Package 'mlr3spatiotempcv'

<p style="text-align:center">August 9, 2022</p>

Title Spatiotemporal Resampling Methods for 'mlr3'

Version 2.0.2

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 mlr3 (>= 0.12.0), R (>= 3.5.0)

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

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

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.2.1

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 2022-08-09 19:00:02 UTC

R **topics documented:**

```
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/special.html#spatiotemporal
- 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, 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.

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.

Meyer H, Reudenbach C, Hengl T, Katurji M, Nauss T (2018). "Improving performance of spatiotemporal 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.

Zhao Y, Karypis G (2002). "Evaluation of Hierarchical Clustering Algorithms for Document Datasets." *11th Conference of Information and Knowledge Management (CIKM)*, 51-524. doi:10.1145/584792.584877.

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_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: Extracts spatial meta data before construction.
4. TaskRegr: Calls convert_task().

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,
  id = deparse(substitute(x)),
  positive = NULL,
  coordinate_names,
  crs = NA_character_,
  coords_as_features = FALSE,
  label = NA_character_,
  ...
)

## S3 method for class 'DataBackend'
as_task_classif_st(
  x,
  target,
  id = deparse(substitute(x)),
  positive = NULL,
  coordinate_names,
  crs,
  coords_as_features = FALSE,
  label = NA_character_,
  ...
)

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

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 the data argument.

positive   (character(1))
                  Only for binary classification: Name of the positive class. The levels of the tar-
                  get columns are reordered accordingly, so that the first element of $class_names
                  is the positive class, and the second element is the negative class.

coordinate_names
                  (character(1))
                  The column names of the coordinates in the data.

crs   (character(1))
                  Coordinate reference system. WKT2 or EPSG string.

coords_as_features
                  (logical(1))
                  If TRUE, coordinates are used as features. This is a shortcut for task$set_col_roles(c("x",
                  "y"), role = "feature") with the assumption that the coordinates in the data
                  are named "x" and "y".

label   (character(1))
                  Label for the new instance. Shown in as.data.table(mlr_tasks).

Value

[TaskClassifST](#).

Examples

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

---

as_task_regr_st.TaskClassifST
                              Convert to a Spatiotemporal Regression Task

---

Description

Convert object to a [TaskRegrST](#).

This is a S3 generic, specialized for at least the following objects:

1. TaskRegrST: Ensure the identity.

2. data.frame() and DataBackend: Provides an alternative to the constructor of TaskRegrST.

3. sf::sf: Extracts spatial meta data before construction.

4. TaskClassif: Calls convert_task().

Usage

```
## S3 method for class 'TaskClassifST'
as_task_regr_st(
  x,
  target = NULL,
  drop_original_target = FALSE,
  drop_levels = TRUE,
  ...
)

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,
  id = deparse(substitute(x)),
  coordinate_names,
  crs = NA_character_,
  coords_as_features = FALSE,
  label = NA_character_,
  ...
)

## S3 method for class 'DataBackend'
as_task_regr_st(
  x,
  target,
  id = deparse(substitute(x)),
  positive = NULL,
  coordinate_names,
  crs,
  coords_as_features = FALSE,
  label = NA_character_,
  ...
)

## S3 method for class 'sf'
as_task_regr_st(
```

```
  x,
  target = NULL,
  id = deparse(substitute(x)),
  coords_as_features = FALSE,
  label = NA_character_,
  ...
)

## S3 method for class 'TaskClassifST'
as_task_regr_st(
  x,
  target = NULL,
  drop_original_target = FALSE,
  drop_levels = TRUE,
  ...
)
```

Arguments

x	(any) Object to convert.
target	(character(1)) Name of the target column.
drop_original_target	
	(logical(1)) If FALSE (default), the original target is added as a feature. Otherwise the original target is dropped.
drop_levels	(logical(1)) If TRUE (default), unused levels of the new target variable are dropped.
...	(any) Additional arguments.
clone	(logical(1)) If TRUE, ensures that the returned object is not the same as the input x.
id	(character(1)) Id for the new task. Defaults to the (deparsed and substituted) name of the data argument.
coordinate_names	
	(character(1)) The column names of the coordinates in the data.
crs	(character(1)) Coordinate reference system. WKT2 or EPSG string.
coords_as_features	
	(logical(1)) If TRUE, coordinates are used as features. This is a shortcut for task$set_col_roles(c("x", "y"), role = "feature") with the assumption that the coordinates in the data are named "x" and "y".

label	(character(1))
	Label for the new instance. Shown in `as.data.table(mlr_tasks)`.
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.

Value

[TaskRegrST](#)

Examples

