Package 'mlr3spatiotempcv'

January 5, 2021

Title Spatiotemporal Resampling Methods for 'mlr3'

Version 0.1.1

Date 2021-01-05

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

License LGPL-3

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

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

Depends R (>= 3.5.0)

Imports checkmate, data.table, ggplot2, mlr3 (>= 0.7.0), mlr3misc (>= 0.1.7), paradox, R6, testthat (>= 3.0.0), utils

Suggests bbotk, blockCV (>= 2.1.1), CAST, ggsci, ggtext, GSIF, knitr, lgr, mlr3filters, mlr3pipelines, mlr3tuning, patchwork, plotly, rmarkdown, rpart, sf, skmeans, vdiffr, withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.1.1

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

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

Repository CRAN

Date/Publication 2021-01-05 18:20:02 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.

Additional resources

- Book on mlr3: https://mlr3book.mlr-org.com
- 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**
- Spatiotemporal resampling methods: **mlr3spatiotempcv**
- 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.

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

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",
  crs = NULL,
  ...
)

## S3 method for class 'ResamplingRepeatedCV'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  crs = 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 only a list with all **ggplot2** resamplings 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.
crs	[character] EPSG code of the CRS for x and y axes.
...	Not used.
repeats_id	[numeric] Repetition ID to plot.
x	[Resampling] mlr3 spatial resampling object of class ResamplingCV or ResamplingRepeatedCV.

Examples

```
if (mlr3misc::require_namespaces(c("sf", "patchwork", "ggtext"), 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, 1)
  autoplot(resampling, task, c(1, 2))
}
```

---

autoplot.ResamplingSpCVBlock

Visualization Functions for SpCV Block Methods.

Description

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

Usage

```
## S3 method for class 'ResamplingSpCVBlock'
autoplot(
  object,
```

```
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  crs = NULL,
  ...
)

## S3 method for class 'ResamplingRepeatedSpCVBlock'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  crs = 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` only a list with all **ggplot2** resamplings 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.
`crs`	[character] EPSG code of the CRS for x and y axes.
`...`	Not used.

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.

Examples

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

  ## 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)

  ## Visualize train/test splits of multiple folds
  autoplot(resampling, task, fold_id = c(1, 2))

  # list of ggplot2 resamplings
  plot_list = autoplot(resampling, task,
    fold_id = c(1, 2), grid = FALSE)
}
```

---

autoplot.ResamplingSpCVBuffer

Visualization Functions for SpCV Buffer Methods.

Description

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

Usage

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

## 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` only a list with all **ggplot2** resamplings 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.
`crs`	[character] EPSG code of the CRS for x and y axes.
`...`	Not used.
`x`	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVBuffer.

Examples

```
if (mlr3misc::require_namespaces(c("sf", "blockCV"), quietly = TRUE)) {
  library(mlr3)
  library(mlr3spatiotempcv)
  task = tsk("ecuador")
  resampling = rsmp("spcv_buffer", theRange = 10000)
  resampling$instantiate(task)
```

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

---

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",
  crs = NULL,
  ...
)

## S3 method for class 'ResamplingRepeatedSpCVCoords'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  crs = 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 only a list with all **ggplot2** resamplings 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.
crs	[character] EPSG code of the CRS for x and y axes.
...	Not used.
repeats_id	[numeric] Repetition ID to plot.
x	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVCoords or ResamplingRepeatedSpCVCoords.

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, 1)
  autoplot(resampling, task, c(1, 2))
}
```

---

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",
  crs = NULL,
  ...
)

## S3 method for class 'ResamplingRepeatedSpCVEnv'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  crs = 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` only a list with all **ggplot2** resamplings 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.
crs	[character] EPSG code of the CRS for x and y axes.
...	Not used.
repeats_id	[numeric] Repetition ID to plot.
x	[Resampling] mlr3 spatial resampling object of class ResamplingSpCVEnv or ResamplingRepeatedSpCVEnv.

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, 1)
  autoplot(resampling, task, c(1, 2))
}
```

---

autoplot.ResamplingSptCVCluto

Visualization Functions for SptCV Cluto Methods.

Description

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

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",
  crs = NULL,
  nticks_x = 3,
```

```
  nticks_y = 3,
  point_size = 3,
  axis_label_fontsize = 11,
  ...
)

## S3 method for class 'ResamplingRepeatedSptCVCluto'
autoplot(
  object,
  task,
  fold_id = NULL,
  repeats_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  crs = 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` only a list with all **ggplot2** resamplings 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.
`crs`	[character] EPSG code of the CRS for x and y axes.

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.

... Not used.

repeats_id [numeric]
Repetition ID to plot.

