Journal of Statistical Software

November 2024, Volume 111, Issue 7. doi: 10.18637/jss.v111.i07

mlr3spatiotempcv: Spatiotemporal Resampling Methods for Machine Learning in R

Patrick Schratz
Friedrich Schiller University Jena

Marc Becker
Ludwig-Maximilians-Universität München

Michel Lang
TU Dortmund University

Alexander Brenning
Friedrich Schiller University Jena

Abstract

Spatial and spatiotemporal machine-learning models require a suitable framework for their model assessment, model selection, and hyperparameter tuning, in order to avoid error estimation bias and over-fitting. This contribution provides an overview of the state-of-the-art in spatial and spatiotemporal cross-validation techniques and their implementations in R while introducing the R package **mlr3spatiotempcv** as an extension package of the machine-learning framework **mlr3**. Currently various R packages implementing different spatiotemporal partitioning strategies exist: **blockCV**, **CAST**, **skmeans** and **sperrorest**. The goal of **mlr3spatiotempcv** is to gather the available spatiotemporal resampling methods in R and make them available to users through a simple and common interface. This is made possible by integrating the package directly into the **mlr3** machine-learning framework, which already has support for generic non-spatiotemporal resampling methods such as random partitioning. One advantage is the use of a consistent nomenclature in an overarching machine-learning toolkit instead of a varying package-specific syntax, making it easier for users to choose from a variety of spatiotemporal resampling methods. This package avoids giving recommendations which method to use in practice as this decision depends on the predictive task at hand, the autocorrelation within the data, and the spatial structure of the sampling design or geographic objects being studied.

Keywords: cross-validation, predictive performance, machine learning, autocorrelation, spatial, temporal, R.

1. Introduction

Spatial and spatiotemporal prediction tasks are common in applications ranging from en-

vironmental sciences to archaeology and epidemiology. While sophisticated mathematical frameworks have long been developed in spatial statistics to characterize predictive uncertainties under well-defined mathematical assumptions such as intrinsic stationarity (e.g., Cressie 1993), computational estimation procedures have only been proposed more recently to assess predictive performances of spatial and spatiotemporal prediction models (Brenning 2005, 2012; Pohjankukka, Pahikkala, Nevalainen, and Heikkonen 2017; Roberts *et al.* 2017).

Although alternatives such as the bootstrap exist since some decades (Efron and Gong 1983; Hand 1997), cross-validation (CV) is a particularly well-established, easy-to-implement algorithm for *model assessment* of supervised machine-learning models (Efron and Gong 1983, and next section) and *model selection* (Arlot and Celisse 2010). In its basic form, CV is based on resampling the data without paying attention to any possible dependence structure, which may arise from, e.g., grouped or structured data, or underlying environmental processes inducing some sort of spatial coherence at the landscape scale. In treating dependent observations as independent, or ignoring autocorrelation, CV test samples may in fact be heavily correlated with, or even pseudo-replicates of, the data used for training the model, which introduces a potentially severe bias in assessing the transferability of flexible machine-learning (ML) models.

This CV bias is well-known in spatial as well as non-spatial prediction (Brenning 2005; Brenning and Lausen 2008; Arlot and Celisse 2010; Roberts *et al.* 2017) and in forecasting (Bergmeir, Hyndman, and Koo 2018). It is most easily understood from a predictive modeling perspective by focusing on the question of where (and when) the model should be used for prediction. In crop classification from remotely-sensed data, for instance, learning samples routinely contain multiple grid cells from a sample of fields with known crop type, for instance 2000 grid cells from 100 fields scattered across a large study region. The purpose of training a model on this particular sample is to make predictions on other, new fields within the same geographic domain (*intra-domain* prediction, Brenning 2005) – not *within* the same field, which obviously presents only a single crop type that is already known from the training sample. In this specific situation it would therefore seem rather unwise to train a model on a simple random subsample of grid cells, and to test it on the remaining data, using other grid cells from the same fields, as if one wanted to predict within a field. The results from this performance assessment would be over-optimistic, and will not reflect the model's true ability to make predictions for new fields. To mimic the predictive situation for which the model is trained, one would rather have to resample at the level of fields, not grid cells (Peña and Brenning 2015). If the model was to be applied to adjacent agricultural regions, i.e., outside the learning sample's spatial domain (*extra-domain* prediction, Brenning 2005), it would even seem necessary to resample at a higher level of spatial aggregation, i.e., at the level of sub-regions within the learning sample, in order to realistically mimic the actual prediction task. The CV resampling needed therefore depends as much on the prediction task itself as on the data structure or dependency at hand.

While it is not the purpose of this article to recommend specific resampling schemes for specific use cases, the example from above may suffice to motivate the use of appropriate spatial and spatiotemporal cross-validation techniques, and the need for a unified framework and computational toolbox that accommodates a variety of prediction tasks that may be applicable to a broad range of application scenarios. Package **mlr3spatiotempcv** is such a toolbox. We would like to note that design-based approaches are also a viable approach in situations in which it is possible to obtain additional independently sampled test data as a probability sample (Wadoux, Heuvelink, De Bruin, and Brus 2021).

This toolbox, implemented as an open-source R package, builds upon and generalizes several existing toolboxes that have been developed in recent years for more specific settings (Table 1). The earliest and most comprehensive of these implementations is the **sperrorest** R package (Brenning 2012), which provides an extensible framework and includes predefined resampling strategies based on geometric blocking, clustering, and buffering. In contrast, packages **blockCV** (Valavi *et al.* 2019) and **ENMeval** (Muscarella *et al.* 2014) were developed for block and buffer resampling with a focus on species distribution modeling (Rest, Pinaud, Monestiez, Chadoeuf, and Bretagnolle 2014). However, neither of these have been integrated into established machine-learning frameworks such as **mlr3** (Lang *et al.* 2019) or **caret**/**tidymodels** (Kuhn 2008; Kuhn and Wickham 2020), and all of them lack support for temporal prediction tasks. The **CAST** package, in contrast, focuses on spatiotemporal prediction tasks and makes use of some functions of the **caret** framework (Meyer, Milà, Ludwig, Linnenbrink, and Schumacher 2024; Meyer, Reudenbach, Hengl, Katurji, and Nauss 2018). One limitation of all the packages which provide only a subset of the machine learning workflow, e.g., resampling methods, is the sole focus on model assessment. Through the integration into the **mlr3** framework, **mlr3spatiotempcv** offers model selection capabilities, i.e., the seamless evaluation and application of various algorithms across different preprocessing and optimization settings while being able to make use of parallel execution and enhanced logging abilities. During the initial development of **mlr3spatiotempcv**, it has been the first toolbox aiming to integrate standalone spatiotemporal validation strategies into an existing machine-learning framework. Meanwhile, package **spatialsample** has undertaken similar efforts and added the ability to use spatial resampling methods for the **tidymodels** framework (Mahoney, Silge, and Posit Software, PBC 2023b; Mahoney, Johnson, Silge, Frick, Kuhn, and Beier 2023a). In addition parallel efforts to support spatial validation in the Python world have been made by the **spacv** (Comber 2020) and **spatial-kfold** (Ghariani 2023) packages, yet often with only a subset of available methods compared to what **mlr3spatiotempcv** offers.

Thus, **mlr3spatiotempcv** implements for the first time a comprehensive state-of-the-art compilation of spatial and spatiotemporal partitioning schemes that is well-integrated into a comprehensive machine-learning framework in R, the **mlr3** ecosystem. This package is furthermore equipped with a variety of two- and three-dimensional visualization capabilities. The hope is that this implementation will simplify and facilitate reproducible geospatial modeling and code-sharing across a broad range of application domains.

The purpose of this article is to give an overview of the methods implemented in the R package **mlr3spatiotempcv**. After presenting the conceptual background in Section 2, the overall structure of the **mlr3spatiotempcv** package is outlined in Section 3. Next, various spatial and spatiotemporal partitioning techniques are contrasted and compared Section 4, before their application is demonstrated in a machine-learning model assessment in Section 5. Finally, recommendations for the selection of suitable resampling techniques are given in Section 6. Section 7 concludes.

2. Spatial and spatiotemporal CV

In CV for predictive model assessment, the following formal setting is considered. The interest is in predicting a numerical or categorical response y of an object or instance using a feature vector $\mathbf{x} = (x^{(1)}, \ldots, x^{(p)})^\top \in \mathbb{R}^p$ and a model $\hat{f}_{\mathcal{L}}$ that has been trained on a learning sample $\mathcal{L} = \{(y_i, \mathbf{x}_i),\ i = 1, ..., n\}$. The goal is to estimate the expected value of the performance

of $\hat{f}_{\mathcal{L}}$,

$$perf(\hat{f}_{\mathcal{L}}) := \mathsf{E}(l(Y, \hat{f}_{\mathcal{L}}(X))),$$

where l is a real-valued loss function, and the expected value is with respect to the probability distribution of X, the features of an instance (Y, X) drawn randomly from the underlying population. This is referred to as the *actual* or *conditional* performance measure, as it is conditional on $\mathcal{L}$ (Hand 1997). The loss function can take a variety of forms such as the misclassification error $I(Y \neq \hat{f}_{\mathcal{L}}(X))$ in classification, or the squared error $(Y - \hat{f}_{\mathcal{L}}(X))^2$ in regression, among many other possible measures. The choice of the performance measure is equally critical as the choice of the estimation procedure, but it is beyond the scope of this contribution to discuss performance measures for regression and classification (see, e.g., Hand 1997 for classification, and Hyndman and Koehler 2006 for regression and forecasting tasks).

Since there is only a sample $\mathcal{T}$ of test data drawn from the population, one can only *estimate* the conditional performance of $\hat{f}_{\mathcal{L}}$:

$$\widehat{perf}_T(\hat{f}_{\mathcal{L}}) = \frac{1}{|\mathcal{T}|} \sum_{(Y,X)\in\mathcal{T}} l(Y, \hat{f}_{\mathcal{L}}(X)).$$

This representation as a point estimator of $perf(\hat{f}_{\mathcal{L}})$ underlines the importance of using a random sample for model assessment to avoid estimation bias. Other estimators than the simple mean may be required when $\mathcal{T}$ is not a simple random sample, for instance a stratified random sample (e.g., Thompson 2012). As always, judgment sampling may lead to uncontrollable bias.

Since re-using the learning sample $\mathcal{L}$ for testing, i.e., $\mathcal{T} := \mathcal{L}$, would over-optimistically estimate the model's predictive performance on new instances (so-called *resubstitution* or *apparent* model performance; Hand 1997), CV partitions the sample $\mathcal{L}$ into disjoint training and test sets. Specifically, $\mathcal{L}$ is split into k partitions,

$$\mathcal{L} = \mathcal{L}_1 \cup \ldots \cup \mathcal{L}_k, \qquad \mathcal{L}_i \cap \mathcal{L}_j = \emptyset \quad \text{for all } i \neq j,$$

and a model $\hat{f}_{(i)}$ is fitted on $\mathcal{L}_{(i)} := \mathcal{L} \setminus \mathcal{L}_i$, while $\mathcal{L}_i$ is withheld for testing. This is repeated for $i = 1, \ldots, k$ in order to effectively use the entire sample for testing, while keeping training and test sets disjoint at all times. The k-fold CV estimator can therefore be written as

$$\widehat{perf}_{\mathcal{L},CV}(f) := \frac{1}{k} \sum_{i=1}^{k} \widehat{perf}_{\mathcal{L}_i}(\hat{f}_{\mathcal{L}_{(i)}}),$$

where f is a ML algorithm, i.e., a mapping that trains a model $\hat{f}_{\mathcal{S}}$ using any suitable training sample $\mathcal{S}$. The use of $k = 5$ or $k = 10$ folds is most commonly seen in practice, and these preferences are also supported by theory (Bengio and Grandvalet 2004; Cawley and Talbot 2010). The k-fold CV estimator of model performance is a nearly unbiased estimator of the conditional performance measure when the observations were drawn independently (Efron and Gong 1983). Since $\widehat{perf}_{\mathcal{L},CV}(f)$ still depends on the particular partitioning chosen for $\mathcal{L}$, it is sometimes recommended to repeat the estimation using different random partitionings (r-repeated k-fold cross-validation) to reduce the influence of randomness when creating partitions (Vanwinckelen and Blockeel 2012).

In traditional CV, the partitioning is based on uniform random resampling, which ignores spatial or temporal autocorrelation or any existing grouping structure as well as the structure

of the prediction task, and may result in over-optimistic performance estimates. Several approaches have therefore been proposed in the literature and implemented in software to accommodate a variety of predictive situations (Table 1).

Approaches based on *spatial blocking* (or sometimes called *grouping*) require either the construction of spatial zones, or the use of pre-existing spatial structures in the data. Let's refer to these spatial units or blocks as $\mathcal{Z}_i$, $1 \leq i \leq n_z$. These blocks are often constructed to serve as the $k = n_z$ spatial partitions, for example by performing k-means clustering of the sample coordinates (Ruß and Brenning 2010), which we refer to as *coordinate-based clustering*; or generating the desired number of rectangular blocks as an example of *geometric partitioning*. The blocks may also be defined by a modeler based on an arbitrary partitioning of the study region based on an external data source, which we refer to as *custom resampling*. This often used when the data is grouped. For example, when using to multi-level sampling designs or studying spatial objects, it has been proposed to apply leave-one-out (LOO) cross-validation at the site level (Martin, Plourde, Ollinger, Smith, and McNeil 2008; Kasurak, Kelly, and Brenning 2011) or, in animal movement studies, at the animal level (Anderson *et al.* 2005). We will broadly refer to such groups of observations as "blocks" in a generic sense, regardless of the shape or origin of the groups. Also, data can be partitioned in feature space instead of geographic space, which has been referred to as "environmental blocking" (Roberts *et al.* 2017).