```
if (mlr3misc::require_namespaces(c("sf"), quietly = TRUE)) {
  library("mlr3")
  data("cookfarm_mlr3", package = "mlr3spatiotempcv")

  # data.frame
  as_task_regr_st(cookfarm_mlr3, target = "PHIHOX",
    coords_as_features = FALSE, crs = 26911,
    coordinate_names = c("x", "y"))

  # sf
  cookfarm_sf = sf::st_as_sf(cookfarm_mlr3, coords = c("x", "y"), crs = 26911)
  as_task_regr_st(cookfarm_sf, target = "PHIHOX")
}
```

---

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",
  sample_fold_n = NULL,
  ...
)
```

```
## 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.
sample_fold_n	[integer] Number of points in a random sample stratified over partitions. This argument aims to keep file sizes of resulting plots reasonable and reduce overplotting in dense datasets.
...	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 "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 = rsmp("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",
  sample_fold_n = NULL,
  ...
)

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

```
## 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.
sample_fold_n	[integer] Number of points in a random sample stratified over partitions. This argument aims to keep file sizes of resulting plots reasonable and reduce overplotting in dense datasets.
...	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 "Spatiotemporal Visualization"
- autoplot.ResamplingSpCVBlock()
- autoplot.ResamplingSpCVBuffer()
- autoplot.ResamplingSpCVCoords()
- autoplot.ResamplingSpCVEnv()
- autoplot.ResamplingSpCVDisc()
- autoplot.ResamplingSpCVTiles()
- autoplot.ResamplingSptCVCstf()
- autoplot.ResamplingSptCVCluto()

Examples

```
if (mlr3misc::require_namespaces(c("sf", "patchwork", "ggtext", "ggsci"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmp("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,
  sample_fold_n = NULL,
  ...
)

## 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,
    sample_fold_n = NULL,
    ...
)

## 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.
sample_fold_n	[integer] Number of points in a random sample stratified over partitions. This argument aims to keep file sizes of resulting plots reasonable and reduce overplotting in dense datasets.
...	Passed to geom_sf(). 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 "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 = rsmp("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

object	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVBuffer.
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 ResamplingSpCVBuffer.

See Also

- mlr3book chapter on "Spatiotemporal Visualization"
- autoplot.ResamplingSpCVBlock()
- autoplot.ResamplingSpCVCoords()
- autoplot.ResamplingSpCVEnv()
- autoplot.ResamplingCV()
- autoplot.ResamplingSptCVCstf()
- autoplot.ResamplingSptCVCluto()

Examples

```
if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmp("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",
    sample_fold_n = NULL,
    ...
)

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

## 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.
sample_fold_n	[integer] Number of points in a random sample stratified over partitions. This argument aims to keep file sizes of resulting plots reasonable and reduce overplotting in dense datasets.
...	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 Resamplin-
                         gRepeatedSpCVCoords.

See Also

- mlr3book chapter on "Spatiotemporal Visualization"

- autoplot.ResamplingSpCVBlock()

- autoplot.ResamplingSpCVBuffer()

- autoplot.ResamplingSpCVEnv()

- 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 = rsmp("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,
  sample_fold_n = NULL,
  ...
)

## 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,
  sample_fold_n = NULL,
  ...
)

## 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.
sample_fold_n	[integer] Number of points in a random sample stratified over partitions. This argument aims to keep file sizes of resulting plots reasonable and reduce overplotting in dense datasets.
...	Passed to geom_sf(). Helpful for adjusting point sizes and shapes.
x	[Resampling] mlr3 spatial resampling object. One of class ResamplingSpCVBuffer, ResamplingSpCVBlock, ResamplingSpCVCoords, ResamplingSpCVEnv.

