz-Monfort JJ, Guillera-Arroita G (2018). "blockCV: an R package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models." *bioRxiv*. doi: [10.1101/357798](https://doi.org/10.1101/357798).

Examples

```

if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mlr3)
  task = tsk("ecuador")

  # Instantiate Resampling
  rcv = rsmpl("spcv_block", range = 1000L)
  rcv$instantiate(task)

  # Individual sets:
  rcv$train_set(1)
  rcv$test_set(1)
  intersect(rcv$train_set(1), rcv$test_set(1))

  # Internal storage:
  rcv$instance
}

```

ResamplingSpCVBuffer *(blockCV) Spatial buffering resampling*

Description

This function generates spatially separated train and test folds by considering buffers of the specified distance around each observation point. This approach is a form of *leave-one-out* cross-validation. Each fold is generated by excluding nearby observations around each testing point within the specified distance (ideally the range of spatial autocorrelation, see [spatialAutoRange](#)). In this method, the testing set never directly abuts a training presence or absence (0s and 1s i.e. the response class). For more information see the details section.

Details

When working with presence-background (presence and pseudo-absence) data (specified by `spDataType` argument), only presence records are used for specifying the folds. Consider a target presence point. The buffer is defined around this target point, using the specified range (`theRange`). The testing fold comprises the target presence point and all background points within the buffer (this is the default. If `addBG = FALSE` the background points are ignored). Any non-target presence points inside the buffer are excluded. All points (presence and background) outside of buffer are used for the training set. The methods cycles through all the *presence* data, so the number of folds is equal to the number of presence points in the dataset.

For presence-absence data, folds are created based on all records, both presences and absences. As above, a target observation (presence or absence) forms a test point, all presence and absence points other than the target point within the buffer are ignored, and the training set comprises all presences and absences outside the buffer. Apart from the folds, the number of *training-presence*, *training-absence*, *testing-presence* and *testing-absence* records is stored and returned in the records table. If `species = NULL` (no column with 0s and 1s is defined), the procedure is like presence-absence data. All other types of data (continuous, count or multi-class responses) should be used like this.

mlr3spatiotempcv notes

The 'Description' and 'Details' fields are inherited from the respective upstream function. For a list of available arguments, please see [blockCV::buffering](#).

Super class

`mlr3::Resampling` -> `ResamplingSpCVBuffer`

Active bindings

`iters` `integer(1)`

Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods**Public methods:**

- `ResamplingSpCVBuffer$new()`
- `ResamplingSpCVBuffer$instantiate()`
- `ResamplingSpCVBuffer$clone()`

Method `new()`: Create an "Environmental Block" resampling instance.

For a list of available arguments, please see [blockCV::buffering\(\)](#).

Usage:

```
ResamplingSpCVBuffer$new(id = "spcv_buffer")
```

Arguments:

`id` `character(1)`

Identifier for the resampling strategy.

Method `instantiate()`: Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingSpCVBuffer$instantiate(task)
```

Arguments:

`task` `Task`

A task to instantiate.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Valavi R, Elith J, Lahoz-Monfort JJ, Guillera-Arroita G (2018). "blockCV: an R package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models." *bioRxiv*. doi: [10.1101/357798](https://doi.org/10.1101/357798).

See Also

ResamplingSpCVDisc

Examples

```
if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mlr3)
  task = tsk("ecuador")

  # Instantiate Resampling
  rcv = rsmpl("spcv_buffer", theRange = 10000)
  rcv$instantiate(task)

  # Individual sets:
  rcv$train_set(1)
  rcv$test_set(1)
  intersect(rcv$train_set(1), rcv$test_set(1))

  # Internal storage:
  # rcv$instance
}
```

ResamplingSpCVCoords *(sperrorest) Coordinate-based k-means clustering*

Description

partition_cv creates a [resampling](#) object for length(repetition)-repeated nfold-fold cross-validation.

Details

This function does not actually perform a cross-validation or partition the data set itself; it simply creates a data structure containing the indices of training and test samples.

mlr3spatiotempcv notes

The 'Description' and 'Details' fields are inherited from the respective upstream function.

For a list of available arguments, please see [sperrorest::partition_cv](#).

Super class

[mlr3::Resampling](#) -> ResamplingSpCVCoords

Active bindings

iters integer(1)

Returns the number of resampling iterations, depending on the values stored in the param_set.

Methods

Public methods:

- [ResamplingSpCVCoords\\$new\(\)](#)
- [ResamplingSpCVCoords\\$instantiate\(\)](#)
- [ResamplingSpCVCoords\\$clone\(\)](#)

Method `new()`: Create an "coordinate-based" repeated resampling instance. For a list of available arguments, please see [sperrorest::partition_cv](#).

Usage:

```
ResamplingSpCVCoords$new(id = "spcv_coords")
```

Arguments:

`id` character(1)
Identifier for the resampling strategy.

Method `instantiate()`: Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingSpCVCoords$instantiate(task)
```

Arguments:

`task` [Task](#)
A task to instantiate.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Brenning A (2012). "Spatial cross-validation and bootstrap for the assessment of prediction rules in remote sensing: The R package sperrorest." In *2012 IEEE International Geoscience and Remote Sensing Symposium*. doi: [10.1109/igarss.2012.6352393](https://doi.org/10.1109/igarss.2012.6352393).

Examples

```
library(mlr3)
task = tsk("ecuador")

# Instantiate Resampling
rcv = rsm("spcv_coords", folds = 5)
rcv$instantiate(task)

# Individual sets:
rcv$train_set(1)
rcv$test_set(1)
# check that no obs are in both sets
```

```
intersect(rcv$train_set(1), rcv$test_set(1)) # good!

# Internal storage:
rcv$instance # table
```

ResamplingSpCVDisc *(sperrorest) Spatial "disc" resampling*

Description

`partition_disc` partitions the sample into training and tests set by selecting circular test areas (possibly surrounded by an exclusion buffer) and using the remaining samples as training samples (leave-one-disc-out cross-validation). `partition_loo` creates training and test sets for leave-one-out cross-validation with (optional) buffer.

mlr3spatiotempcv notes

The 'Description' and 'Note' fields are inherited from the respective upstream function.

For a list of available arguments, please see [sperrorest::partition_disc](#).

This method is similar to [ResamplingSpCVBuffer](#).

Super class

[mlr3::Resampling](#) -> ResamplingSpCVDisc

Active bindings

`iters` integer(1)
Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods

Public methods:

- [ResamplingSpCVDisc\\$new\(\)](#)
- [ResamplingSpCVDisc\\$instantiate\(\)](#)
- [ResamplingSpCVDisc\\$clone\(\)](#)

Method `new()`: Create a "Spatial 'Disc' resampling" resampling instance.

For a list of available arguments, please see [sperrorest::partition_disc](#).

Usage:

```
ResamplingSpCVDisc$new(id = "spcv_disc")
```

Arguments:

`id` character(1)
Identifier for the resampling strategy.

Method `instantiate()`: Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingSpCVDisc$instantiate(task)
```

Arguments:

```
task Task
```

A task to instantiate.

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

Usage:

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

Arguments:

```
deep Whether to make a deep clone.
```

Note

Test area discs are centered at (random) samples, not at general random locations. Test area discs may (and likely will) overlap independently of the value of `replace`. `replace` only controls the replacement of the center point of discs when drawing center points from the samples.

References

Brenning A (2012). “Spatial cross-validation and bootstrap for the assessment of prediction rules in remote sensing: The R package `sperrorest`.” In *2012 IEEE International Geoscience and Remote Sensing Symposium*. doi: [10.1109/igarss.2012.6352393](https://doi.org/10.1109/igarss.2012.6352393).

See Also

`ResamplingSpCVBuffer`

Examples

```
library(mlr3)
task = tsk("ecuador")

# Instantiate Resampling
rcv = rsm("spcv_disc", folds = 3L, radius = 200L, buffer = 200L)
rcv$instantiate(task)

# Individual sets:
rcv$train_set(1)
rcv$test_set(1)
# check that no obs are in both sets
intersect(rcv$train_set(1), rcv$test_set(1)) # good!