When n_z is much larger than the desired number of folds, k, then a partitioning can be applied to the zones themselves. In this case, the zone indices $1, \ldots, n_z$ are grouped into k equally sized subsets $\mathcal{I}_1, \ldots, \mathcal{I}_k$. This approach has been applied, for example, in spatial CV at the agricultural field level (Peña and Brenning 2015). We would like to emphasize the conceptual distinction between *CV at the block level*, referring to this scenario, and *leave-one-block-out CV*, where the blocks themselves define the CV partitions. Figure 2 gives an overview of the conceptual framework and terminology used in this work.

One variant of CV is LOO CV, which has long been established in geostatistics (Cressie 1993), sometimes with a focus on the spatial distribution of LOO error (Willmott and Matsuura 2006). Although this is just a special case of non-spatial CV with $k = n$, it is sometimes also referred to as spatial CV (Willmott and Matsuura 2006).

Spatial variants of CV have been proposed that apply an exclusion *buffer* or guard zone to the test locations to separate them from the training data (Brenning 2005; Roberts *et al.* 2017). One approach that has been proposed for defining a separation distance is to use the range of autocorrelation of model residuals to determine the buffer distance, as this seeks to establish independence conditional on the predictors (Brenning 2005; Roberts *et al.* 2017). Recently, spatial LOO CV has been extended to generate spatial *prediction error profiles*, which relate prediction error to prediction distance (Brenning 2023).

It should be noted that k-fold CV with a large k, and LOO CV in particular ($k = n$) is computationally expensive due to the large number of models being fitted. There is ongoing debate regarding the bias–variance trade-off of these methods in model selection and assessment (Kohavi 1995; Arlot and Celisse 2010; Zhang and Yang 2015).

In the purely temporal domain, a special case is to leave out temporal observational units (or time slices; leave-time-out or LTO CV), as in leave-one-year-out CV (Anderson *et al.* 2005; Brenning 2005). CV and hold-out validation strategies for time series have been discussed more extensively in the forecasting literature, considering also the effects of serial autocorrela-

tion (Bergmeir *et al.* 2018); these methods are not the focus of the implementation presented in this work.

Turning to prediction tasks with spatiotemporal data, various spatial, temporal, or spatiotemporal partitioning strategies are being used, depending on the specific study objectives. While the former two ignore the temporal and spatial dimension of the data, respectively, it has also been proposed to leave out random subsets of locations and time points (Meyer *et al.* 2018). Details of these and other implementations are outlined in the respective subsections of Section 4.

3. mlr3spatiotempcv within the mlr3 ecosystem

With the increased awareness of the importance of spatial and spatiotemporal resampling strategies and the growing popularity of R in environmental modeling and geocomputation, it is important to equip ML frameworks such as **mlr3** with suitable algorithms. In this context, the **mlr3** ecosystem stands out as a unified, object-oriented and extensible framework designed to accommodate numerous ML tasks with a variety of learners, feature and model selection tools, and model assessment capabilities (Lang *et al.* 2019; Bischl, Sonabend, Kotthoff, and Lang 2024). All of these are supported by advanced visualization tools, which are particularly important in a spatial and spatiotemporal setting. Additionally, **mlr3pipelines** (Binder, Pfisterer, Lang, Schneider, Kotthoff, and Bischl 2021) provides a plethora of preprocessing operators to conveniently build ML pipelines which can be resampled, tuned and benchmarked as regular learners.

With its integrative approach and its aim to provide long-term support, **mlr3** overcomes the challenges of combining multiple specialized packages with poorly standardized interfaces. Issues that practitioners often face include varying argument lists of learners, different return values of `predict()` methods, and support for only specific feature types. These challenges result in substantial overhead and possible reproducibility issues, which are exacerbated by asynchronous development timelines of different components of the used ML pipelines.

Compared to other existing machine-learning frameworks in R (e.g., **caret** or **tidymodels**), **mlr3** is the only one that provides a dedicated object-oriented framework for spatial and spatiotemporal resampling methods. In addition **mlr3** uses efficient core dependencies from the **data.table** (Barrett, Dowle, Srinivasan, Gorecki, Chirico, and Hocking 2024) and **R6** (Chang 2021) packages, which are particularly well suited for large datasets. Through this object-oriented approach, **mlr3** uses substantially less memory than other frameworks due to the use of pointer references and the avoidance of deep object copies whenever possible (Bischl *et al.* 2024).

Within the **mlr3** ecosystem, partitioning strategies are represented by their own objects of class '`Resampling`', most of which are available within **mlr3** itself (e.g., random CV); other specialized strategies are defined in extension packages such as **mlr3spatiotempcv**. In the ML pipeline, these objects define the data splits used for model assessment and selection (hyperparameter tuning) by ML algorithms. Spatial and spatiotemporal partitioning techniques in **mlr3spatiotempcv** are currently mostly imported and interfaced from other packages, in particular **sperrorest**, **blockCV** and **CAST**, in order to expose them to **mlr3** functionality. By closely following previously proposed methods and existing implementations, we allow users of these established and tested approaches to transition into **mlr3** without having to adjust

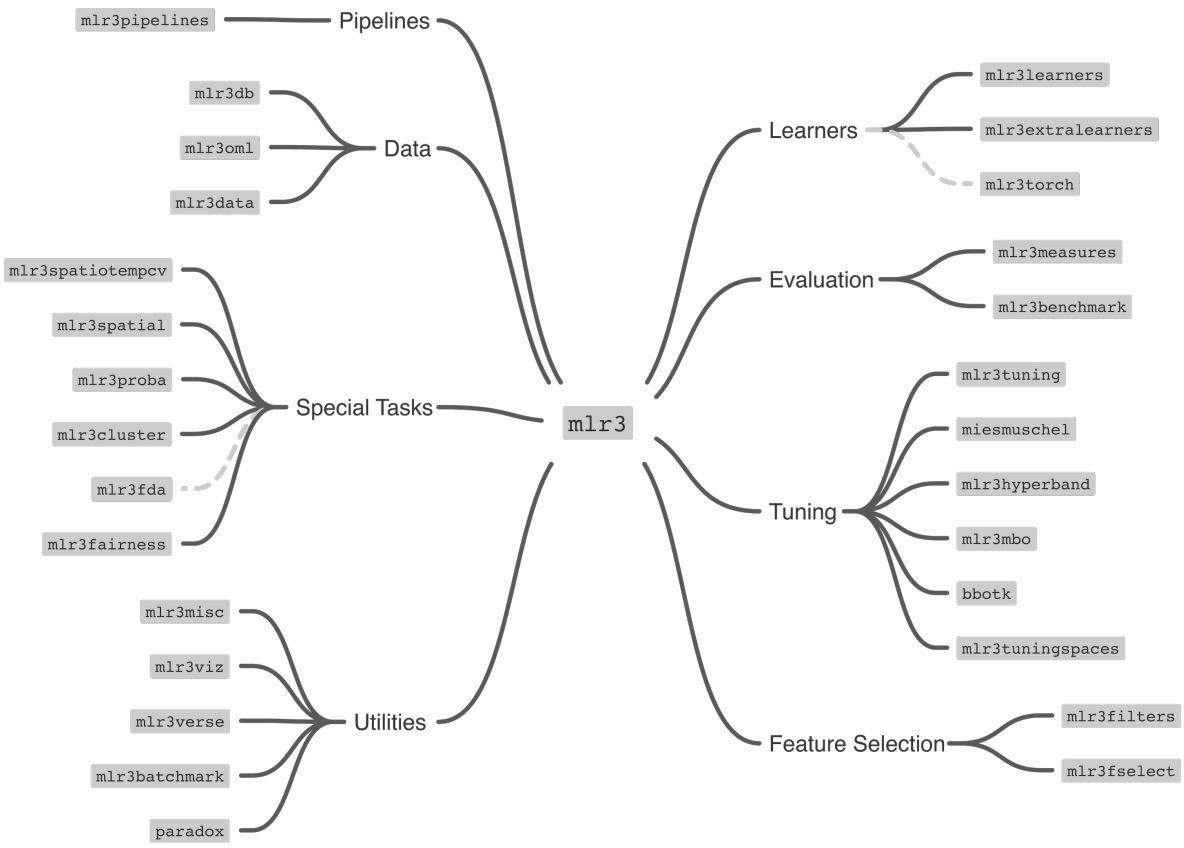

Figure 1: Overview of the **mlr3** ecosystem. The packages with gray dashed lines are still in development, all others have a stable interface. Source: Bischl *et al.* (2024).

their resampling procedures. To reduce unnecessary upstream dependencies, some methods were re-implemented instead of importing them from the respective upstream packages.

Resampling objects in **mlr3spatiotmpcv** inherit from class '`mlr3::Resampling`' and can be created from established object classes for geospatial data in R, including simple features (Pebesma 2018), which facilitates their integration into domain-specific workflows in the geospatial sciences. Support for projected (planar) and unprojected (geographic) coordinate reference systems (CRS) currently varies depending on the partitioning techniques used, since these inherit their behavior from the underlying upstream packages.

Partitioning objects in **mlr3spatiotempcv** are equipped with generic `plot()` and `autoplot()` methods to visualize the created partitions. `autoplot()` is **ggplot2**-based (Wickham 2016) and uses **ggplot2** in two-dimensional geographic space and **plotly** (Sievert 2020) in the three dimensional case, i.e., geographic space plus time.

While **mlr3spatiotempcv** solely focuses on spatiotemporal resampling methods and their visualization, other packages such as **mlr3spatial** (Becker and Schratz 2024) or **mlr3temporal** (Lang, Pfisterer, Gruber, Nawrath, and Arshadipour 2023) in the **mlr3** ecosystem provide dedicated spatiotemporal learner and prediction methods (see Figure 1). From a user perspective, this package structure results in the following workflow for model assessment with **mlr3spatiotempcv** within **mlr3**: After choosing a ML algorithm that is supported by **mlr3**

and setting up a learner object, users need to select hyperparameters that should be tuned and specify these in a `paradox::ParamSet`. Next, a suitable resampling scheme available within **mlr3spatiotempcv** is selected that mimics the spatial and/or temporal structure of the prediction task, such as spatial extrapolation, or forecasting of spatial time series. This information is used to create a '`Resampling`' object which is used within a (nested) CV to estimate the model performance. When using nested CV, the resampling schemes in the inner (tuning, `mlr3tuning::AutoTuner`) and outer loop (performance estimation, `resample()`) should be identical (Schratz, Muenchow, Iturritxa, Richter, and Brenning 2019). To evaluate the (nested) resampling, an adequate performance measure with respect to the response variable, such as the misclassification rate (classification) or the root-mean-square error (regression), must be selected and specified within `mlr3tuning::AutoTuner` and `resample$score()`. These choices now allow the user to execute the model assessment via either `resample()` (single model) or `benchmark()` (multiple models), and the results can be summarized visually (via **mlr3viz**; Lang, Schratz, Sonabend, Becker, Richter, and Zobolas 2024b) or in tabular form by accessing the respective fields of the returned '`ResampleResult`' object.

Additional examples and tutorial can be found in the **mlr3** book (https://mlr3book.mlr-org.com/) or the **mlr3** gallery (https://mlr-org.com/gallery/).

4. Spatiotemporal partitioning methods

At the most general level, resampling methods are categorized according to the level at which the data is partitioned and resampled (see Figure 2):

- *Spatial leave-one-out resampling*: Each individual observation forms a test set.
- *Leave-one-block-out CV*: Individual blocks are left out as test data, i.e., the number of folds equals the number of blocks.
- *CV at the block level*: Blocks are grouped into k partitions, each of which is used as a test fold.

In this context, a block can refer to an arbitrarily shaped spatial (or spatiotemporal) group of observations, not necessarily a rectangular region. A finer distinction can then be made by looking at how the blocks are derived:

- Using a geometry-based approach (rectangular or circular).
- Using an unsupervised clustering approach.
- Using a custom input, i.e., specifying the blocks with an external grouping variable.

In some resampling schemes, separation buffers or guard zones can be imposed to separate the training and test data. Package **mlr3spatiotempcv** currently implements the partitioning methods identified in Table 1. Several of the implemented algorithms are themselves versatile toolboxes with multiple options. Comprehensive and up-to-date information can be found in the package's online documentation (https://mlr3spatiotempcv.mlr-org.com/). The following sections give an overview of most implemented partitioning strategies and their visualization options. The available methods are further discussed in Section 6.