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 "Spatiotemporal Visualization"
- Vignette Spatiotemporal Visualization.
- autoplot.ResamplingSpCVBlock()
- autoplot.ResamplingSpCVBuffer()
- autoplot.ResamplingSpCVCoords()
- autoplot.ResamplingSpCVTiles()
- autoplot.ResamplingSpCVEnv()
- autoplot.ResamplingCV()
- autoplot.ResamplingSptCVCluto()

Examples

```
if (mlr3misc::require_namespaces("sf", quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmp("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.ResamplingSpCVEnv

Visualization Functions for SpCV Env Methods.

Description

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

Usage

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

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

```
    sample_fold_n = NULL,
    ...
)

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

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

Arguments

object
: [Resampling]
mlr3 spatial resampling object of class [ResamplingSpCVEnv](#) or [ResamplingRepeatedSpCVEnv](#).

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.

sample_fold_n
: [integer]
Number of points in a random sample stratified over partitions. This argument aims to keep file sizes of resulting plots reasonable and reduce overplotting in dense datasets.

...
: 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 [ResamplingSpCVEnv](#) or [ResamplingRepeatedSpCVEnv](#).

See Also

- mlr3book chapter 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 = rsmp("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,
  sample_fold_n = NULL,
  ...
)

## 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,
    sample_fold_n = NULL,
    ...
)

## 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.
sample_fold_n	[integer] Number of points in a random sample stratified over partitions. This argument aims to keep file sizes of resulting plots reasonable and reduce overplotting in dense datasets.
...	Passed to geom_sf(). 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 "Spatiotemporal Visualization"
- Vignette Spatiotemporal Visualization.
- autoplot.ResamplingSpCVBlock()
- autoplot.ResamplingSpCVBuffer()
- autoplot.ResamplingSpCVCoords()
- autoplot.ResamplingSpCVDisc()
- autoplot.ResamplingSpCVEnv()
- autoplot.ResamplingCV()
- autoplot.ResamplingSptCVCluto()

Examples

```
if (mlr3misc::require_namespaces(c("sf", "sperrorest"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmp("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,
  sample_fold_n = NULL,
  ...
)

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

## 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.

sample_fold_n   [integer]
                 Number of points in a random sample stratified over partitions. This argument aims to keep file sizes of resulting plots reasonable and reduce overplotting in dense datasets.

...   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 ResamplingSptCVCluto or Resamplin-gRepeatedSptCVCluto.

See Also

- mlr3book chapter on "Spatiotemporal Visualization"

- Vignette Spatiotemporal Visualization.

- autoplot.ResamplingSpCVBlock()

- autoplot.ResamplingSpCVBuffer()

- autoplot.ResamplingSpCVCoords()

- autoplot.ResamplingSpCVEnv()

- 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_mlr3")
  resampling = rsmp("sptcv_cluto", folds = 5)
  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",
  repeats_id = NULL,
  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,
  plot_time_var = NULL,
  sample_fold_n = 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,
  plot_time_var = 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 ResamplingRepeatedSptCVCstf.
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.
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 `plotly::orca()`. This requires the `orca` utility to be available. See https://github.com/plotly/orca for more information. When used, by default a file named `plot.png` 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**.
plot_time_var	`[character]` The variable to use for the z-axis (time). Remove the column role `feature` for this variable to only use it for plotting.
sample_fold_n	`[integer]` Number of points in a random sample stratified over partitions. This argument aims to keep file sizes of resulting plots reasonable and reduce overplotting in dense datasets.
...	Passed down to `plotly::orca()`. Only effective when `static_image = TRUE`.
x	`[Resampling]` mlr3 spatial resampling object of class [ResamplingSptCVCstf](#) or [ResamplingRepeatedSptCVCstf](#).

Details

This method requires to set argument `fold_id`. No plot showing all folds in one plot can be created. This is because the LLTO method does not make use of all observations but only a subset of them (many observations are omitted). 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 be misleading.

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 "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_mlr3")
  task_st$set_col_roles("SOURCEID", "space")
  task_st$set_col_roles("Date", "time")
  resampling = rsmp("sptcv_cstf", folds = 5)
  resampling$instantiate(task_st)

  # with both `"space"` and `"time"` column roles set (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_resamplings_repeated_spcv_block
                              (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 deviding `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.

Parameters

- `repeats (integer(1))`
  Number of repeats.

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.

Examples