# Internal storage:
rcv$instance # table
```

ResamplingSpCEnv *(blockCV) "Environmental blocking" resampling*

Description

Environmental blocking for cross-validation. This function uses clustering methods to specify sets of similar environmental conditions based on the input covariates. Species data corresponding to any of these groups or clusters are assigned to a fold. This function does the clustering in raster space and species data. Clustering is done using [kmeans](#) for both approaches. This function works on single or multiple raster files; multiple rasters need to be in a raster brick or stack format.

Details

As k-means algorithms use Euclidean distance to estimate clusters, the input covariates should be quantitative variables. Since variables with wider ranges of values might dominate the clusters and bias the environmental clustering (Hastie et al., 2009), all the input rasters are first standardized within the function. This is done either by normalizing based on subtracting the mean and dividing by the standard deviation of each raster (the default) or optionally by standardizing using linear scaling to constrain all raster values between 0 and 1.

By default, the clustering is done in the raster space. In this approach the clusters will be consistent throughout the region and across species (in the same region). However, this may result in a cluster(s) that covers none of the species records (the spatial location of response samples), especially when species data is not dispersed throughout the region or the number of clusters (k or folds) is high. In this case, the number of folds is less than specified k. If `rasterBlock = FALSE`, the clustering will be done in species points and the number of the folds will be the same as k.

Note that the input raster layer should cover all the species points, otherwise an error will rise. The records with no raster value should be deleted prior to the analysis or another raster layer would be provided.

mlr3spatiotempcv notes

The 'Description' and 'Details' fields are inherited from the respective upstream function.

For a list of available arguments, please see [blockCV::envBlock](#).

Super class

[mlr3::Resampling](#) -> ResamplingSpCEnv

Active bindings

`iters` integer(1)

Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods

Public methods:

- [ResamplingSpCEnv\\$new\(\)](#)
- [ResamplingSpCEnv\\$instantiate\(\)](#)
- [ResamplingSpCEnv\\$clone\(\)](#)

Method `new()`: Create an "Environmental Block" resampling instance. For a list of available arguments, please see [blockCV::envBlock](#).

Usage:

```
ResamplingSpCEnv$new(id = "spcv_env")
```

Arguments:

`id` character(1)
Identifier for the resampling strategy.

Method `instantiate()`: Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingSpCEnv$instantiate(task)
```

Arguments:

`task` [Task](#)
A task to instantiate.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Valavi R, Elith J, Lahoz-Monfort JJ, Guillera-Aroita G (2018). "blockCV: an R package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models." *bioRxiv*. doi: [10.1101/357798](https://doi.org/10.1101/357798).

Examples

```
if (mLr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mLr3)
  task = tsk("ecuador")

  # Instantiate Resampling
  rcv = rsmpl("spcv_env", folds = 4)
  rcv$instantiate(task)

  # Individual sets:
  rcv$train_set(1)
  rcv$test_set(1)
}
```

```

intersect(rcv$train_set(1), rcv$test_set(1))

# Internal storage:
rcv$instance
}

```

ResamplingSpCVTiles *(sperrorest) Spatial "Tiles" resampling*

Description

partition_tiles divides the study area into a specified number of rectangular tiles. Optionally small partitions can be merged with adjacent tiles to achieve a minimum number or percentage of samples in each tile.

mlr3spatiotempcv notes

The 'Description' and 'Note' fields are inherited from the respective upstream function.

For a list of available arguments, please see [sperrorest::partition_tiles](#).

This method is similar to [ResamplingSpCVBlock](#).

Super class

[mlr3::Resampling](#) -> ResamplingSpCVTiles

Active bindings

iters integer(1)
Returns the number of resampling iterations, depending on the values stored in the param_set.

Methods

Public methods:

- [ResamplingSpCVTiles\\$new\(\)](#)
- [ResamplingSpCVTiles\\$instantiate\(\)](#)
- [ResamplingSpCVTiles\\$clone\(\)](#)

Method new(): Create a "Spatial 'Tiles' resampling" resampling instance.

Usage:

```
ResamplingSpCVTiles$new(id = "spcv_Tiles")
```

Arguments:

id character(1)

Identifier for the resampling strategy. For a list of available arguments, please see [sperrorest::partition_tiles](#).

Method instantiate(): Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingSpCVTiles$instantiate(task)
```

Arguments:

```
task Task
```

A task to instantiate.

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

Usage:

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

Arguments:

```
deep Whether to make a deep clone.
```

Note

Default parameter settings may change in future releases. This function, especially the rotation and shifting part of it and the algorithm for cleaning up small tiles is still a bit experimental. Use with caution. For non-zero offsets (`offset!=none`), the number of tiles may actually be greater than `nsplit[1]*nsplit[2]` because of fractional tiles lurking into the study region. `reassign=TRUE` with suitable thresholds is therefore recommended for non-zero (including random) offsets.

References

Brenning A (2012). "Spatial cross-validation and bootstrap for the assessment of prediction rules in remote sensing: The R package `sperrorest`." In *2012 IEEE International Geoscience and Remote Sensing Symposium*. doi: [10.1109/igarss.2012.6352393](https://doi.org/10.1109/igarss.2012.6352393).

See Also

`ResamplingSpCVBlock`

Examples

```
if (mlr3misc::require_namespaces("sperrorest", quietly = TRUE)) {
  library(mlr3)
  task = tsk("ecuador")

  # Instantiate Resampling
  rcv = rsmpl("spcv_tiles", nsplit = c(4L, 3L), reassign = FALSE)
  rcv$instantiate(task)

  # Individual sets:
  rcv$train_set(1)
  rcv$test_set(1)
  # check that no obs are in both sets
  intersect(rcv$train_set(1), rcv$test_set(1)) # good!