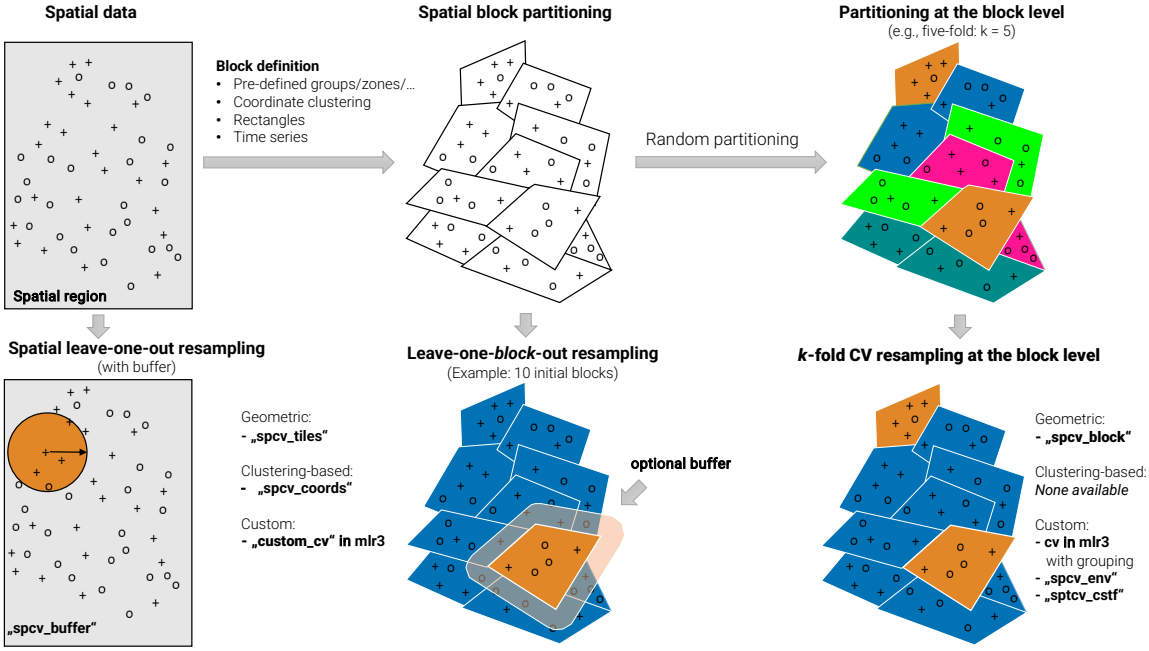

Figure 2: Conceptual overview of various spatial partitioning schemas. Starting from unpartioned spatial observations (top left) either a "spatial block partitioning" or a "spatial leave-one-out resampling" is applied in the first step. A spatial block partitioning can further be turned into a "leave-one-block-out resampling" or a "k fold CV resampling at the block level". The use of a buffer is theoretically possible in any scenario but in practice only offered by specific method implementations. In **mlr3**, statistics are then calculated per fold and aggregated using the mean by default.

Users are encouraged to contribute new or missing spatiotemporal resampling methods directly to **mlr3spatiotempcv**. The already implemented methods can be inspected to get to know the class structure, active bindings and methods.

4.1. Spatial leave-one-out

Spatial leave-one-out methods use individual observations in space as test partitions and apply circular buffer or guard zones around around these test points to enforce a minimum prediction distance. Leave-one-disc-out resampling modifies this approach to leave out circular regions centered at observation points. This group of methods discards observations when applying buffers, meaning that these are not used for training or testing. Depending on the fraction of data being discarded, this might result in a substantial reduction of the sample size and consequently pessimistically biased performance estimates.

Spatial leave-one-out with buffer – `"spcv_buffer"`

Leave-one-out CV with buffer and several adaptations for species distribution modeling (Hijmans, Phillips, Leathwick, and Elith 2023) are implemented in the **blockCV** package as the so-called "buffering" method and integrated into **mlr3spatiotempcv** under the label `"spcv_buffer"`. In species distribution modeling, the response variable can either be recorded

Type	Sub-type	Name	R package	References
Spatial leave-one-out	single point, with buffer	`"spcv_buffer"`	**blockCV** (2)	Ploton *et al.* (2020) Diesing (2020)
	disc, with buffer	`"spcv_disc"`	**sperrorest** (3)	Karasiak *et al.* (2021) Møller *et al.* (2021) Endicott *et al.* (2017)
Leave-one-block-out CV	clustering of coordinates	`"spcv_coords"`*	**sperrorest** (6)	Morera *et al.* (2021) Geiß *et al.* (2017) Wu *et al.* (2020)
	geometric: rectangular	`"spcv_tiles"`	**sperrorest**	Bebber and Butt (2017) Zurell *et al.* (2020) Brenning *et al.* (2015)
	custom	`"custom_cv"`	**mlr3** (0)	–
CV at the block level	geometric: rectangular	`"spcv_block"`	**blockCV** (28)	Jensen *et al.* (2021) Escobar *et al.* (2021) Stewart *et al.* (2021)
	custom	`"cv"` with grouping	**mlr3** (0)	–
	clustering in feature space	`"spcv_env"`*	**blockCV** (1)	Morera *et al.* (2021)
	clustering of coordinates + aggregation	`"spcv_knndm"`*	**CAST** (1)	Ludwig *et al.* (2023)
Spatiotemp. CV	custom	`"sptcv_cstf"`*	**CAST** (6)	Gao *et al.* (2019) Reitz *et al.* (2021) Egli and Höpke (2020)

Table 1: Available spatiotemporal resampling methods in the **mlr3** ecosystem. The "Name" column shows the **mlr3** method name as found in the `mlr3::mlr_resamplings` dictionary. The count in brackets after the package name represents the number of studies that were found having used this resampling technique until May 2021. For each method, up to three randomly selected references were added to the table. Methods suffixed with a * in the "Name" column have been reimplemented in **mlr3spatiotempcv** for efficiency or to reduce upstream dependencies.

as presence/absence data or as presence/background information; both options are supported by this implementation. By default, the dataset contains confirmed presence and confirmed absence observations, i.e., locations where a species was observed and not observed, respectively, and therefore spatial LOO CV in its usual sense can be carried out. Figure 3 shows the first test fold generated with this method for presence/absence data with a buffer distance of 1000 m.

```
R> library("mlr3")
R> library("mlr3spatiotempcv")
R> task <- tsk("ecuador")
R> rsmp_buffer <- rsmp("spcv_buffer", theRange = 1000)
R> rsmp_buffer

<ResamplingSpCVBuffer>: Spatial buffering resampling
```

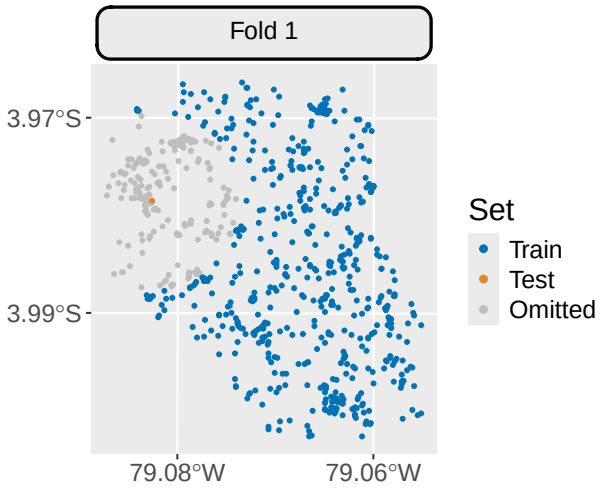

Figure 3: Visualization of the spatial buffering method from package **blockCV** (method `"spcv_buffer"` in **mlr3spatiotempcv**). The buffer distance is 1000 m.

```
* Iterations: 0
* Instantiated: FALSE
* Parameters: theRange=1000
```

```
R> autoplot(rsmp_buffer, size = 0.8, task = task, fold_id = 1,
+    show_omitted = TRUE)
```

In the presence/background (or presence-only) situation, in contrast, only presence observations are recorded, and all other locations within the study area are referred to as background and considered as pseudo-absences. Presence/background modeling can be enabled with the argument `spDataType = "PB"`. In this situation, the method constructs test folds that are centered at the recorded presence locations, offering two different modes of operation. With `addBG = TRUE` (the default), all background points with a distance of `theRange` around a test (presence) point are included in the test fold as absence data; note that in this case, there is no separation buffer between training and test samples. The `addBG = FALSE` setting, in contrast, for which no background data is added to the test fold, then contains only one (presence) observation, and only the data at a distance of `theRange` or greater are included in the training sample, including background data from these areas.

The application of LOO methods can be computationally expensive since the method cycles through the entire dataset and fits one model for each test fold.

Leave-one-disc-out with optional buffer – `"spcv_disc"`

Leave-one-disc-out resampling from package **sperrorest** defines circular test sets that are centered at sample locations, and optionally excludes a buffer zone from the remaining training data. It thus ensures that a minimum separation distance between training and test data is maintained. The number of discs is specified by the `folds` argument, which defaults to the sample size n. Sample locations are selected randomly when `folds` is smaller than n; it is optionally possible to sample with replacement (`replace = TRUE`). Leave-one-disc-out resampling becomes LOO CV and when each observation is at a unique location.

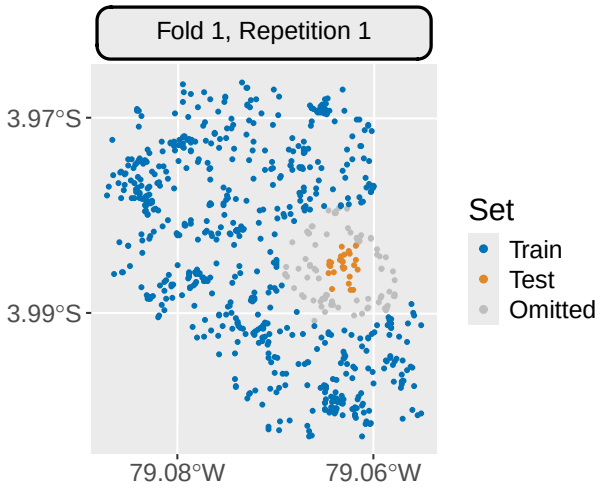

Figure 4: Visualization of one training set / test set combination generated with the leave-one-disc-out method from package **sperrorest** (method `"spcv_disc"` in **mlr3spatiotempcv**). The disc has a radius of 300 m and is surrounded by a 400 m buffer.

It should be noted that the resampled discs will potentially overlap. Strictly speaking, this straightforward extension of spatial LOO does therefore not establish a disjoint partitioning as used for CV resampling in the traditional sense.

```
R> rsmp_disc <- rsmp("spcv_disc", folds = 100, radius = 300L, buffer = 400L)
R> rsmp_disc

<ResamplingSpCVDisc>: Repeated Spatial 'disc' resampling
* Iterations: 100
* Instantiated: FALSE
* Parameters: folds=100, radius=300, buffer=400
```

Figure 4 is produced by the command:

```
R> autoplot(rsmp_disc, size = 0.8, task = task, fold_id = 1,
+    show_omitted = TRUE)
```

4.2. Leave-one-block-out cross-validation

Leave-one-block-out resampling methods partition the dataset spatially in order to use each of the resulting partitions as a CV test fold.

Clustering-based: Using coordinates – `"spcv_coords"`

Cluster analysis provides a flexible approach to creating irregularly shaped spatial blocks for spatial resampling. Numerous techniques are available that can potentially be applied to the spatial coordinates of observations, to the features, or to a combination of both. In spatial model assessment, the focus has been on coordinate-based clustering, and specifically on

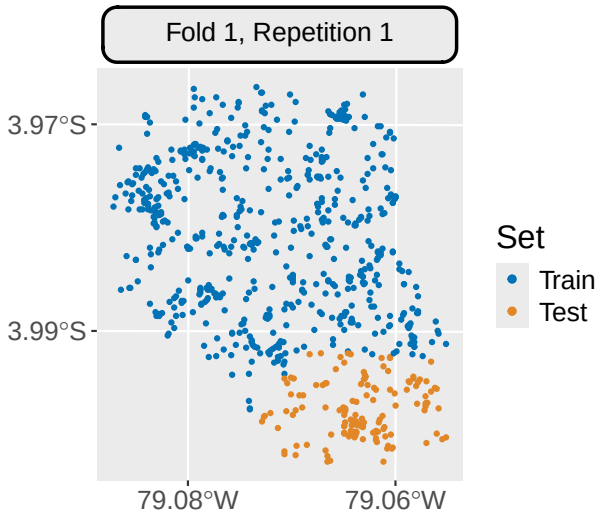

Figure 5: Leave-one-block-out CV based on k means clustering of the coordinates as implemented in package **sperrorest** (method `"spcv_coords"` in **mlr3spatiotempcv**).

leave-one-block-out resampling with blocks created by k-means clustering of the coordinates (Ruß and Brenning 2010).

Coordinate-based clustering for spatial CV (Ruß and Brenning 2010; Brenning 2012) as implemented in package **sperrorest** uses the coordinates of all observations to create clusters in the spatial domain with the help of the k-means clustering algorithm. This can be regarded as a leave-one-block-out resampling method, or as a k-fold CV in which each test set is a spatial cluster. This method is referred to as `"spcv_coords"` in **mlr3spatiotempcv**.

The coordinate-based clustering approach is very versatile as it adapts to irregularly-shaped study areas and ensures that exactly k partitions are created, which are usually of very similar size when the sample locations are spread out evenly. Nevertheless, despite the random selection of initial cluster centers, repeated partitionings may in some cases be nearly identical. Also, k-means clustering may be less suitable for data sets with pre-existing clusters of points and/or with isolated, distant sample locations. When distinct clusters of points are present, as in multi-level sampling, it may be better to define clusters using a factor variable (see method `"custom_cv"` in Section 4.2). The output of the following code chunk can be seen in Figure 5.

```
R> rsmp_coords <- rsmp("spcv_coords", folds = 5)
R> autoplot(rsmp_coords, size = 0.8, fold_id = 1, task = task)
```

Geometric: Using rectangular blocks – `"spcv_tiles"`

Leave-one-tile-out resampling is implemented in the `"spcv_tiles"` method imported from package **sperrorest**. It uses rectangular blocks that can be rotated (argument `rotation`), and a minimum number or fraction of observations per block can optionally be achieved by iteratively merging small blocks into adjacent blocks (argument `reassign` in conjunction with `min_n` or `min_frac`). Block size or number is specified via the argument `dsplit` or `nsplit`, respectively, and square blocks can be obtained with a single (or two identical) `dsplit` value(s).