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

Examples

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

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

## End(Not run)
```

---

autoplot.ResamplingSptCVCstf

Visualization Functions for SptCV Cstf Methods.

Description

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

Usage

```
## S3 method for class 'ResamplingSptCVCstf'
autoplot(
  object,
```

```
  task,
  fold_id = NULL,
  plot_as_grid = TRUE,
  train_color = "#0072B5",
  test_color = "#E18727",
  tickformat_date = "%Y-%m",
  crs = NULL,
  nticks_x = 3,
  nticks_y = 3,
  point_size = 3,
  axis_label_fontsize = 11,
  ...
)

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

## 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 only a list with all **ggplot2** resamplings 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.

crs
: [character]
EPSG code of the CRS for x and y axes. If not set, EPSG 4326 (WGS84) is used.

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.

...
: Not used.

repeats_id
: [numeric]
Repetition ID to plot.

x
: [Resampling]
mlr3 spatial resampling object of class ResamplingSptCVCstf or ResamplingRepeatedSptCVCstf.

Examples

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

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

`mlr_tasks_cookfarm` *Cookfarm Profiles Regression Task*

Description

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

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

See GSIF::cookfarm for more detailed information.

Usage

```
data(cookfarm)
```

Format

R6::R6Class inheriting from TaskRegr.

Usage

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

References

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

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

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

See Also

Dictionary of Tasks: mlr_tasks

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

Other Task: `TaskClassifST`, `TaskRegrST`, `mlr_tasks_diplodia`, `mlr_tasks_ecuador`

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`

ResamplingRepeatedSpCVBlock

Repeated Spatial Cross Validation Resampling

Description

Spatial Block Cross validation implemented by the `blockCV` package.

Details

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.

Super class

`mlr3::Resampling` -> `ResamplingRepeatedSpCVBlock`

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 "coordinate-based" repeated resampling instance.

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

```
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(5000, 10000))
  rrcv$instantiate(task)

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

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

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

`ResamplingRepeatedSpCVCoords`

Repeated Spatial Cross Validation Resampling

Description

Spatial Cross validation following the "k-means" approach after Brenning 2012.

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.

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
```

`ResamplingRepeatedSpCVEnv` *Repeated Environmental Block Cross Validation Resampling*

Description

Environmental Block Cross Validation. This strategy uses k-means clustering to specify blocks of similar environmental conditions. Only numeric features can be used. The `features` used for building blocks can be specified in the `param_set`. By default, all numeric features are used.

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 "coordinate-based" repeated resampling instance.

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.

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
}
```

`ResamplingRepeatedSptCVCluto`

Repeated Spatiotemporal Cluster Resampling

Description

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

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

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

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

Details

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

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

Copyright

CLUTO's copyright is as follows:

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

Super class

`mlr3::Resampling` -> `ResamplingRepeatedSptCVCluto`

Public fields

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

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

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

`verbose` logical
: Whether to show `vcluster` progress and summary output.

Active bindings

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

Methods

Public methods:

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

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

Usage:

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

Arguments:

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

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

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

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

`verbose` logical
: Whether to show `vcluster` progress and summary output.

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

Usage:

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

Arguments:

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

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

Usage:

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

Arguments:

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

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

Usage:

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

Arguments:

`task` Task
: A task to instantiate.

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

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

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

`verbose` logical
: Whether to show `vcluster` progress and summary output.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

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

Examples

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

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

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

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

  # Internal storage:
  rrcv$instance # table
}

## End(Not run)
```

`ResamplingRepeatedSptCVCstf` *Repeated Spatiotemporal Cluster Resampling*

Description

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

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

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

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

Details

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

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

Copyright

CLUTO's copyright is as follows:

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

Super class

`mlr3::Resampling` -> `ResamplingRepeatedSptCVCstf`

Public fields

`space_var character(1)`
: Column name identifying the spatial units.

`time_var character(1)`
: Column name identifying the temporal units.

`class character(1)`
Column name identifying a class unit (e.g. land cover).

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",
  space_var = NULL,
  time_var = NULL,
  class = NULL
)
```

Arguments:

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

`space_var character(1)`
Column name identifying the spatial units.

`time_var character(1)`
Column name identifying the temporal units.

`class character(1)`
Column name identifying a class unit (e.g. land cover).

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. http://glaros.dtc.umn.edu/gkhome/node/167.

Examples

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

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

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

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

`ResamplingSpCVBlock` *Spatial Block Cross Validation Resampling*

Description

Spatial Block Cross validation implemented by the `blockCV` package.

Details

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.

Super class

`mlr3::Resampling` -> `ResamplingSpCVBlock`

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 "Environmental Block" resampling instance.

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 = 1000)
  rcv$instantiate(task)

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

  # Internal storage:
  rcv$instance
}
```

`ResamplingSpCVBuffer` *Spatial Buffer Cross Validation Resampling*

Description

Spatial Buffer Cross validation implemented by the `blockCV` package.