  # Internal storage:
  rcv$instance # table
}
```

ResamplingSptCVCluto (*skmeans*) *Spatiotemporal clustering resampling*

Description

Spatiotemporal cluster partitioning via the `vcluster` executable of the CLUTO clustering application.

This partitioning method relies on the external CLUTO library. To use it, CLUTO's executables need to be downloaded and installed into this package.

See <https://gist.github.com/pat-s/6430470cf817050e27d26c43c0e9be72> for an installation approach that should work on Windows and Linux. macOS is not supported by CLUTO.

Before using this method, please check the restrictive copyright shown below.

Details

By default, `-clmethod='direct'` is passed to the `vcluster` executable in contrast to the upstream default `-clmethod='rb'`. There is no evidence or research that this method is the best among the available ones ("rb", "rbr", "direct", "agglo", "graph", "bagglo"). Also, various other parameters can be set via argument `cluto_parameters` to achieve different clustering results.

Parameter `-clusterfile` is handled by **skmeans** and cannot be changed.

Copyright

CLUTO's copyright is as follows:

The CLUTO package is copyrighted by the Regents of the University of Minnesota. It can be freely used for educational and research purposes by non-profit institutions and US government agencies only. Other organizations are allowed to use CLUTO only for evaluation purposes, and any further uses will require prior approval. The software may not be sold or redistributed without prior approval. One may make copies of the software for their use provided that the copies, are not sold or distributed, are used under the same terms and conditions. As unestablished research software, this code is provided on an "as is" basis without warranty of any kind, either expressed or implied. The downloading, or executing any part of this software constitutes an implicit agreement to these terms. These terms and conditions are subject to change at any time without prior notice.

Super class

```
mlr3::Resampling -> ResamplingSptCVCluto
```

Public fields

`time_var` **character**

The name of the variable which represents the time dimension. Must be of type numeric.

`clmethod` **character**

Name of the clustering method to use within `vcluster`. See Details for more information.

- `cluto_parameters` **character**
 Additional parameters to pass to `vcluster`. Must be given as a single character string, e.g. `"param1='value1'param2='value2'"`. See the CLUTO documentation for a full list of supported parameters.
- `verbose` **logical**
 Whether to show `vcluster` progress and summary output.

Active bindings

- `iters` **integer(1)**
 Returns the number of resampling iterations, depending on the values stored in the `param_set`.

Methods

Public methods:

- `ResamplingSptCVCluto$new()`
- `ResamplingSptCVCluto$instantiate()`
- `ResamplingSptCVCluto$clone()`

Method `new()`: Create an repeated resampling instance using the CLUTO algorithm.

Usage:

```
ResamplingSptCVCluto$new(
  id = "sptcv_cluto",
  time_var = NULL,
  clmethod = "direct",
  cluto_parameters = NULL,
  verbose = TRUE
)
```

Arguments:

- `id` **character(1)**
 Identifier for the resampling strategy.
- `time_var` **character**
 The name of the variable which represents the time dimension. Must be of type numeric.
- `clmethod` **character**
 Name of the clustering method to use within `vcluster`. See Details for more information.
- `cluto_parameters` **character**
 Additional parameters to pass to `vcluster`. Must be given as a single character string, e.g. `"param1='value1'param2='value2'"`. See the CLUTO documentation for a full list of supported parameters.
- `verbose` **logical**
 Whether to show `vcluster` progress and summary output.

Method `instantiate()`: Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingSptCVCluto$instantiate(task)
```

Arguments:

task [Task](#)

A task to instantiate.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Zhao Y, Karypis G (2002). "Evaluation of Hierarchical Clustering Algorithms for Document Datasets." *11th Conference of Information and Knowledge Management (CIKM)*, 51-524. <http://glaros.dtc.umn.edu/gkhome/node/167>.