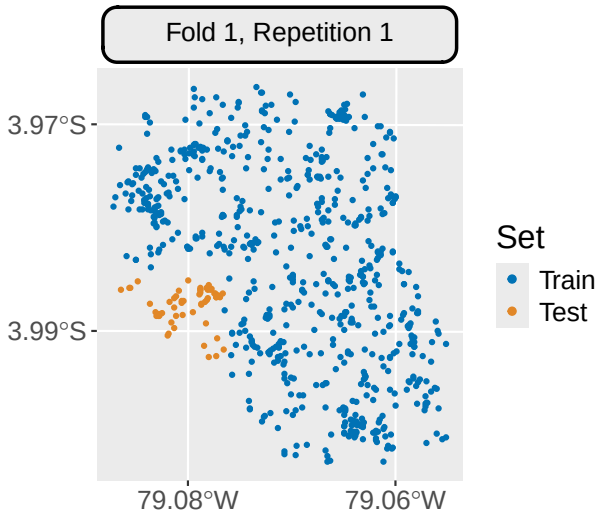

Figure 6: Leave-one-block-out resampling from package **sperrorest** (method `"spcv_tiles"` in package **mlr3spatiotempcv** with argument `nsplit = c(3, 4)` indicating the number of rows and columns).

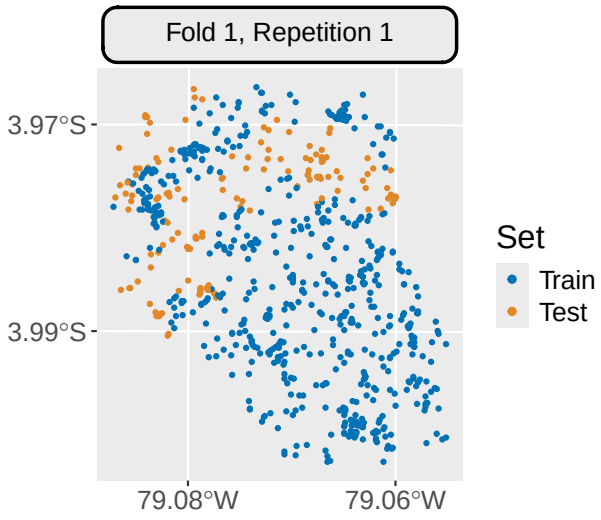

Figure 7: Leave-one-level-out (custom) resampling from package **mlr3** (method `"custom_cv"`). A factor variable is used to define all partitions.

Note that the actual number of folds obtained may be smaller than `nsplit[1] * nsplit[2]` (or smaller than what would be expected based on `dsplit`) since some blocks may be empty or (optionally) merged into adjacent folds. In the example, there are only eleven folds instead of twelve because the southwestern part of the study area's bounding box does not contain observations (Figure 6).

```
R> library("sperrorest")
R> rsmp_tiles <- rsmp("spcv_tiles", nsplit = c(3L, 4L))
R> autoplot(rsmp_tiles, size = 0.8, fold_id = 1, task = task)
```

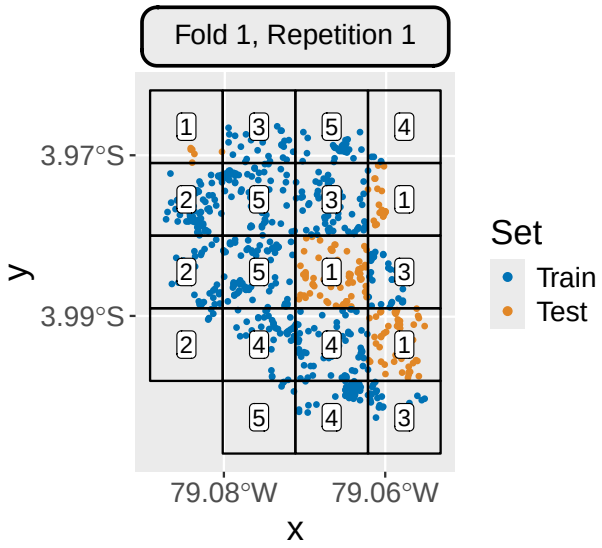

Figure 8: Random resampling of square spatial blocks using the implementation in package **blockCV** (method `"spcv_block"` with option `selection = "random"` in **mlr3spatiotempcv**). The size of the squares is 1000 m, and four out of the 19 blocks were assigned to the test partition.

Custom: `"custom_cv"` in ***mlr3***

Support for user-defined partitioning strategies is built into **mlr3** directly. In this so-called "custom CV", users supply a factor variable, each level of which defines a partition. The factor variable can either be specified through a factor vector of the same length as number of observations, or by passing the name of a feature within the task (argument `col`). The following simple example (taken from `sperrorest::partition_factor()`) creates altitudinal zones that define the spatial partitions (see Figure 7 for the resulting plot).

```
R> breaks <- quantile(task$data()$dem, seq(0, 1, length = 6))
R> zclass <- cut(task$data()$dem, breaks, include.lowest = TRUE)
R> rsmp_custom <- rsmp("custom_cv")
R> rsmp_custom$instantiate(task, f = zclass)
R> autoplot(rsmp_custom, size = 0.8, task = task, fold_id = 1)
```

4.3. Cross-validation at the block level

Methods which operate at the block level first group the observations into blocks and then combine these blocks into CV partitions. In k-fold CV resampling at the block level, there are therefore k partitions, each consisting of $1/k$th of the blocks. The special case in which k equals the number of blocks, CV at the block level simply becomes leave-one-block-out CV, for which dedicated implementations exist (see Section 4.2).

Geometric: Using rectangular blocks – ***`"spcv_block"`***

The `"spcv_block"` method from package **blockCV** supports both random and systematic resampling of square blocks with argument `selection = "random"` and `"systematic"`, respectively; see Figure 8 and Figure 9. There are additional options for modeling presence-only

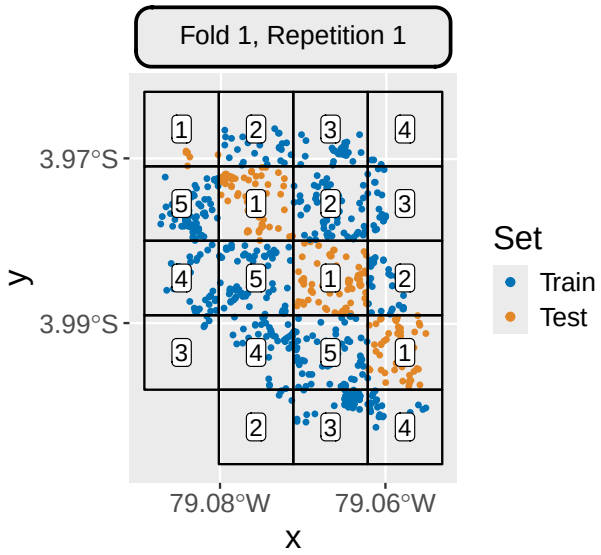

Figure 9: Sytematic resampling of square spatial blocks using the implementation in package **blockCV** (method `"spcv_block"` with option `selection = "systematic"` in **mlr3spatiotempcv**). The size of the squares is 1000 m, and four out of the 19 blocks were assigned to this test sample.

data, which is a typical use case in species distribution modeling. Users can furthermore supply a user-defined polygon via argument `rasterLayer` with predefined blocking zones.

The size of the square blocks (in meters) are determined by the `range` argument. Rectangular blocks can be created by specifying the number of desired rows and columns (arguments `rows` and `cols`). Due to the non-trivial specification of argument `range`, package **blockCV** provides the helper functions `spatialAutoRange()` and `rangeExplorer()` to conduct a data-driven estimation of the distance at which the spatial autocorrelation within the data levels off. According to the package authors, this estimate should then be used for argument `range` to have a sensible value for the block sizes created in method `"spcv_block"`.

It should be noted that rectangular partitioning can be problematic in irregularly shaped study areas as shown in Figure 8, where some of the resulting partitions may contain substantially fewer observations than others.

```
R> rsmp_block_random <- rsmp("spcv_block", range = 1000, folds = 5)
R> autoplot(rsmp_block_random, size = 0.8, fold_id = 1, task = task,
+    show_blocks = TRUE, show_labels = TRUE, label_size = 4)
```

In systematic resampling, the blocks are numbered row by row, and blocks $i + j \cdot$`folds` are assigned to fold i (see Figure 9). This may create undesired patterns when the number of columns is equal to or a multiple of the number of folds.

```
R> rsmp_block_systematic <- rsmp("spcv_block",
+    range = 1000, folds = 5, selection = "systematic")
R> autoplot(rsmp_block_systematic, size = 0.8, fold_id = 1, task = task,
+    show_blocks = TRUE, show_labels = TRUE, label_size = 4)
```

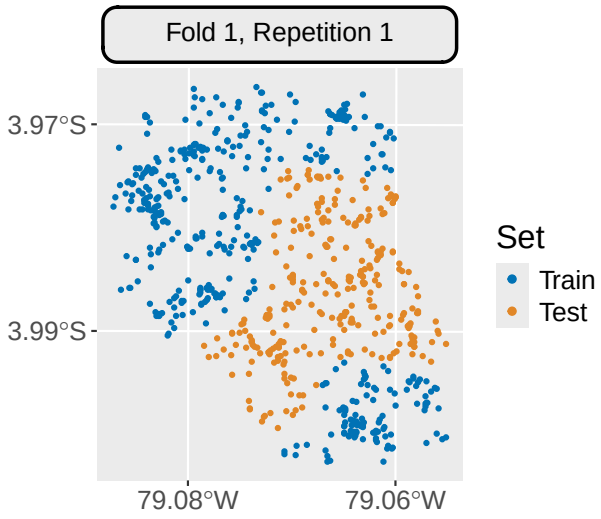

Figure 10: Cross-validation at the block level including predefined groups from package **mlr3** (method `"cv"`). A factor variable is used to define the grouping. Each class is either assigned to the test or training set.

Checkerboard partitioning is a special case of a systematic block partitioning (`selection = "checkerboard"`) which is why we omitted a practical example for this option. It inherently supports only two folds, making it less appealing than the more commonly used five- or ten-fold resampling, which achieve larger training set sizes.

Custom: `"cv"` with grouping in **mlr3**

Although the `"cv"` resampling strategy in **mlr3** performs random, non-spatial partitioning by default, it can also be used for CV at the block level. This is achieved by specifying the `"group"` column role in a **mlr3** 'Task' object, which uses the factor levels as blocks. A complete group or block of observations is therefore assigned to a specific partition, which consequently honors the grouping structure.

In contrast to geometric or clustering-based blocks, the spatial or temporal location is not used explicitly, but rather implicitly through the spatial or spatiotemporal footprint of each user-defined block. Studies which have applied custom grouping approaches include Meyer *et al.* (2018), Anderson *et al.* (2005), Kasurak *et al.* (2011).

The following example uses k-means clustering to generate classes that are used as blocks. To underline the honoring of the groups, a number of groups (eight) that is not a multiple of the number of folds (three) was chosen. The test sets in the first and second folds are therefore composed of three groups while the third one holds two groups. The resulting plot is presented in Figure 10.

```
R> task_cv <- tsk("ecuador")
R> group <- as.factor(kmeans(task$coordinates(), 8)$cluster)
R> task_cv$cbind(data.frame("group" = group))
R> task_cv$set_col_roles("group", roles = "group")
R> rsmp_cv_group <- rsmp("cv", folds = 3)$instantiate(task_cv)
R> autoplot(rsmp_cv_group, size = 0.8, task = task_cv, fold_id = 1)
```

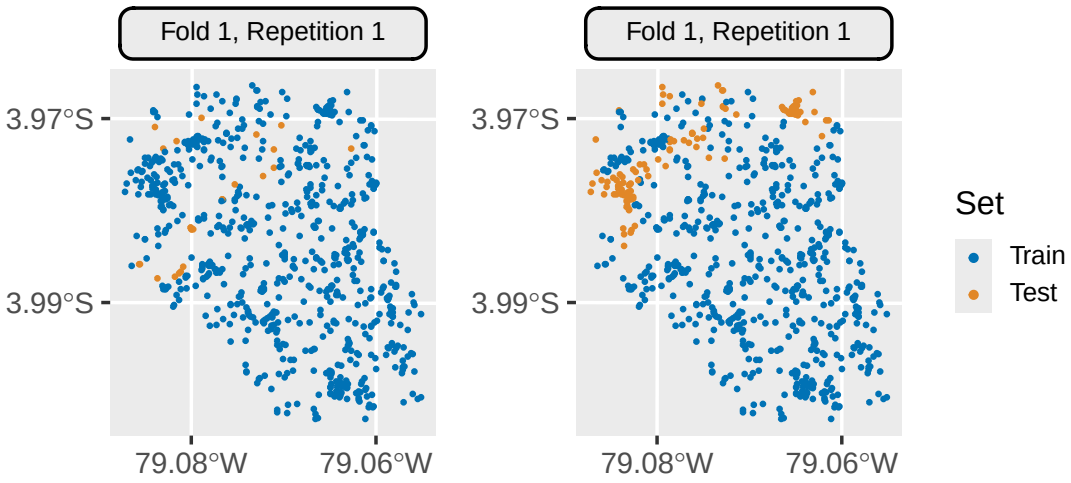

Figure 11: Environmental leave-one-block-out CV from package **blockCV** using one (left, `"distdeforest"`) and two (right, `"distdeforest"` and `"slope"`) predictors to define blocks in the feature space. Due to feature space clustering observations are not (necessarily) grouped in the spatial domain.