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.

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.

Note

However, the default settings allow to conduct a leave-one-out cross validation for two-class, multi-class and continuous response data, where each observation is one test set. For each test, all observations outside the buffer around the test observation are included in the training set.

The parameter `spDataType = PB` and `addBG` are designed for presence-background data in species distribution modelling. If `spDataType = PB`, test sets are only created for each presence observation (task$positive). The option `addBG = TRUE` adds the background data inside the buffer to the corresponding test sets. For each test set, all observations outside the buffer around the test observation are included in the training set.

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_buffer", theRange = 10000)
  rcv$instantiate(task)

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

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

ResamplingSpCVCoords *Spatial Cross Validation Resampling*

Description

Spatial Cross validation following the "k-means" approach after Brenning 2012.

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 "Environmental Block" resampling instance.

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
```

ResamplingSpCVEnv *Environmental Block Cross Validation Resampling*

Description

Environmental Block Cross Validation. This strategy uses k-means clustering to specify blocks of similar environmental conditions. Only numeric features can be used. The `features` used for building blocks can be specified in the `param_set`. By default, all numeric features are used.

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.

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
}
```

`ResamplingSptCVCluto` *Spatioemporal Cluster Resampling*

Description

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

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

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

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

Details

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

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

Copyright

CLUTO's copyright is as follows:

The CLUTO package is copyrighted by the Regents of the University of Minnesota. It can be freely used for educational and research purposes by non-profit institutions and US government agencies only. Other organizations are allowed to use CLUTO only for evaluation purposes, and any further uses will require prior approval. The software may not be sold or redistributed without

Super class

`mlr3::Resampling` -> `ResamplingSptCVCluto`

Public fields

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

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

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

`verbose` logical
: Whether to show `vcluster` progress and summary output.

Active bindings

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

Methods

Public methods:

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

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

Usage:

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

Arguments:

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

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

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

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

`verbose` logical
: Whether to show `vcluster` progress and summary output.

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

Usage:

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

Arguments:

`task` Task
: A task to instantiate.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

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

Examples

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

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

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

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

## End(Not run)
```

`ResamplingSptCVCstf` *Create Spatiotemporal Folds Using Predefined Groups*

Description

Implementation of `CAST::CreateSpaceTimeFolds()`.

Details

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

Super class

`mlr3::Resampling` -> `ResamplingSptCVCstf`

Public fields

`space_var character(1)`
: Column name identifying the spatial units.

`time_var character(1)`
: Column name identifying the temporal units.

`class character(1)`
: Column name identifying a class unit (e.g. land cover).

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",
  space_var = NULL,
  time_var = NULL,
  class = NULL
)
```

Arguments:

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

`space_var character(1)`
: Column name identifying the spatial units.

`time_var character(1)`
: Column name identifying the temporal units.

`class character(1)`
: Column name identifying a class unit (e.g. land cover).

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

Usage:

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

Arguments:

`task` Task
: A task to instantiate.

`space_var` [character]
: Column name identifying the spatial units.

`time_var` [character]
: Column name identifying the temporal units.

`class` [character]
: Column name identifying a class unit (e.g. land cover).

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")
```

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

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

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

`TaskClassifST` *SpatioTemporal Classification Task*

Description

This task specializes Task and TaskSupervised for spatiotemporal classification problems. The target column is assumed to be a factor. The `task_type` is set to `"classif"` and `"spatiotemporal"`.

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

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

Super classes

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

Public fields

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

Methods

Public methods:

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

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

Usage:

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

Arguments:

`id` [character(1)]
Identifier for the task.

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

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

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

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

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

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

Usage:

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

Arguments:

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

Method `print()`**:** Print the task.

Usage:

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

Arguments:

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

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Task: TaskRegrST, mlr_tasks_cookfarm, mlr_tasks_diplodia, mlr_tasks_ecuador

Examples

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

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

TaskRegrST *SpatioTemporal Regression Task*

Description

This task specializes Task and TaskSupervised for spatiotemporal classification problems.

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

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

Super classes

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

Public fields

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

Methods

Public methods:

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

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

Usage:

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

Arguments:

`id` [character(1)]
Identifier for the task.

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

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

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

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

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

Usage:

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

Arguments:

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

Method `print()`**:** Print the task.

Usage:

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

Arguments:

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

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Task: `TaskClassifST`, `mlr_tasks_cookfarm`, `mlr_tasks_diplodia`, `mlr_tasks_ecuador`

Index