```
## Not run:
if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mlr3)
  task = tsk("diplodia")

  # Instantiate Resampling
  rrcv = rsmp("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
}

## End(Not run)
```

---

mlr_resamplings_repeated_spcv_coords

(sperrorest) Repeated coordinate-based k-means clustering

Description

Splits data by clustering in the coordinate space. See the upstream implementation at `sperrorest::partition_kmeans()` and Brenning (2012) for further information.

Details

Universal partitioning method that splits the data in the coordinate space. Useful for spatially homogeneous datasets that cannot be split well with rectangular approaches like `ResamplingSpCVBlock`.

Parameters

- `folds` `(integer(1))`
  Number of folds.

- `repeats` `(integer(1))`
  Number of repeats.

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.

Examples

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

# Instantiate Resampling
rrcv = rsmp("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
```

mlr_resamplings_repeated_spcv_disc
                          (sperrorest) Repeated spatial "disc" resampling

Description

(sperrorest) Repeated spatial "disc" resampling

(sperrorest) Repeated spatial "disc" resampling

Parameters

- repeats (integer(1))
  Number of repeats.

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.

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
```

---

mlr_resamplings_repeated_spcv_env
(blockCV) Repeated "environmental blocking" resampling

Description

Splits data by clustering in the feature space. See the upstream implementation at blockCV::envBlock()
and Valavi et al. (2018) for further information.

Details

Useful when the dataset is supposed to be split on environmental information which is present in
features. The method allows for a combination of multiple features for clustering.

The input of raster images directly as in blockCV::envBlock() is not supported. See **mlr3spatial**
and its raster DataBackends for such support in **mlr3**.

Parameters

- folds (integer(1))
  Number of folds.

- features (character())
  The features to use for clustering.

- repeats (integer(1))
  Number of repeats.

Super class

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

Active bindings

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

Methods

Public methods:

- [ResamplingRepeatedSpCVEnv$new()](#)
- [ResamplingRepeatedSpCVEnv$folds()](#)
- [ResamplingRepeatedSpCVEnv$repeats()](#)
- [ResamplingRepeatedSpCVEnv$instantiate()](#)

- ResamplingRepeatedSpCVEnv$clone()

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

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

Usage:

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

Arguments:

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

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

Usage:

```
ResamplingRepeatedSpCVEnv$folds(iters)
```

Arguments:

`iters` `integer()`
    Iteration number.

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

Usage:

```
ResamplingRepeatedSpCVEnv$repeats(iters)
```

Arguments:

`iters` `integer()`
    Iteration number.

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

Usage:

```
ResamplingRepeatedSpCVEnv$instantiate(task)
```

Arguments:

`task` Task
    A task to instantiate.

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

Usage:

```
ResamplingRepeatedSpCVEnv$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](doi: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
}
```

mlr_resamplings_repeated_spcv_tiles

(sperrorest) Repeated spatial "tiles" resampling

Description

Spatial partitioning using rectangular tiles. Small partitions can optionally be merged into adjacent ones to avoid partitions with too few observations. This method is similar to `ResamplingSpCVBlock` by making use of rectangular zones in the coordinate space. See the upstream implementation at `sperrorest::partition_disc()` and Brenning (2012) for further information.

Parameters

- dsplit (integer(2))
  Equidistance of splits in (possibly rotated) x direction (`dsplit[1]`) and y direction (`dsplit[2]`) used to define tiles. If dsplit is of length 1, its value is recycled. Either dsplit or nsplit must be specified.

- nsplit (integer(2))
  Number of splits in (possibly rotated) x direction (`nsplit[1]`) and y direction (`nsplit[2]`) used to define tiles. If nsplit is of length 1, its value is recycled.

- rotation (character(1))
  Whether and how the rectangular grid should be rotated; random rotation is only possible between -45 and +45 degrees. Accepted values: One of `c("none", "random", "user")`.

- user_rotation (character(1))
  Only used when `rotation = "user"`. Angle(s) (in degrees) by which the rectangular grid is to be rotated in each repetition. Either a vector of same length as repeats, or a single number that will be replicated `length(repeats)` times.

- `offset` `(logical(1))`
  Whether and how the rectangular grid should be shifted by an offset. Accepted values: One of `c("none", "random", "user")`.

- `user_offset` `(logical(1))`
  Only used when `offset = "user"`. A list (or vector) of two components specifying a shift of the rectangular grid in (possibly rotated) x and y direction. The offset values are relative values, a value of 0.5 resulting in a one-half tile shift towards the left, or upward. If this is a list, its first (second) component refers to the rotated x (y) direction, and both components must have same length as `repeats` (or length 1). If a vector of length 2 (or list components have length 1), the two values will be interpreted as relative shifts in (rotated) x and y direction, respectively, and will therefore be recycled as needed (`length(repeats)` times each).

- `reassign` `(logical(1))`
  If `TRUE`, 'small' tiles (as per `min_frac` and `min_n`) are merged with (smallest) adjacent tiles. If `FALSE`, small tiles are 'eliminated', i.e., set to `NA`.

- `min_frac` `(numeric(1))`
  Value must be >=0, <1. Minimum relative size of partition as percentage of sample.

- `min_n` `(integer(1))`
  Minimum number of samples per partition.

- `iterate` `(integer(1))`
  Passed down to [`sperrorest::tile_neighbors()`](sperrorest::tile_neighbors()).

- `repeats` `(integer(1))`
  Number of repeats.

Super class

[`mlr3::Resampling`](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$new()`](ResamplingRepeatedSpCVTiles$new())
- [`ResamplingRepeatedSpCVTiles$folds()`](ResamplingRepeatedSpCVTiles$folds())
- [`ResamplingRepeatedSpCVTiles$repeats()`](ResamplingRepeatedSpCVTiles$repeats())
- [`ResamplingRepeatedSpCVTiles$instantiate()`](ResamplingRepeatedSpCVTiles$instantiate())
- [`ResamplingRepeatedSpCVTiles$clone()`](ResamplingRepeatedSpCVTiles$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.

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.

See Also

ResamplingSpCVBlock

Examples

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

  # Instantiate Resampling
  rrcv = rsmp("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
}
```

---

mlr_resamplings_repeated_sptcv_cluto

(skmeans) Repeated spatiotemporal clustering resampling via CLUTO

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

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",
  clmethod = "direct",
  cluto_parameters = NULL,
```

```
    verbose = TRUE
)
```

Arguments:

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

`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.

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. doi:10.1145/584792.584877.

Examples

```
## Not run:
if (mlr3misc::require_namespaces("skmeans", quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("cookfarm_mlr3")
  task$set_col_roles("Date", "time")

  # Instantiate Resampling
  rrcv = rsmp("repeated_sptcv_cluto", 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
}

## End(Not run)
```

---

mlr_resamplings_repeated_sptcv_cstf

>   *(CAST) Repeated spatiotemporal "leave-location-and-time-out" re-*
>   *sampling*

Description

Splits data using Leave-Location-Out (LLO), Leave-Time-Out (LTO) and Leave-Location-and-Time-Out (LLTO) partitioning. See the upstream implementation at `CreateSpacetimeFolds()` (package **CAST**) and Meyer et al. (2018) for further information.

Details

LLO predicts on unknown locations i.e. complete locations are left out in the training sets. The "space" role in Task$col_roles identifies spatial units. If `stratify` is TRUE, the target distribution is similar in each fold. This is useful for land cover classification when the observations are polygons. In this case, LLO with stratification should be used to hold back complete polygons and have a similar target distribution in each fold. LTO leaves out complete temporal units which are identified by the "time" role in Task$col_roles. LLTO leaves out spatial and temporal units. See the examples.

Parameters

- folds (integer(1))
  Number of folds.

- stratify
  If TRUE, stratify on the target column.

- repeats (integer(1))
  Number of repeats.

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.

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.

References

Zhao Y, Karypis G (2002). "Evaluation of Hierarchical Clustering Algorithms for Document Datasets." *11th Conference of Information and Knowledge Management (CIKM)*, 51-524. doi:10.1145/584792.584877.

Examples