Clustering: Using feature-based clustering – `"spcv_env"`

The last method from the **blockCV** package, referred to as "environmental blocking" (Roberts *et al.* 2017), makes use of k means clustering (Hartigan and Wong 1979) in a possibly multivariate space to define blocks for resampling at the block level. The user can select one or multiple numeric features via argument `feature` from which the clusters are created. Hereby, k-means will use Euclidean distance. To avoid a potential bias introduced by features with high variance when selecting multiple features, all features are standardized by default.

In the following example, the observations are clustered based on the feature "distance to deforestation" (left sub-figure of Figure 11), which results in a distance-based zonification. This method also allows to use multiple features for clustering. The right sub-figure of Figure 11 shows the outcome when using "distance to deforestation" and "slope angle".

```
R> rsmp_env <- rsmp("spcv_env", features = "distdeforest", folds = 5)
R> rsmp_env_multi <- rsmp("spcv_env", features = c("distdeforest", "slope"),
+    folds = 5)
R> plot_env_single <- autoplot(rsmp_env, size = 0.3, fold_id = 1,
+    task = task)
R> plot_env_multi = autoplot(rsmp_env_multi, size = 0.3, fold_id = 1,
+    task = task)
R> library("patchwork")
R> plot_env_single + plot_env_multi
```

Nearest neighbour distance matching – `"spcv_knndm"`

The `spcv_knndm` method from the **CAST** package uses a two-stage approach: first groups of observations are created using nearest neighbour clustering. These are then merged into k clusters with the aim to minimize the differences between the empirical nearest neighbour

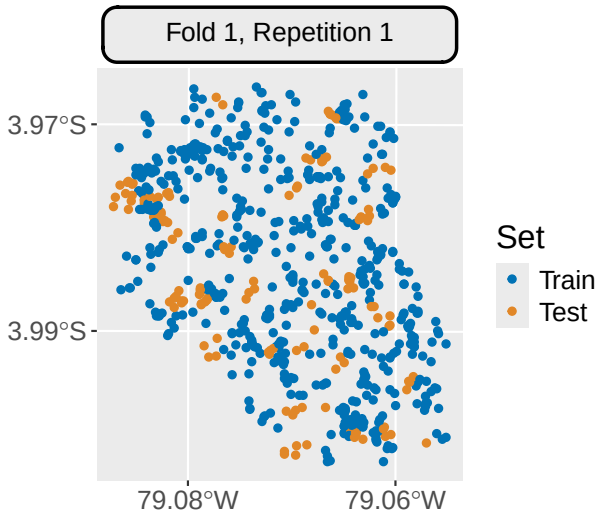

Figure 12: Random resampling of square spatial blocks using the implementation in package **CAST** (method `"spcv_knndm"`). Hierarchical clustering has been used together with the `"ward.D2"` link function and a target of five folds.

distribution function of training and prediction points and training and test data during CV using the Wasserstein W statistic (Linnenbrink, Milà, Ludwig, and Meyer 2023).

The main goal is to ensure similar conditions during CV compared to the actual prediction scenario of the trained model. The k-fold variant is an extension of the LOOCV variant proposed initially by Milà, Mateu, Pebesma, and Meyer (2022) and more suited for large datasets.

The following example uses the default settings of the implementation in the **CAST** package. The method requires a prediction area as an `sf` polygon or point object. In this case the former was used and passed as an argument to the `"spcv_knndm"` method. The resulting plot is presented in Figure 12.

```
R> points <- sf::st_as_sf(task$coordinates(), crs = task$crs,
+    coords = c("x", "y"))
R> modeldomain <- sf::st_as_sfc(sf::st_bbox(points))
R> rsmp_knndm <- rsmp("spcv_knndm", modeldomain = modeldomain, folds = 5)
R> autoplot(rsmp_knndm, size = 0.8, fold_id = 1, task = task,
+    label_size = 4)
```

4.4. Cross-validation for spatiotemporal data

Some of the implemented resampling methods operate in multiple dimensions, i.e., in space, time, or space–time. In this section, only examples of these methods in the spatiotemporal domain will be shown. For their application in lower dimensions, usually only either the space or time coordinates need to be omitted from the user input.

Custom: "Leave-location-and-time-out" and related methods

Meyer *et al.* (2018) proposed a spatiotemporal resampling method in which a test set is

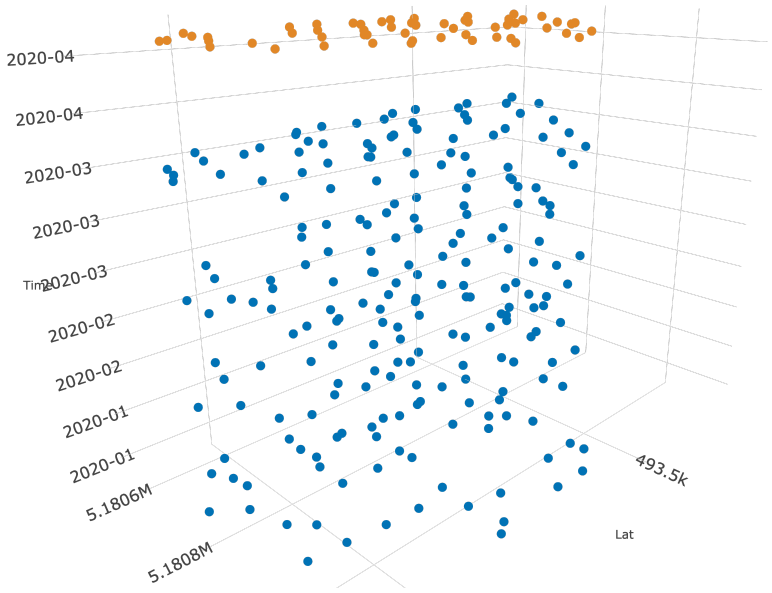

Figure 13: Perspective plot of `"leave-time-out"` CV from package **CAST** (method `"sptcv_cstf"` and column role `"time" = "Date"`). Only five folds and five time points were used in this example. Note that the blue dots correspond to five discrete time levels, which appear as a point cloud due to the viewing angle.

selected and all observations that correspond to the same location or time point are omitted from the training sample. This method is referred to as "leave-location-and-time-out" (LLTO) in package **CAST**. Additional methods that resample in the temporal and spatial domain only are named "leave-time-out" (LTO) and "leave-location-out" (LLO), respectively. Note that despite their names, LLTO, LTO and LLO are conceptually not leave-*one*-out methods as they place a certain fraction of observations in the test set, as in ordinary CV. Also, LTO and LLO are conceptually similar to **mlr3**'s `"cv"` method with a custom grouping as they perform a CV at the block level using a grouping structure defined by time points (LTO) and locations (i.e., time series; LLO).

In this section the `cookfarm` dataset is used as an example because it has a temporal dimension identified by the variable `Date`.

`mlr3spatiotempcv::autoplot()` supports two visualization types for spatiotemporal methods which can be selected via the logical argument `plot3D`. The heavy lifting of the 3D visualization (i.e., 2D + time) option is done via package **plotly**. Because a dynamic image cannot be included in this manuscript, static versions, which can be generated by setting `static_image = TRUE`, are shown (see for example Figure 13).

CV at the time-point level: "leave-time-out" (LTO) – `"sptcv_cstf"` In the LTO method, the time points are resampled into the desired number of folds. In the terminology used in this work, this can be referred to as resampling at the level of time points, which effectively define blocks. Thus, observations from the same time point are jointly sampled into the same test (or training) fold, with no constraints on the temporal distance between the sampled time points. This method does therefore not implement block CV in the sense of the time series literature.

In the `cookfarm_mlr3` example dataset, the `Date` variable was reduced to five unique levels for better visualization, and then used to create a spatiotemporal regression task in **mlr3spatiotempcv** (Figure 13). In `autoplot()`, a stratified sample based on the partitions is taken to reduce the number of points plotted.

```
R> data <- cookfarm_mlr3
R> set.seed(42)
R> data$Date <- sample(rep(c("2020-01-01", "2020-02-01", "2020-03-01",
+    "2020-04-01", "2020-05-01"), times = 1, each = 35768))
R> task_spt <- as_task_regr_st(data, id = "cookfarm", target = "PHIHOX",
+    coordinate_names = c("x", "y"), coords_as_features = FALSE, crs = 26911)
R> task_spt$set_col_roles("Date", roles = "time")
R> rsmp_cstf_time <- rsmp("sptcv_cstf", folds = 5)
R> p_lto <- autoplot(rsmp_cstf_time, fold_id = 5, task = task_spt,
+    plot3D = TRUE, point_size = 6, axis_label_fontsize = 15,
+    sample_fold_n = 3000L)
R> p_lto_print <- plotly::layout(p_lto,
+    scene = list(camera = list(eye = list(z = 0.58))),
+    showlegend = FALSE, title = "",
+    margin = list(l = 0, b = 0, r = 0, t = 0))
R> plotly::save_image(p_lto_print, "lto.pdf", scale = 2, width = 1100,
+    height = 800)
```

CV at the location level: "Leave-location-out" (LLO) – `"sptcv_cstf"` In contrast to LTO, the LLO method randomly resamples locations that may, for example, correspond to time series. The sampled locations form the test partition while the temporal information is ignored (Figure 14). Unlike spatial CV methods that are based on geometric regions or the clustering of coordinates, the sampled test locations include no particular spatial relationship.

To tell the resampling method to use the `"space"` column for partitioning, the `"time"` column needs to be unset and the `"space"` column defined. Because the temporal variable `Date` is not in use in this scenario, `autoplot()` needs to be instructed explicitily to use it for 3D plotting via argument `plot_time_var`.

```
R> task_spt$col_roles$time <- character()
R> task_spt$set_col_roles("SOURCEID", roles = "space")
R> rsmp_cstf_loc <- rsmp("sptcv_cstf", folds = 5)
R> p_llo <- autoplot(rsmp_cstf_loc, fold_id = 5, task = task_spt,
+    point_size = 6, axis_label_fontsize = 15, plot3D = TRUE,
+    plot_time_var = "Date", sample_fold_n = 3000L)
R> p_llo_print <- plotly::layout(p_llo,
+    scene = list(camera = list(eye = list(z = 2.5, x = -0.1, y = -0.1))),
+    showlegend = FALSE, title = "", polar = TRUE,
+    margin = list(l = 0, b = 0, r = 0, t = 0))
R> plotly::save_image(p_llo_print, "llo.pdf", scale = 2, width = 1000,
+    height = 800)
```

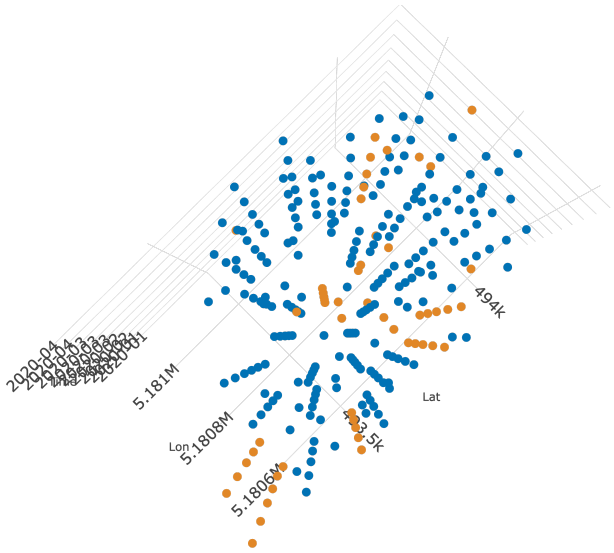

Figure 14: Birds-eye view of `"leave-location-out"` CV from package **CAST** (method `"sptcv_cstf"` and column role `"space" = "SOURCEID"`). The readers viewpoint is located on top of the figure, looking down. The transects are individual locations being left out across multiple time steps.

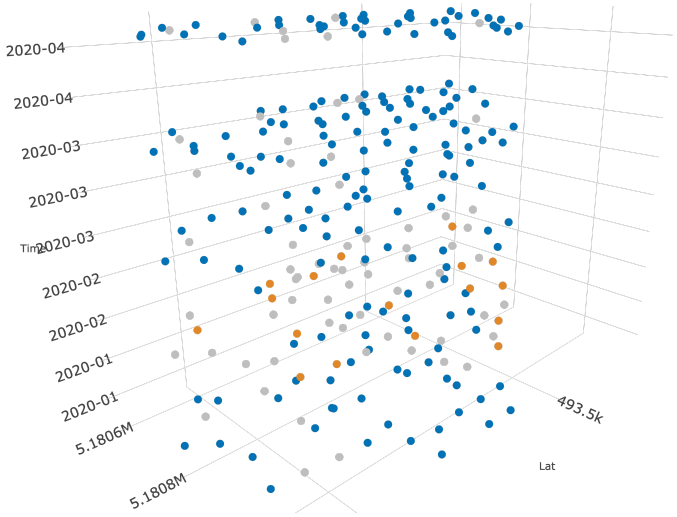

Figure 15: Perspective plot of `"leave-location-and-time-out"` CV from package **CAST** (method `"sptcv_cstf"` and column roles `"time" = "Date"` and `"space" = "SOURCEID"`). The grey points are excluded from both the training and the test set in this example.