Examples

```
## Not run:
if (mlr3misc::require_namespaces("skmeans", quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("cookfarm")

  # Instantiate Resampling
  rcv = rsmpl("sptcv_cluto", folds = 5, time_var = "Date")
  rcv$instantiate(task)

  # Individual sets:
  rcv$train_set(1)
  rcv$test_set(1)
  # check that no obs are in both sets
  intersect(rcv$train_set(1), rcv$test_set(1)) # good!

  # Internal storage:
  rcv$instance # table
}

## End(Not run)
```

ResamplingSptCVCstf (CAST) "Leave-location-and-time-out" resampling

Description

Create spatial, temporal or spatio-temporal Folds for cross validation

Details

Using "class" is helpful in the case that data are clustered in space and are categorical. E.g This is the case for land cover classifications when training data come as training polygons. In this case the data should be split in a way that entire polygons are held back (spacevar="polygonID") but at the same time the distribution of classes should be similar in each fold (class="LUC").

mlr3spatiotempcv notes

The 'Description', 'Details' and 'Note' fields are inherited from the respective upstream function. For a list of available arguments, please see [CAST::CreateSpacetimeFolds](#).

Super class

[mlr3::Resampling](#) -> ResamplingSptCVCstf

Active bindings

iters integer(1)

Returns the number of resampling iterations, depending on the values stored in the param_set.

Methods

Public methods:

- [ResamplingSptCVCstf\\$new\(\)](#)
- [ResamplingSptCVCstf\\$instantiate\(\)](#)
- [ResamplingSptCVCstf\\$clone\(\)](#)

Method new(): Create a "Spacetime Folds" resampling instance.

For a list of available arguments, please see [CAST::CreateSpacetimeFolds](#).

Usage:

```
ResamplingSptCVCstf$new(id = "sptcv_cstf")
```

Arguments:

id character(1)

Identifier for the resampling strategy.

Method instantiate(): Materializes fixed training and test splits for a given task.

Usage:

```
ResamplingSptCVCstf$instantiate(task)
```

Arguments:

task [Task](#)

A task to instantiate.

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Note

Standard k-fold cross-validation can lead to considerable misinterpretation in spatial-temporal modelling tasks. This function can be used to prepare a Leave-Location-Out, Leave-Time-Out or Leave-Location-and-Time-Out cross-validation as target-oriented validation strategies for spatial-temporal prediction tasks. See Meyer et al. (2018) for further information.

References

Meyer H, Reudenbach C, Hengl T, Katurji M, Nauss T (2018). “Improving performance of spatio-temporal machine learning models using forward feature selection and target-oriented validation.” *Environmental Modelling & Software*, **101**, 1–9. doi: [10.1016/j.envsoft.2017.12.001](https://doi.org/10.1016/j.envsoft.2017.12.001).

Examples

```
library(mlr3)
task = tsk("cookfarm")

# Instantiate Resampling
rcv = rsm("sptcv_cstf",
  folds = 5,
  time_var = "Date", space_var = "SOURCEID")
rcv$instantiate(task)

# Individual sets:
rcv$train_set(1)
rcv$test_set(1)
# check that no obs are in both sets
intersect(rcv$train_set(1), rcv$test_set(1)) # good!

# Internal storage:
rcv$instance # table
```

TaskClassifST

Crate a Spatiotemporal Classification Task

Description

This task specializes [Task](#) and [TaskSupervised](#) for spatiotemporal classification problems. The target column is assumed to be a factor. The `task_type` is set to "classif" and "spatiotemporal". A spatial example task is available via `tsk("ecuador")`, a spatiotemporal one via `tsk("cookfarm")`.

The coordinate reference system passed during initialization must match the one which was used during data creation, otherwise offsets of multiple meters may occur. By default, coordinates are not used as features. This can be changed by setting `extra_args$coords_as_features = TRUE`.

Super classes

```
mlr3::Task -> mlr3::TaskSupervised -> mlr3::TaskClassif -> TaskClassifST
```

Public fields

`extra_args` (named list())
 Additional task arguments set during construction. Required for `convert_task()`.

Methods**Public methods:**

- `TaskClassifST$new()`
- `TaskClassifST$coordinates()`
- `TaskClassifST$print()`
- `TaskClassifST$clone()`

Method `new()`: Create a new spatiotemporal resampling Task

Usage:

```
TaskClassifST$new(
  id,
  backend,
  target,
  positive = NULL,
  extra_args = list(coords_as_features = FALSE, crs = NA, coordinate_names = NA)
)
```

Arguments:

`id` [character(1)]

Identifier for the task.

`backend` [DataBackend](#)

Either a [DataBackend](#), or any object which is convertible to a [DataBackend](#) with `as_data_backend()`.
 E.g., a `data.frame()` will be converted to a [DataBackendDataTable](#).