```
library(mlr3)
task = tsk("cookfarm_mlr3")
task$set_col_roles("SOURCEID", roles = "space")
task$set_col_roles("Date", roles = "time")

# Instantiate Resampling
rcv = rsmp("repeated_sptcv_cstf", folds = 5, repeats = 3)
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
```

mlr_resamplings_spcv_block
(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 deviding 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](#).

Examples

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

  # Instantiate Resampling
  rcv = rsmp("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
}
```

mlr_resamplings_spcv_buffer

(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 bacground 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.

See Also

ResamplingSpCVDisc

Examples

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

  # Instantiate Resampling
  rcv = rsmp("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
}
```

---

mlr_resamplings_spcv_coords

(sperrorest) Coordinate-based k-means clustering

Description

Splits data by clustering in the coordinate space. See the upstream implementation at sperrorest::partition_kmeans()
and Brenning (2012) for further information.

Details

Universal partitioning method that splits the data in the coordinate space. Useful for spatially homo-
geneous datasets that cannot be split well with rectangular approaches like ResamplingSpCVBlock.

Parameters

- folds (integer(1))
  Number of folds.

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](#).

Examples

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

# Instantiate Resampling
rcv = rsmp("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
```

---

mlr_resamplings_spcv_disc

(sperrorest) Spatial "disc" resampling

Description

Spatial partitioning using circular test areas of one of more observations. Optionally, a buffer around the test area can be used to exclude observations. See the upstream implementation at sperrorest::partition_disc() and Brenning (2012) for further information.

Parameters

- folds (integer(1))
  Number of folds.

- radius (numeric(1))
  Radius of test area disc.

- buffer (integer(1))
  Radius around test area disc which is excluded from training or test set.

- prob (integer(1))
  Optional argument passed down to sample().

- replace (logical(1))
  Optional argument passed down to sample(). Sample with or without replacement.

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.

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.

Examples

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

# Instantiate Resampling
rcv = rsmp("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
```

mlr_resamplings_spcv_env

(blockCV) "Environmental blocking" resampling

Description

Splits data by clustering in the feature space. See the upstream implementation at blockCV::envBlock() and Valavi et al. (2018) for further information.

Details

Useful when the dataset is supposed to be split on environmental information which is present in features. The method allows for a combination of multiple features for clustering.

The input of raster images directly as in blockCV::envBlock() is not supported. See **mlr3spatial** and its raster DataBackends for such support in **mlr3**.

Parameters

- folds (integer(1))
  Number of folds.

- features (character())
  The features to use for clustering.

Super class

mlr3::Resampling -> ResamplingSpCVEnv

Active bindings

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

Methods

Public methods:

- ResamplingSpCVEnv$new()
- ResamplingSpCVEnv$instantiate()
- ResamplingSpCVEnv$clone()

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

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

Usage:
ResamplingSpCVEnv$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:

`ResamplingSpCVEnv$instantiate(task)`

Arguments:

task [Task](#)

A task to instantiate.

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

Usage:

`ResamplingSpCVEnv$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.

Examples

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

  # Instantiate Resampling
  rcv = rsmp("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
}
```

`mlr_resamplings_spcv_tiles`

(sperrorest) Spatial "Tiles" resampling

Description

Spatial partitioning using rectangular tiles. Small partitions can optionally be merged into adjacent ones to avoid partitions with too few observations. This method is similar to `ResamplingSpCVBlock` by making use of rectangular zones in the coordinate space. See the upstream implementation at `sperrorest::partition_disc()` and Brenning (2012) for further information.

Parameters

- dsplit (integer(2))
  Equidistance of splits in (possibly rotated) x direction (dsplit[1]) and y direction (dsplit[2]) used to define tiles. If dsplit is of length 1, its value is recycled. Either dsplit or nsplit must be specified.