"Leave-location-and-time-out" (LLTO) – `"sptcv_cstf"` In LLTO, a test set is first randomly sampled from the data set, and then all observations that correspond to the same location or time point are omitted from the training sample (Figure 15). LLTO resampling mimics the situation where a model is trained on time series data from a number of locations and time points, and used to predict the time series at other locations and time points that are not included in the training sample.

Conceptually, LLTO applies zero-distance buffering in both space and time: The buffer zones consist of all observations whose distance to the test sample in either space or time equals zero. In a mathematical sense, however, this buffering is not based on a valid metric (or distance function) in three-dimensional space (2D + time) as neither the identity of detectability nor the triangle inequality are satisfied by the underlying combined "distance" measure. Also note that LLTO does not "combine" LTO with LLO, as neither of these applies a buffer zone.

The `"spcv_cstf"` methods LLO and LTO (with only one of `space_var` or `time_var` set) require a variable in the dataset which should be used for grouping. The specification of the variable(s) which should be used for a spatial, temporal or spatiotemporal grouping is not trivial because the final partitioning should, in the optimal case, ensure that the selected groups inherit substantial autocorrelation within themselves and simultaneously differ substantially from other partitions. Also, if the selected variable contains too many groups, the difference within train/test splits may become undesirably high and tend towards a LOO CV (Meyer *et al.* 2018).

```
R> task_spt$set_col_roles("SOURCEID", roles = "space")
R> task_spt$set_col_roles("Date", roles = "time")
R> rsmp_cstf_time_loc <- rsmp("sptcv_cstf", folds = 5)
R> p_lto <- autoplot(rsmp_cstf_time_loc, point_size = 6,
+    axis_label_fontsize = 15, fold_id = 4, task = task_spt, plot3D = TRUE,
+    show_omitted = TRUE, sample_fold_n = 3000L)
R> p_lto_print <- plotly::layout(p_lto,
+    scene = list(camera = list(eye = list(z = 0.58))),
+    showlegend = FALSE, title = "",
+    margin = list(l = 0, b = 0, r = 0, t = 0))
R> plotly::save_image(p_lto_print, "llto.pdf", scale = 2, width = 1100,
+    height = 800)
```

5. Example: Comparing spatial and non-spatial CV

A case study is used to demonstrate the application of spatial and non-spatial resampling techniques for model assessment in **mlr3spatiotempcv**. The objective of landslide susceptibility modeling is to predict how prone to landslide initiation a location is. Models are fitted to historical landslide occurrences, but they need to learn generalizable relationships between predisposing variables and the response as opposed to perfectly reproducing or memorizing the historical distribution. This binary classification task on landslides in Ecuador (Muenchow, Brenning, and Richter 2012) is available as a built-in task via `tsk("ecuador")`, but is generated from the learning sample in this example. Random forest is used as a classifier, and the area under the ROC curve (AUROC) as the performance measure.

Spatial CV is implemented in the form of leave-one-block-out CV using coordinate-based k-means clustering to generate irregularly shaped blocks of roughly equal size. This approach is better suited for the irregular shape of the present study area than a rectangular partitioning. Figure 16 and Figure 17 show the contrasting distributions of training and test samples. For demonstration purposes only four CV folds and two repetitions are used.

Besides the practical example shown below, additional tutorials covering **mlr3** use cases can be found in the **mlr3** gallery (https://mlr-org.com/gallery/).

5.1. Task preparation

In **mlr3**, machine-learning tasks with their respective dataset and response variable are represented by objects of class 'Task'. **mlr3spatiotempcv**'s spatial and spatiotemporal machine-learning tasks are also derived from this superclass. Specifically, the 'TaskClassifST' and 'TaskRegrST' classes for classification and regression tasks require several additional arguments that must be passed as a named list using the `extra_args` argument:

- `coordinate_names`: Names of the features that represent the spatial coordinates. This is automatically inferred when an '`sf`' object is passed.
- `coords_as_features`: Whether the coordinates should be used as features; by default they are not.
- `crs`: The coordinate reference system of the data as a PROJ string or EPSG code in the format `ESPG:<code>`.

At first all necessary R packages are loaded and a lower verbosity is set to keep the output tidy. A random-number seed is set for reproducibility. Package **lgr** (Fleck 2022) is employed as a logging framework.

```
R> library("mlr3")
R> library("mlr3spatiotempcv")
R> lgr::get_logger("bbotk")$set_threshold("warn")
R> lgr::get_logger("mlr3")$set_threshold("warn")
R> set.seed(42)
```

The task `"ecuador"` is available as an example task in **mlr3spatiotempcv** through the command `tsk("ecuador")`. To create it manually from a `data.frame` named `ecuador`, one would do:

```
R> data("ecuador", package = "mlr3spatiotempcv")
R> task <- as_task_classif_st(ecuador, target = "slides", positive = "TRUE",
+    coordinate_names = c("x", "y"), coords_as_features = FALSE,
+    crs = "EPSG:32717")
```

5.2. Model preparation

Next, the random forest learner (`"classif.ranger"`) is initialized with default hyperparameters and the prediction type is set to `"probability"` because the model is used for soft classification. A set of commonly used learners is available in package **mlr3learners** (Lang, Au, Coors, Schratz, and Becker 2024a), including the random forest implementation of **ranger** (Wright and Ziegler 2017).

```
R> library("mlr3learners")
R> learner <- lrn("classif.ranger", predict_type = "prob")
```

5.3. Non-spatial cross-validation

To define a resampling strategy, the `rsmp()` function is used to generate a resampling object using four folds and two repetitions following a random sampling logic `"repeated_cv"`).

Next, the created resampling object `rsmp_nsp` is passed to the `resample()` function together with the task and learner objects created earlier to execute the model assessment. This is the actual, potentially time-consuming CV estimation. With the present settings, eight random forest classifiers are fitted and evaluated in this step – one model fitted on each CV training set.

Model performances are calculated from the CV predictions using the AUROC (`"classif.auc"` in **mlr3** notation).

```
R> rsmp_nsp <- rsmp("repeated_cv", folds = 4, repeats = 2)
R> rr_nsp <- resample(task = task, learner = learner, resampling = rsmp_nsp)
R> rr_nsp$aggregate(measures = msr("classif.auc"))

classif.auc
  0.7600664
```

5.4. Spatial cross-validation via coordinate-based clustering

The model assessment is now repeated again using spatial CV resampling, for which the only required change is to replace `"repeated_cv"` with `"repeated_spcv_coords"`.

```
R> rsmp_sp <- rsmp("repeated_spcv_coords", folds = 4, repeats = 2)
R> rr_sp <- resample(task = task, learner = learner, resampling = rsmp_sp)
R> rr_sp$aggregate(measures = msr("classif.auc"))

classif.auc
  0.6100402
```

5.5. Visualization of CV partitions

Finally, we visualize (two of) the partitions that were used during performance estimation by making use of the generic `autoplot()` function in package **mlr3spatiotempcv** (Figure 16 and Figure 17).

```
R> autoplot(rsmp_sp, task, fold_id = 1:2, size = 0.3)
R> autoplot(rsmp_nsp, task, fold_id = 1:2, size = 0.3)
```

5.6. Interpretation

If one takes a closer look at the results, the non-spatial CV estimate of AUC (0.76) is substantially higher compared to the spatial CV estimate of 0.64. Since test points in non-spatial CV may be from the same slopes or even the same landslides as the training data, the non-spatial CV result should/can be considered as an over-optimistic estimate of the model's ability to predict the susceptibility to "new" landslides. Spatial CV, in contrast, provides a bias-reduced measure of a model's ability to generalize from the training sample – in this case study, from the specific hillslopes and historical landslides in the training sample. It is also expected that

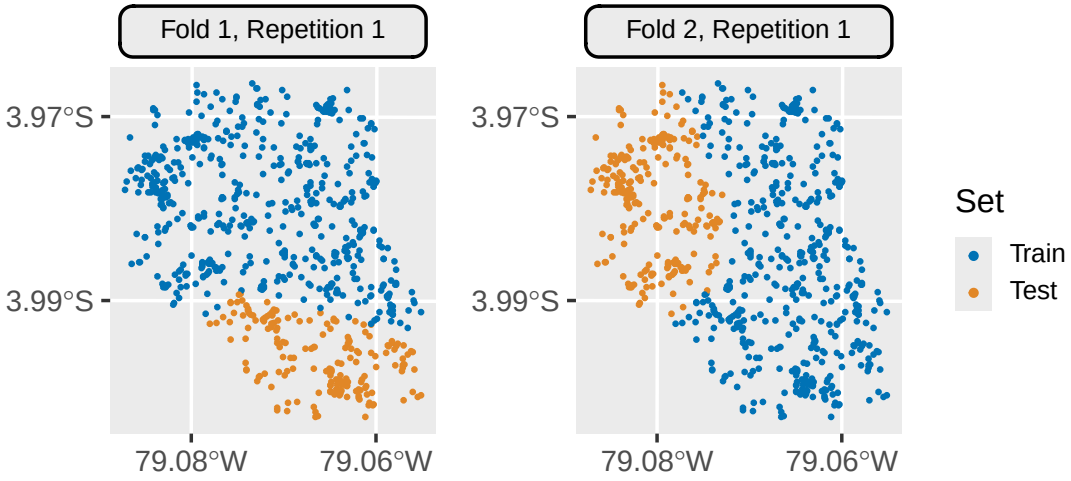

Figure 16: Spatial leave-one-block-out partitioning using coordinate-based clustering to create roughly equally sized polygonal blocks. Due to space limitations only the first two folds of the first repetition are shown.

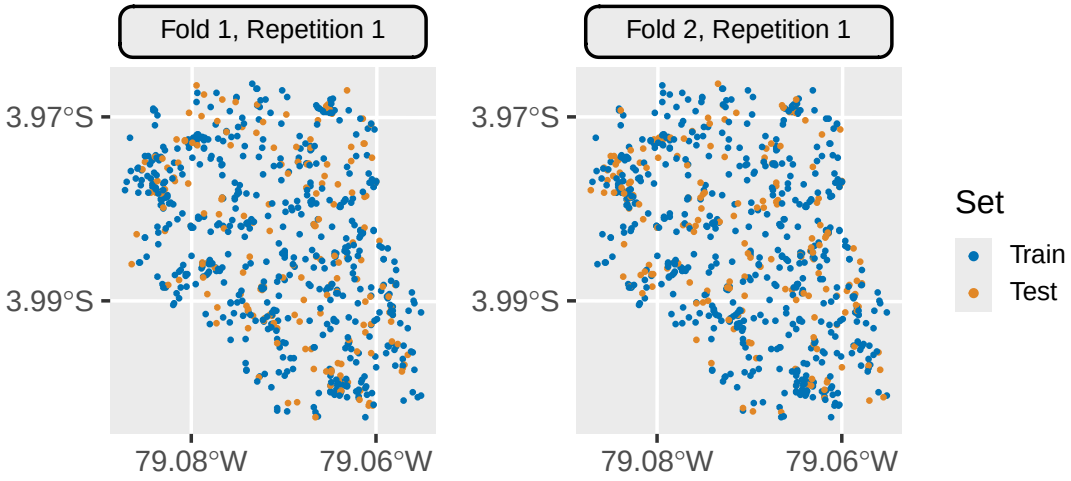

Figure 17: Random (non-spatial) four-fold CV partitioning. Only the first two folds of the first repetition are shown.

spatial CV results better represent the model's transferability to geologically and topographically similar areas adjacent to the training area. Yet, it must be kept in mind that using spatial CV might also lead to a pessimistic estimate of the model's predictive performance if large parts of the study area are being left out during training, depending on the number of CV folds being used. The magnitude of the difference between spatial and non-spatial CV estimates may depend on the dataset, the strength on spatial or spatiotemporal autocorrelation, and the learner itself. Algorithms with a higher tendency to overfit to the training set will tend to have a larger spread in such scenarios.

6. Discussion

6.1. Choosing a resampling method for model assessment

The question of which resampling method should be chosen for a prediction task and dataset at hand comes up regularly in practice. Even though there is and most likely will be no definitive answer to this question, we would like to give some guidance in this section to help find an appropriate method. As a general rule, we recommend to use a resampling scheme that (1) mimics the predictive situation in which the model will be applied operationally, and (2) is consistent with the structure of the data. Both aspects are outlined in this section, starting with two concrete modeling scenarios.

Although the case study example in Section 5 used the `"spcv_coords"` method for coordinate-based clustering, other methods would be suitable as well. In this particular application setting, we want to assess how well the model generalized from the concrete set of historical landslide occurrences, which is why we ensured that training and test sets contain different, "new" hillslopes and landslides. Coordinate-based clustering is particularly appealing in this setting because of its ability to adapt to the irregularly shaped study area of this example. Resampling at the level of sub-catchments could have been a viable alternative approach that can be implemented using custom resampling (`"custom_cv"` method); however, this may result in less balanced sizes of test sets as catchment sizes may vary. When the timing of landslides is known (event-based inventories) or multiple inventories have been compiled for different time points, it can also be recommendable to additionally sample training and test data from different time points, as with the LLTO and LTO (Meyer *et al.* 2018) or similar methods (Brenning 2005).