`target` [character(1)]

Name of the target column.

`positive` [character(1)]

Only for binary classification: Name of the positive class. The levels of the target columns are reordered accordingly, so that the first element of `$class_names` is the positive class, and the second element is the negative class.

`extra_args` [named list]

Additional task arguments set during construction. Required for `convert_task()`.

- `crs` [character(1)]
 Coordinate reference system. Either a PROJ string or an [EPSG](#) code.
- `coords_as_features` [logical(1)]
 Whether the coordinates should also be used as features.
- `coordinate_names` [character(2)]
 The variables names of the coordinates in the data.

Method `coordinates()`: Return the coordinates of the task

Usage:

```
TaskClassifST$coordinates(rows = NULL)
```

Arguments:

rows Row IDs. Can be used to subset the returned coordinates.

Method print(): Print the task.

Usage:

```
TaskClassifST$print(...)
```

Arguments:

... Arguments passed to the \$print() method of the superclass.

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other Task: [TaskRegrST](#), [mlr_tasks_cookfarm](#), [mlr_tasks_diplodia](#), [mlr_tasks_ecuador](#)

Examples

```
if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {

  data = mlr3::as_data_backend(ecuador)
  task = TaskClassifST$new("ecuador",
    backend = data, target = "slides",
    positive = "TRUE", extra_args = list(coordinate_names = c("x", "y")))
  )

  # passing objects of class 'sf' is also supported
  data_sf = sf::st_as_sf(ecuador, coords = c("x", "y"))
  task = TaskClassifST$new("ecuador_sf",
    backend = data_sf, target = "slides", positive = "TRUE"
  )

  task$task_type
  task$formula()
  task$class_names
  task$positive
  task$negative
  task$coordinates()
  task$coordinate_names
}
```

TaskRegrST

*Create a Spatiotemporal Regression Task***Description**

This task specializes [Task](#) and [TaskSupervised](#) for spatiotemporal classification problems.

A spatial example task is available via `tsk("ecuador")`, a spatiotemporal one via `tsk("cookfarm")`.

The coordinate reference system passed during initialization must match the one which was used during data creation, otherwise offsets of multiple meters may occur. By default, coordinates are not used as features. This can be changed by setting `extra_args$coords_as_features = TRUE`.

Super classes

```
mlr3::Task -> mlr3::TaskSupervised -> mlr3::TaskRegr -> TaskRegrST
```

Public fields

`extra_args` (named list())

Additional task arguments set during construction. Required for [convert_task\(\)](#).

Methods**Public methods:**

- [TaskRegrST\\$new\(\)](#)
- [TaskRegrST\\$coordinates\(\)](#)
- [TaskRegrST\\$print\(\)](#)
- [TaskRegrST\\$clone\(\)](#)

Method `new()`: Create a new spatiotemporal resampling Task

Usage:

```
TaskRegrST$new(
  id,
  backend,
  target,
  extra_args = list(coords_as_features = FALSE, crs = NA, coordinate_names = NA)
)
```

Arguments:

`id` [character(1)]

Identifier for the task.

`backend` [DataBackend](#)

Either a [DataBackend](#), or any object which is convertible to a [DataBackend](#) with `as_data_backend()`.

E.g., a `data.frame()` will be converted to a [DataBackendDataTable](#).

`target` [character(1)]

Name of the target column.

`extra_args` [named list]

Additional task arguments set during construction. Required for `convert_task()`.

- `crs` [character(1)]
Coordinate reference system. Either a PROJ string or an EPSG code.
- `coords_as_features` [logical(1)]
Whether the coordinates should also be used as features.
- `coordinate_names` [character(2)]
The variables names of the coordinates in the data.

Method `coordinates()`: Return the coordinates of the task

Usage:

```
TaskRegrST$coordinates(rows = NULL)
```

Arguments:

`rows` Row IDs. Can be used to subset the returned coordinates.

Method `print()`: Print the task.