- nsplit (integer(2))
  Number of splits in (possibly rotated) x direction (nsplit[1]) and y direction (nsplit[2]) used to define tiles. If nsplit is of length 1, its value is recycled.

- rotation (character(1))
  Whether and how the rectangular grid should be rotated; random rotation is only possible between -45 and +45 degrees. Accepted values: One of c("none", "random", "user").

- user_rotation (character(1))
  Only used when rotation = "user". Angle(s) (in degrees) by which the rectangular grid is to be rotated in each repetition. Either a vector of same length as repeats, or a single number that will be replicated length(repeats) times.

- offset (logical(1))
  Whether and how the rectangular grid should be shifted by an offset. Accepted values: One of c("none", "random", "user").

- user_offset (logical(1))
  Only used when offset = "user". A list (or vector) of two components specifying a shift of the rectangular grid in (possibly rotated) x and y direction. The offset values are relative values, a value of 0.5 resulting in a one-half tile shift towards the left, or upward. If this is a list, its first (second) component refers to the rotated x (y) direction, and both components must have same length as repeats (or length 1). If a vector of length 2 (or list components have length 1), the two values will be interpreted as relative shifts in (rotated) x and y direction, respectively, and will therefore be recycled as needed (length(repeats) times each).

- reassign (logical(1))
  If TRUE, 'small' tiles (as per min_frac and min_n) are merged with (smallest) adjacent tiles. If FALSE, small tiles are 'eliminated', i.e., set to NA.

- min_frac (numeric(1))
  Value must be >=0, <1. Minimum relative size of partition as percentage of sample.

- min_n (integer(1))
  Minimum number of samples per partition.

- iterate (integer(1))
  Passed down to sperrorest::tile_neighbors().

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 sperror-est::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.

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.

See Also

ResamplingSpCVBlock

Examples

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

  # Instantiate Resampling
  rcv = rsmp("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
}
```

mlr_resamplings_sptcv_cluto

(skmeans) Spatiotemporal clustering resampling via CLUTO

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

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",
>   clmethod = "direct",
>   cluto_parameters = NULL,
>   verbose = TRUE
> )
> ```
> *Arguments:*
>
> id character(1)
>
> > Identifier for the resampling strategy.
>
> 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.

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

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. doi:10.1145/584792.584877.

Examples

```
## Not run:
if (mlr3misc::require_namespaces("skmeans", quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("cookfarm_mlr3")
  task$set_col_roles("Date", "time")

  # Instantiate Resampling
  rcv = rsmp("sptcv_cluto", 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
}

## End(Not run)
```

---

mlr_resamplings_sptcv_cstf

> *(CAST) Spatiotemporal "Leave-location-and-time-out" resampling*

Description

Splits data using Leave-Location-Out (LLO), Leave-Time-Out (LTO) and Leave-Location-and-Time-Out (LLTO) partitioning. See the upstream implementation at `CreateSpacetimeFolds()` (package **CAST**) and Meyer et al. (2018) for further information.

Details

LLO predicts on unknown locations i.e. complete locations are left out in the training sets. The `"space"` role in `Task$col_roles` identifies spatial units. If `stratify` is `TRUE`, the target distribution is similar in each fold. This is useful for land cover classification when the observations are polygons. In this case, LLO with stratification should be used to hold back complete polygons and have a similar target distribution in each fold. LTO leaves out complete temporal units which are identified by the `"time"` role in `Task$col_roles`. LLTO leaves out spatial and temporal units. See the examples.

Parameters

- `folds` `(integer(1))`
  Number of folds.

- `stratify`
  If `TRUE`, stratify on the target column.

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.

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.

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.

Examples

```
library(mlr3)
task = tsk("cookfarm_mlr3")
task$set_col_roles("SOURCEID", roles = "space")
task$set_col_roles("Date", roles = "time")

# Instantiate Resampling
rcv = rsmp("sptcv_cstf", 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
```

---

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. In addition coordinates were appended to the task as variables "x" and "y".

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

Usage

```
data(cookfarm_mlr3)
```

Format

R6::R6Class inheriting from TaskRegr.

Usage