In other scenarios, such as when predicting crop types across monocultural fields (see Section 1) (Peña and Brenning 2015), it makes sense to group observations into blocks based on previously known boundaries and resample at the polygon level (`"cv"` method with grouping). If, in contrast, the objective is to apply the model to an adjacent agricultural region (e.g., adjacent county) where the same crop types are present, it may be advisable to use coordinate-based clustering (`"spcv_coords"` method) to obtain larger, contiguous test regions.

In summary, there are various factors that may be considered in judging the suitability of a resampling method:

- Will the model be applied to predict "new" outcomes at near or more distant spatial locations?
- Will it be applied to predict into the future, or hindcast gaps between spatiotemporal observations in the past?
- Is it necessary to impose a separation distance or prediction horizon as a spatial or temporal buffer between training and prediction locations?
- How densely are the observations distributed in space and time? Are they more densely distributed than the intended spatial or temporal prediction distance?
- Is the data naturally grouped, e.g., because of the spatial extent of the studied objects, or as a consequence of multi-level (cluster) sampling?

- With an eye on environmental blocking and extrapolation in feature space, is it intended to apply the model to predict "new" outcomes for unobserved values of predictor variables?

Based on these criteria users may choose a matching resampling method that is either more restrictive (by discarding nearby observations for fold creation) or more liberal (by not removing observations and eventually ignoring natural grouping patterns). The specific publications related to the methods integrated into **mlr3spatiotempcv** may give further advice and provide additional use cases for the application of each respective approach. Users should therefore also refer to publications that are referenced of linked in the help files of this package or its respective upstream packages.

6.2. Resampling in model optimization

CV is also widely used to assess model performance when tuning hyperparameters or performing feature selection (Cai, Luo, Wang, and Yang 2018; Bischl *et al.* 2023). The **mlr3** framework supports the use of CV for both approaches. Using the CV methods introduced here, **mlr3** can therefore be used to optimize models to show an improved performance in specific spatial or spatiotemporal predictive settings (Schratz *et al.* 2019). Such an optimization may, for example, result in a reduced maximum tree depth or increased minimum node size in the Ecuador case study, since these hyperparameter settings would result in a stronger generalization and reduced overfitting. Usually both hyperparameter and feature selection (wrapper feature selection or filters) are combined within a single, nested optimization process (Schratz, Muenchow, Iturritxa, Cortés, Bischl, and Brenning 2021).

In nested CV specifically, an "inner" CV is performed on each CV training set, since hyperparameter tuning is an integral part of model fitting that should not be able to use information from the outer CV test set as this would result in information leakage. In such scenarios it is recommended to use the same spatial resampling method for the inner CV (hyperparameter tuning) as for the outer CV (model assessment) in order to use the appropriate objective function for optimization. See Schratz *et al.* (2019) for more details as well as chapter 11 of Lovelace, Nowosad, and Muenchow (2019).

6.3. Additional practical issues

Since **mlr3spatiotempcv** harvests already implemented resampling methods from existing R packages, the broader overview presented in this work has highlighted that there are still several gaps that may need to be closed in the future, if specific use cases require those features.

For example, buffering, or the use of a spatial or temporal separation distance between training and test sets, is currently only implemented for some methods (`"spcv_buffer"`, `"spcv_disc"`, and `"sptcv_cstf"` with both `space_var` and `time_var`). Its use should, however, be limited to use cases involving a prediction distance, as a buffer zone reduces the size of the training sample and introduces the risk of geographically biased training data (Zhu *et al.* 2015; Meyer and Pebesma 2021).

CV is often executed repeatedly to reduce the possible influence of random variability on CV estimates. In general, only methods that involve a random mechanism for generating or resampling blocks are suited for this. In leave-one-block-out CV, coordinate-based and

environmental clustering (`"spcv_coords"` and `"spcv_env"`) achieve this as their clusters are generated based on random seeds. However, experience with `"spcv_coords"` shows that clusters from repeated executions may in some situations be nearly identical to each other, resulting in very little variability between CV repetitions. While this effect also depends on the variable used for clustering, similar effects could potentially also apply to `"spcv_env"`. However, such effects are more difficult to quantify because selected features of these methods are always different, in contrast to `"spcv_coords"` which always uses coordinates for clustering. This issue is even more critical in CV at the block level with `"spcv_block"` with options `selection = "systematic"` and `selection = "checkerboard"` because identical folds are assigned in each repetition. In contrast, `"spcv_block"` with option `selection = "random"` avoids this problem.

7. Conclusion and outlook

The **mlr3spatiotempcv** package is the first package to bundle and categorize spatiotemporal resampling methods implemented in multiple other packages in R. The available resampling techniques allow users to vary the scale or granularity of the resampled spatiotemporal units as well as their shape and possible buffer distance between training and test samples. These settings may account for the specific characteristics of spatiotemporal prediction tasks, but modelers now have to make the important decision of choosing a method that is adequate for their situation. They are advised to focus on the spatial or spatiotemporal structure of the model's prediction task, consider the structure of the learning sample at hand, and think about how the autocorrelation between training and test samples might affect their model assessment and selection.

The compilation of resampling techniques in **mlr3spatiotempcv** is by no means complete. Additional methods or parameters may therefore be added in the future as they become available in upstream packages or are contributed directly to this package.

Spatiotemporal cross-validation as a paradigm is not yet fully established in scientific workflows, although it has been discussed intensively for more than a decade now. We anticipate that making the existing methods easily accessible to users is an important step to foster the acceptance of spatiotemporal cross-validation in the community and to allow modelers to produce bias-reduced model assessments in environmental and ecological studies.

Computational details

The results in this paper have been obtained with R version 4.4.0 on platform aarch64-apple-darwin20 (64-bit) running under macOS Sonoma 14.1.1.

Package versions: **mlr3learners** 0.7.0, **patchwork** 1.2.0, **mlr3spatiotempcv** 2.3.2, **mlr3** 0.20.0, **blockCV** 3.1-4, **plotly** 4.10.4, **ggplot2** 3.5.1, **DBI** 1.2.2, **pROC** 1.18.5, **rlang** 1.1.4, **magrittr** 2.0.3, **e1071** 1.7-14, **compiler** 4.4.0, **png** 0.1-8, **vctrs** 0.6.5, **reshape2** 1.4.4, **stringr** 1.5.1, **pkgconfig** 2.0.3, **fastmap** 1.2.0, **backports** 1.4.1, **utf8** 1.2.4, **rmarkdown** 2.27, **prodlim** 2024.06.25, **markdown** 1.13, **tinytex** 0.50, **purrr** 1.0.2, **xfun** 0.43, **mlr3misc** 0.15.0, **jsonlite** 1.8.8, **recipes** 1.1.0, **pak** 0.7.2, **uuid** 1.2-0, **mlr3measures** 0.5.0, **terra** 1.7-78, **parallel** 4.4.0, **R6** 2.5.1, **stringi** 1.8.4, **ranger** 0.16.0, **reticulate** 1.38.0, **parallelly** 1.37.1, **rpart** 4.1.23, **lubridate** 1.9.3, **Rcpp** 1.0.12, **iterators** 1.0.14, **knitr** 1.47, **future.apply** 1.11.2, **FNN** 1.1.4, **Matrix** 1.7-0, **splines**

4.4.0, **nnet** 7.3-19, **timechange** 0.3.0, **tidyselect** 1.2.1, **yaml** 2.3.8, **timeDate** 4032.109, **gg-text** 0.1.2, **codetools** 0.2-20, **listenv** 0.9.1, **lattice** 0.22-6, **tibble** 3.2.1, **plyr** 1.8.9, **withr** 3.0.0, **ROCR** 1.0-11, **evaluate** 0.24, **future** 1.33.2, **survival** 3.6-4, **sf** 1.0-16, **units** 0.8-5, **proxy** 0.4-27, **xml2** 1.3.6, **pillar** 1.9.0, **KernSmooth** 2.23-22, **checkmate** 2.3.1, **foreach** 1.5.2, **stats4** 4.4.0, **generics** 0.1.3, **rprojroot** 2.0.4, **munsell** 0.5.1, **commonmark** 1.9.1, **scales** 1.3.0, **globals** 0.16.3, **class** 7.3-22, **RhpcBLASctl** 0.23-42, **glue** 1.7.0, **lazyeval** 0.2.2, **sperrorest** 3.0.5, **tools** 4.4.0, **data.table** 1.15.4, **ModelMetrics** 1.2.2.2, **gower** 1.0.1, **forcats** 1.0.0, **grid** 4.4.0, **tidyr** 1.3.1, **crosstalk** 1.2.1, **ipred** 0.9-14, **colorspace** 2.1-0, **paradox** 1.0.1, **nlme** 3.1-164, **palmerpenguins** 0.1.1, **cli** 3.6.3, **twosamples** 2.0.1, **fansi** 1.0.6, **viridisLite** 0.4.2, **lava** 1.8.0, **dplyr** 1.1.4, **gtable** 0.3.5, **digest** 0.6.35, **classInt** 0.4-10, **caret** 6.0-94, **htmlwidgets** 1.6.4, **lgr** 0.4.4, **farver** 2.1.2, **htmltools** 0.5.8.1, **lifecycle** 1.0.4, **httr** 1.4.7, **hardhat** 1.4.0, **here** 1.0.1, **gridtext** 0.1.5, **MASS** 7.3-60.2, **CAST** 1.0.2.

References

Anderson P, Turner MG, Forester JD, Zhu J, Boyce MS, Beyer H, Stowell L (2005). "Scale-Dependent Summer Resource Selection by Reintroduced Elk in Wisconsin, USA." *The Journal of Wildlife Management*, **69**(1), 298–310. doi:10.2193/0022-541x(2005)069<0298:ssrsbr>2.0.co;2.

Arlot S, Celisse A (2010). "A Survey of Cross-Validation Procedures for Model Selection." *Statistics Surveys*, **4**(none), 40–79. doi:10.1214/09-ss054.

Barrett T, Dowle M, Srinivasan A, Gorecki J, Chirico M, Hocking T (2024). ***data.table**: Extension of* `data.frame`. doi:10.32614/CRAN.package.data.table. R package version 1.15.4.

Bebber DP, Butt N (2017). "Tropical Protected Areas Reduced Deforestation Carbon Emissions by One Third from 2000–2012." *Scientific Reports*, **7**(1), 14005. doi:10.1038/s41598-017-14467-w.

Becker M, Schratz P (2024). ***mlr3spatial**: Support for Spatial Objects within the **mlr3** Ecosystem*. doi:10.32614/CRAN.package.mlr3spatial. R package version 0.5.0.

Bengio Y, Grandvalet Y (2004). "No Unbiased Estimator of the Variance of k Fold Cross-Validation." *Journal of Machine Learning Research*, **5**, 1089–1105. doi:10.1007/0-387-24555-3_5.

Bergmeir C, Hyndman RJ, Koo B (2018). "A Note on the Validity of Cross-Validation for Evaluating Autoregressive Time Series Prediction." *Computational Statistics & Data Analysis*, **120**, 70–83. doi:10.1016/j.csda.2017.11.003.

Binder M, Pfisterer F, Lang M, Schneider L, Kotthoff L, Bischl B (2021). "**mlr3pipelines** – Flexible Machine Learning Pipelines in R." *Journal of Machine Learning Research*, **22**(184), 1–7. doi:10.1201/9781003402848.

Bischl B, Binder M, Lang M, Pielok T, Richter J, Coors S, Thomas J, Ullmann T, Becker M, Boulesteix AL, Deng D, Lindauer M (2023). "Hyperparameter Optimization: Foundations,

Algorithms, Best Practices and Open Challenges." *WIREs Data Mining and Knowledge Discovery*, **13**(2), e1484. doi:10.1002/widm.1484.

Bischl B, Sonabend R, Kotthoff L, Lang M (2024). *Applied Machine Learning Using **mlr3** in R*. Chapman & Hall/CRC. doi:10.1201/9781003402848.

Brenning A (2005). "Spatial Prediction Models for Landslide Hazards: Review, Comparison and Evaluation." *Natural Hazards and Earth System Sciences*, **5**(6), 853–862. doi:10.5194/nhess-5-853-2005.

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*. IEEE. doi:10.1109/igarss.2012.6352393.

Brenning A (2023). "Spatial Machine-Learning Model Diagnostics: A Model-Agnostic Distance-Based Approach." *International Journal of Geographical Information Science*, **37**(3), 584–606. doi:10.1080/13658816.2022.2131789.

Brenning A, Lausen B (2008). "Estimating Error Rates in the Classification of Paired Organs." *Statistics in Medicine*, **27**(22), 4515–4531. doi:10.1002/sim.3310.

Brenning A, Schwinn M, Ruiz-Páez AP, Muenchow J (2015). "Landslide Susceptibility near Highways Is Increased by 1 Order of Magnitude in the Andes of Southern Ecuador, Loja Province." *Natural Hazards and Earth System Sciences*, **15**(1), 45–57. doi:10.5194/nhess-15-45-2015.

Cai J, Luo J, Wang S, Yang S (2018). "Feature Selection in Machine Learning: A New Perspective." *Neurocomputing*, **300**, 70–79. doi:10.1016/j.neucom.2017.11.077.

Cawley GC, Talbot NLC (2010). "On Over-Fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation." *Journal of Machine Learning Research*, **11**(70), 2079–2107. doi:10.1007/978-3-319-62416-7_14.

Chang W (2021). ***R6**: Encapsulated Classes with Reference Semantics*. doi:10.32614/CRAN.package.r6. R package version 2.5.1.

Comber S (2020). ***spacv**: Spatial Cross-Validation in Python*. Python package version 0.0.22, URL https://pypi.org/project/spacv/.