Usage:

```
TaskRegrST$print(...)
```

Arguments:

... Arguments passed to the `$print()` method of the superclass.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Task: [TaskClassifST](#), [mlr_tasks_cookfarm](#), [mlr_tasks_diplodia](#), [mlr_tasks_ecuador](#)

Index

* Task

- mlr_tasks_cookfarm, 31
- mlr_tasks_diplodia, 32
- mlr_tasks_ecuador, 33
- TaskClassifST, 68
- TaskRegrST, 71

* datasets

- mlr_tasks_cookfarm, 31
- mlr_tasks_diplodia, 32
- mlr_tasks_ecuador, 33

- as_task_classif, 4
- as_task_classif_st, 5
- as_task_regr, 7
- as_task_regr_st, 8
- autoplot.ResamplingCustomCV, 10
- autoplot.ResamplingCV, 11
- autoplot.ResamplingCV(), 11, 15, 17, 19, 21, 23, 25, 28, 31
- autoplot.ResamplingRepeatedCV (autoplot.ResamplingCV), 11
- autoplot.ResamplingRepeatedSpCVBlock (autoplot.ResamplingSpCVBlock), 13
- autoplot.ResamplingRepeatedSpCVCoords (autoplot.ResamplingSpCVCoords), 17
- autoplot.ResamplingRepeatedSpCVDisc (autoplot.ResamplingSpCVDisc), 19
- autoplot.ResamplingRepeatedSpCVEnv (autoplot.ResamplingSpCVEnv), 22
- autoplot.ResamplingRepeatedSpCVTiles (autoplot.ResamplingSpCVTiles), 24
- autoplot.ResamplingRepeatedSptCVCluto (autoplot.ResamplingSptCVCluto), 26

- autoplot.ResamplingRepeatedSptCVCstf (autoplot.ResamplingSptCVCstf), 28
- autoplot.ResamplingSpCVBlock, 13
- autoplot.ResamplingSpCVBlock(), 11, 13, 17, 19, 21, 23, 25, 28, 31
- autoplot.ResamplingSpCVBuffer, 16
- autoplot.ResamplingSpCVBuffer(), 11, 13, 15, 19, 21, 23, 25, 28, 31
- autoplot.ResamplingSpCVCoords, 17
- autoplot.ResamplingSpCVCoords(), 11, 13, 15, 17, 21, 23, 25, 28, 31
- autoplot.ResamplingSpCVDisc, 19
- autoplot.ResamplingSpCVDisc(), 11, 13, 15, 19, 23, 25, 28
- autoplot.ResamplingSpCEnv, 22
- autoplot.ResamplingSpCEnv(), 11, 13, 15, 17, 19, 21, 25, 28, 31
- autoplot.ResamplingSpCVTiles, 24
- autoplot.ResamplingSpCVTiles(), 11, 13, 15, 19, 21, 23, 28
- autoplot.ResamplingSptCVCluto, 26
- autoplot.ResamplingSptCVCluto(), 11, 13, 15, 17, 19, 21, 23, 25, 31
- autoplot.ResamplingSptCVCstf, 28
- autoplot.ResamplingSptCVCstf(), 11, 13, 15, 17, 19, 23, 28
- blockCV::buffering, 55
- blockCV::buffering(), 55
- blockCV::envBlock, 41, 42, 60, 61
- blockCV::spatialBlock, 35, 52
- blockCV::spatialBlock(), 35, 52, 53
- buffering, 34, 52
- CAST::CreateSpacetimeFolds, 49, 50, 67
- character, 47, 48, 64, 65
- convert_task(), 69, 71, 72
- cookfarm_sample (mlr_tasks_cookfarm), 31
- data.frame(), 5, 8