```
mlr_tasks$get("cookfarm")
tsk("cookfarm_mlr3")
```

Column roles

The task has set column roles "space" and "time" for variables "Date" and "SOURCEID", respectively. These are used by certain methods during partitioning, e.g., mlr_resamplings_sptcv_cstf with variant "Leave-location-and-time-out". If only one of space or time should left out, the column roles must be adjusted by the user!

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 Iturritxa, 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.

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   *Ecuador Classification Task*

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

---

TaskClassifST   *Create 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_mlr3").

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 coords_as_features = TRUE.

Super classes

[mlr3::Task](#) -> [mlr3::TaskSupervised](#) -> [mlr3::TaskClassif](#) -> TaskClassifST

Active bindings

crs (character(1))
     Returns coordinate reference system of task.

coordinate_names (character())
     Coordinate names.

coords_as_features (logical(1))
     If TRUE, coordinates are used as features. This is a shortcut for task$set_col_roles(c("x",
     "y"), role = "feature") with the assumption that the coordinates in the data are named "x"
     and "y".

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,
  label = NA_character_,
  coordinate_names,
  crs = NA_character_,
  coords_as_features = FALSE,
  extra_args = list()
)
```
Arguments:
id (character(1))
     Identifier for the new instance.

backend ([DataBackend](#))
     Either a [DataBackend](#), or any object which is convertible to a [DataBackend](#) with as_data_backend().
     E.g., am sf 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.

label (character(1))
    Label for the new instance. Shown in as.data.table(mlr_tasks).

coordinate_names (character(1))
    The column names of the coordinates in the data.

crs (character(1))
    Coordinate reference system. WKT2 or EPSG string.

coords_as_features (logical(1))
    If TRUE, coordinates are used as features. This is a shortcut for task$set_col_roles(c("x",
    "y"), role = "feature") with the assumption that the coordinates in the data are named
    "x" and "y".

extra_args (named list())
    Named list of constructor arguments, required for converting task types via [convert_task()](#).

Method coordinates(): Returns coordinates of observations.

Usage:
TaskClassifST$coordinates(row_ids = NULL)

Arguments:

row_ids (integer())
    Vector of rows indices as subset of task$row_ids.

Returns: [data.table::data.table()](#)

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)) {
  task = as_task_classif_st(ecuador,
    target = "slides",
    positive = "TRUE", 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 = as_task_classif_st(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_mlr3").

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 coords_as_features = TRUE.

Super classes

[mlr3::Task](#) -> [mlr3::TaskSupervised](#) -> [mlr3::TaskRegr](#) -> TaskRegrST

Active bindings

crs (character(1))
    Returns coordinate reference system of task.

coordinate_names (character())
    Coordinate names.

coords_as_features (logical(1))
    If TRUE, coordinates are used as features. This is a shortcut for task$set_col_roles(c("x",
    "y"), role = "feature") with the assumption that the coordinates in the data are named "x"
    and "y".

Methods

Public methods:

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

Method `new()`:   Create a new spatiotemporal resampling Task Returns coordinates of observations.

Usage:
```
TaskRegrST$new(
  id,
  backend,
  target,
  label = NA_character_,
  coordinate_names,
  crs = NA_character_,
  coords_as_features = FALSE,
  extra_args = list()
)
```

Arguments:

`id (character(1))`
   Identifier for the new instance.

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

`target (character(1))`
   Name of the target column.

`label (character(1))`
   Label for the new instance. Shown in `as.data.table(mlr_tasks)`.

`coordinate_names (character(1))`
   The column names of the coordinates in the data.

`crs (character(1))`
   Coordinate reference system. WKT2 or EPSG string.

`coords_as_features (logical(1))`
   If `TRUE`, coordinates are used as features. This is a shortcut for `task$set_col_roles(c("x", "y"), role = "feature")` with the assumption that the coordinates in the data are named `"x"` and `"y"`.

`extra_args (named list())`
   Named list of constructor arguments, required for converting task types via [convert_task()](#).

Method `coordinates()`:

Usage:
```
TaskRegrST$coordinates(row_ids = NULL)
```

Arguments:

`row_ids (integer())`
   Vector of rows indices as subset of `task$row_ids`.

Returns: [data.table::data.table()](#)

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

76