- DataBackend, [5](#), [8](#), [69](#), [71](#)
- DataBackendDataTable, [69](#), [71](#)
- Dictionary, [32–34](#)
- diplodia (mlr_tasks_diplodia), [32](#)
- ecuador (mlr_tasks_ecuador), [33](#)
- ggplot(), [15](#)
- ggsci::scale_color_ucscgb(), [15](#)
- kmeans, [41](#), [60](#)
- logical, [47](#), [48](#), [65](#)
- mlr3::as_task_classif(), [5](#)
- mlr3::as_task_regr(), [7](#)
- mlr3::Resampling, [35](#), [37](#), [39](#), [42](#), [44](#), [46](#), [50](#),
[53](#), [55](#), [56](#), [58](#), [60](#), [62](#), [64](#), [67](#)
- mlr3::Task, [68](#), [71](#)
- mlr3::TaskClassif, [68](#)
- mlr3::TaskRegr, [71](#)
- mlr3::TaskSupervised, [68](#), [71](#)
- mlr3spatiotempcv
(mlr3spatiotempcv-package), [3](#)
- mlr3spatiotempcv-package, [3](#)
- mlr_tasks, [32–34](#)
- mlr_tasks_cookfarm, [31](#), [33](#), [34](#), [70](#), [72](#)
- mlr_tasks_diplodia, [32](#), [32](#), [34](#), [70](#), [72](#)
- mlr_tasks_ecuador, [32](#), [33](#), [33](#), [70](#), [72](#)
- plot.ResamplingCustomCV
(autoplot.ResamplingCustomCV),
[10](#)
- plot.ResamplingCV
(autoplot.ResamplingCV), [11](#)
- plot.ResamplingRepeatedCV
(autoplot.ResamplingCV), [11](#)
- plot.ResamplingRepeatedSpCVBlock
(autoplot.ResamplingSpCVBlock),
[13](#)
- plot.ResamplingRepeatedSpCVCoords
(autoplot.ResamplingSpCVCoords),
[17](#)
- plot.ResamplingRepeatedSpCVDisc
(autoplot.ResamplingSpCVDisc),
[19](#)
- plot.ResamplingRepeatedSpCEnv
(autoplot.ResamplingSpCEnv),
[22](#)
- plot.ResamplingRepeatedSpCVTiles
(autoplot.ResamplingSpCVTiles),
[24](#)
- plot.ResamplingRepeatedSptCVCluto
(autoplot.ResamplingSptCVCluto),
[26](#)
- plot.ResamplingRepeatedSptCVCstf
(autoplot.ResamplingSptCVCstf),
[28](#)
- plot.ResamplingSpCVBlock
(autoplot.ResamplingSpCVBlock),
[13](#)
- plot.ResamplingSpCVBuffer
(autoplot.ResamplingSpCVBuffer),
[16](#)
- plot.ResamplingSpCVCoords
(autoplot.ResamplingSpCVCoords),
[17](#)
- plot.ResamplingSpCVDisc
(autoplot.ResamplingSpCVDisc),
[19](#)
- plot.ResamplingSpCEnv
(autoplot.ResamplingSpCEnv),
[22](#)
- plot.ResamplingSpCVTiles
(autoplot.ResamplingSpCVTiles),
[24](#)
- plot.ResamplingSptCVCluto
(autoplot.ResamplingSptCVCluto),
[26](#)
- plot.ResamplingSptCVCstf
(autoplot.ResamplingSptCVCstf),
[28](#)
- R6::R6Class, [32](#), [33](#)
- represampling, [37](#), [56](#)
- ResamplingCustomCV, [10](#)
- ResamplingCV, [12](#)
- ResamplingRepeatedCV, [12](#)
- ResamplingRepeatedSpCVBlock, [14](#), [20](#), [25](#),
[34](#)
- ResamplingRepeatedSpCVCoords, [18](#), [19](#), [37](#)
- ResamplingRepeatedSpCVDisc, [39](#)
- ResamplingRepeatedSpCEnv, [23](#), [41](#)
- ResamplingRepeatedSpCVTiles, [43](#)
- ResamplingRepeatedSptCVCluto, [27](#), [28](#), [46](#)
- ResamplingRepeatedSptCVCstf, [30](#), [49](#)
- ResamplingSpCVBlock, [14](#), [15](#), [20](#), [21](#), [25](#), [43](#),
[52](#), [62](#)

ResamplingSpCVBuffer, [15–17](#), [21](#), [25](#), [54](#),
[58](#)
ResamplingSpCVCoords, [15](#), [18](#), [19](#), [21](#), [25](#), [56](#)
ResamplingSpCVDisc, [58](#)
ResamplingSpCEnv, [15](#), [21](#), [23](#), [25](#), [60](#)
ResamplingSpCVTiles, [62](#)
ResamplingSptCVCluto, [27](#), [28](#), [64](#)
ResamplingSptCVCstf, [29](#), [30](#), [66](#)

sf::sf, [5](#), [8](#)
spatialAutoRange, [54](#)
sperrorest::partition_cv, [37](#), [56](#), [57](#)
sperrorest::partition_disc, [39](#), [58](#)
sperrorest::partition_tiles, [43](#), [44](#), [62](#)

Task, [36](#), [38](#), [40](#), [43](#), [45](#), [48](#), [51](#), [53](#), [55](#), [57](#), [59](#),
[61](#), [63](#), [66–68](#), [71](#)
TaskClassif, [4–6](#), [33](#)
TaskClassifST, [4–6](#), [32–34](#), [68](#), [72](#)
TaskRegr, [7](#), [32](#)
TaskRegrST, [7–9](#), [32–34](#), [70](#), [71](#)
Tasks, [32–34](#)
TaskSupervised, [68](#), [71](#)