Cressie NAC (1993). *Statistics for Spatial Data*. John Wiley & Sons. doi:10.1002/9781119115151.

Diesing M (2020). "Deep-Sea Sediments of the Global Ocean." *Earth System Science Data*, **12**(4), 3367–3381. doi:10.5194/essd-12-3367-2020.

Efron B, Gong G (1983). "A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation." *The American Statistician*, **37**(1), 36–48. doi:10.1080/00031305.1983.10483087.

Egli S, Höpke M (2020). "CNN-Based Tree Species Classification Using High Resolution RGB Image Data from Automated UAV Observations." *Remote Sensing*, **12**(23), 3892. doi:10.3390/rs12233892.

Endicott S, Drescher M, Brenning A (2017). "Modelling the Spread of European Buckthorn in the Region of Waterloo." *Biological Invasions*, **19**(10), 2993–3011. doi:10.1007/s10530-017-1504-3.

Escobar S, Helmstetter AJ, Jarvie S, Montúfar R, Balslev H, Couvreur TLP (2021). "Pleistocene Climatic Fluctuations Promoted Alternative Evolutionary Histories in Phytelephas Aequatorialis, an Endemic Palm from Western Ecuador." *Journal of Biogeography*, **48**(5), 1023–1037. doi:10.1111/jbi.14055.

Fleck S (2022). ***lgr**: A Fully Featured Logging Framework*. doi:10.32614/CRAN.package.lgr. R package version 0.4.4.

Gao J, Liang T, Yin J, Ge J, Feng Q, Wu C, Hou M, Liu J, Xie H (2019). "Estimation of Alpine Grassland Forage Nitrogen Coupled with Hyperspectral Characteristics during Different Growth Periods on the Tibetan Plateau." *Remote Sensing*, **11**(18), 2085. doi:10.3390/rs11182085.

Geiß C, Aravena Pelizari P, Schrade H, Brenning A, Taubenböck H (2017). "On the Effect of Spatially Non-Disjoint Training and Test Samples on Estimated Model Generalization Capabilities in Supervised Classification with Spatial Features." *IEEE Geoscience and Remote Sensing Letters*, **14**(11), 2008–2012. doi:10.1109/lgrs.2017.2747222.

Ghariani W (2023). "**spatial-Kfold**: A Python Package for Spatial Resampling toward More Reliable Cross-Validation in Spatial Studies." Python package version 0.0.3, URL https://pypi.org/project/spatial-kfold/.

Hand DJ (1997). *Construction and Assessment of Classification Rules*. John Wiley & Sons, New York.

Hartigan JA, Wong MA (1979). "Algorithm AS 136: A k Means Clustering Algorithm." *Journal of the Royal Statistical Society C*, **28**(1), 100–108. doi:10.2307/2346830.

Hijmans RJ, Phillips S, Leathwick J, Elith J (2023). ***dismo**: Species Distribution Modeling*. doi:10.32614/CRAN.package.dismo. R package version 1.3-14.

Hyndman RJ, Koehler AB (2006). "Another Look at Measures of Forecast Accuracy." *International Journal of Forecasting*, **22**(4), 679–688. doi:10.1016/j.ijforecast.2006.03.001.

Jensen DA, Rao M, Zhang J, Grøn M, Tian S, Ma K, Svenning JC (2021). "The Potential for Using Rare, Native Species in Reforestation – A Case Study of Yews (Taxaceae) in China." *Forest Ecology and Management*, **482**, 118816. doi:10.1016/j.foreco.2020.118816.

Karasiak N, Dejoux JF, Monteil C, Sheeren D (2021). "Spatial Dependence between Training and Test Sets: Another Pitfall of Classification Accuracy Assessment in Remote Sensing." *Machine Learning*. doi:10.1007/s10994-021-05972-1.

Kasurak A, Kelly R, Brenning A (2011). "Linear Mixed Modelling of Snow Distribution in the Central Yukon." *Hydrological Processes*, **25**(21), 3332–3346. doi:10.1002/hyp.8168.

Kohavi R (1995). "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection." In *Proceedings of the 14th International Joint Conference on Artificial*

Intelligence – Volume 2, IJCAI'95, pp. 1137–1143. Montreal, Quebec, Canada. URL https://www.ijcai.org/Proceedings/95-2/Papers/016.pdf.

Kuhn M (2008). "Building Predictive Models in R Using the **caret** Package." *Journal of Statistical Software*, **28**(5), 1–26. doi:10.18637/jss.v028.i05.

Kuhn M, Wickham H (2020). ***tidymodels**: A Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles*. URL https://www.tidymodels.org/.

Lang M, Au Q, Coors S, Schratz P, Becker M (2024a). ***mlr3learners**: Recommended Learners for **mlr3***. doi:10.32614/CRAN.package.mlr3learners. R package version 0.7.0.

Lang M, Binder M, Richter J, Schratz P, Pfisterer F, Coors S, Au Q, Casalicchio G, Kotthoff L, Bischl B (2019). "**mlr3**: A Modern Object-Oriented Machine Learning Framework in R." *Journal of Open Source Software*, **4**(44), 1903. doi:10.21105/joss.01903.

Lang M, Pfisterer F, Gruber C, Nawrath J, Arshadipour A (2023). ***mlr3temporal**: Temporal Prediction/Forecasting for **mlr3***. R package version 0.1.0.9000, URL https://github.com/mlr-org/mlr3temporal.

Lang M, Schratz P, Sonabend R, Becker M, Richter J, Zobolas J (2024b). ***mlr3viz**: Visualizations for **mlr3***. doi:10.32614/CRAN.package.mlr3viz. R package version 0.9.0.

Linnenbrink J, Milà C, Ludwig M, Meyer H (2023). "kNNDM: k Fold Nearest Neighbour Distance Matching Cross-Validation for Map Accuracy Estimation." *EGUsphere*, pp. 1–16. doi:10.5194/egusphere-2023-1308.

Lovelace R, Nowosad J, Muenchow J (2019). *Geocomputation with R*. Chapman & Hall/CRC. doi:10.1201/9780203730058.

Ludwig M, Moreno-Martinez A, Hölzel N, Pebesma E, Meyer H (2023). "Assessing and Improving the Transferability of Current Global Spatial Prediction Models." *Global Ecology and Biogeography*, **32**(3), 356–368. doi:10.1111/geb.13635.

Mahoney MJ, Johnson LK, Silge J, Frick H, Kuhn M, Beier CM (2023a). "Assessing the Performance of Spatial Cross-Validation Approaches for Models of Spatially Structured Data." *arXiv 2303.07334*, arXiv.org E-Print Archive. doi:10.48550/arxiv.2303.07334.

Mahoney MJ, Silge J, Posit Software, PBC (2023b). ***spatialsample**: Spatial Resampling Infrastructure*. doi:10.32614/CRAN.package.spatialsample. R package version 0.5.1.

Martin ME, Plourde LC, Ollinger SV, Smith ML, McNeil BE (2008). "A Generalizable Method for Remote Sensing of Canopy Nitrogen across a Wide Range of Forest Ecosystems." *Remote Sensing of Environment*, **112**(9), 3511–3519. doi:10.1016/j.rse.2008.04.008.

Meyer H, Milà C, Ludwig M, Linnenbrink J, Schumacher F (2024). ***CAST**: **caret** Applications for Spatial-Temporal Models*. doi:10.32614/CRAN.package.cast. R package version 1.0.2.

Meyer H, Pebesma E (2021). "Predicting into Unknown Space? Estimating the Area of Applicability of Spatial Prediction Models." *Methods in Ecology and Evolution*, **12**(9), 1620–1633. doi:10.1111/2041-210x.13650.

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.

Milà C, Mateu J, Pebesma E, Meyer H (2022). "Nearest Neighbour Distance Matching Leave-One-Out Cross-Validation for Map Validation." *Methods in Ecology and Evolution*, **13**(6), 1304–1316. doi:10.1111/2041-210x.13851.

Møller AB, Mulder VL, Heuvelink GBM, Jacobsen NM, Greve MH (2021). "Can We Use Machine Learning for Agricultural Land Suitability Assessment?" *Agronomy*, **11**(4), 703. doi:10.3390/agronomy11040703.

Morera A, Martínez de Aragón J, Bonet JA, Liang J, de Miguel S (2021). "Performance of Statistical and Machine Learning-Based Methods for Predicting Biogeographical Patterns of Fungal Productivity in Forest Ecosystems." *Forest Ecosystems*, **8**(1), 21. doi:10.1186/s40663-021-00297-w.

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. doi:10.1016/j.geomorph.2011.10.029.

Muscarella R, Galante PJ, Soley-Guardia M, Boria RA, Kass JM, Uriarte M, Anderson RP (2014). "**ENMeval**: An R Package for Conducting Spatially Independent Evaluations and Estimating Optimal Model Complexity for Maxent Ecological Niche Models." *Methods in Ecology and Evolution*, **5**(11), 1198–1205. doi:10.1111/2041-210x.12261.

Pebesma E (2018). "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal*, **10**(1), 439–446. doi:10.32614/rj-2018-009.

Peña MA, Brenning A (2015). "Assessing Fruit-Tree Crop Classification from Landsat-8 Time Series for the Maipo Valley, Chile." *Remote Sensing of Environment*, **171**, 234–244. doi:10.1016/j.rse.2015.10.029.

Ploton P, Mortier F, Réjou-Méchain M, Barbier N, Picard N, Rossi V, Dormann C, Cornu G, Viennois G, Bayol N, Lyapustin A, Gourlet-Fleury S, Pélissier R (2020). "Spatial Validation Reveals Poor Predictive Performance of Large-Scale Ecological Mapping Models." *Nature Communications*, **11**(1), 4540. doi:10.1038/s41467-020-18321-y.

Pohjankukka J, Pahikkala T, Nevalainen P, Heikkonen J (2017). "Estimating the Prediction Performance of Spatial Models via Spatial k Fold Cross Validation." *International Journal of Geographical Information Science*, **31**(10), 2001–2019. doi:10.1080/13658816.2017.1346255.

Reitz O, Graf A, Schmidt M, Ketzler G, Leuchner M (2021). "Upscaling Net Ecosystem Exchange over Heterogeneous Landscapes with Machine Learning." *Journal of Geophysical Research: Biogeosciences*, **126**(2), e2020JG005814. doi:10.1029/2020jg005814.

Rest KL, Pinaud D, Monestiez P, Chadoeuf J, Bretagnolle V (2014). "Spatial Leave-One-out Cross-Validation for Variable Selection in the Presence of Spatial Autocorrelation." *Global Ecology and Biogeography*, **23**(7), 811–820. doi:10.1111/geb.12161.

Roberts DR, Bahn V, Ciuti S, Boyce MS, Elith J, Guillera-Arroita G, Hauenstein S, Lahoz-Monfort JJ, Schröder B, Thuiller W, Warton DI, Wintle BA, Hartig F, Dormann CF (2017). "Cross-Validation Strategies for Data with Temporal, Spatial, Hierarchical, or Phylogenetic Structure." *Ecography*, **40**(8), 913–929. doi:10.1111/ecog.02881.

Ruß G, Brenning A (2010). "Data Mining in Precision Agriculture: Management of Spatial Information." In E Hüllermeier, R Kruse, F Hoffmann (eds.), *Computational Intelligence for Knowledge-Based Systems Design*, Lecture Notes in Computer Science, pp. 350–359. Springer-Verlag, Berlin. doi:10.1007/978-3-642-14049-5_36.

Schratz P, Muenchow J, Iturritxa E, Cortés J, Bischl B, Brenning A (2021). "Monitoring Forest Health Using Hyperspectral Imagery: Does Feature Selection Improve the Performance of Machine-Learning Techniques?" *Remote Sensing*, **13**(23), 4832. doi:10.3390/rs13234832.

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.

Sievert C (2020). *Interactive Web-Based Data Visualization with R, **plotly**, and **shiny***. Chapman & Hall/CRC. doi:10.1201/9780429447273.

Stewart SB, Elith J, Fedrigo M, Kasel S, Roxburgh SH, Bennett LT, Chick M, Fairman T, Leonard S, Kohout M, Cripps JK, Durkin L, Nitschke CR (2021). "Climate Extreme Variables Generated Using Monthly Time-Series Data Improve Predicted Distributions of Plant Species." *Ecography*, **44**(4), 626–639. doi:10.1111/ecog.05253.

Thompson SK (2012). *Sampling*. 3rd edition. John Wiley & Sons. doi:10.1002/9781118162934.

Valavi R, Elith J, Lahoz-Monfort JJ, Guillera-Arroita G, Valavi R, Elith J, Lahoz-Monfort JJ, Guillera-Arroita G (2019). "**blockCV**: An R Package for Generating Spatially or Environmentally Separated Folds for k Fold Cross-Validation of Species Distribution Models." *Methods in Ecology and Evolution*, **10**(2), 225–232. doi:10.1111/2041-210x.13107.

Vanwinckelen G, Blockeel H (2012). "On Estimating Model Accuracy with Repeated Cross-Validation." In *BeneLearn 2012: Proceedings of the 21st Belgian-Dutch Conference on Machine Learning*, pp. 39–44. URL https://lirias.kuleuven.be/1655861.

Wadoux AMJC, Heuvelink GBM, De Bruin S, Brus DJ (2021). "Spatial Cross-Validation Is Not the Right Way to Evaluate Map Accuracy." *Ecological Modelling*, **457**, 109692. doi:10.1016/j.ecolmodel.2021.109692.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag, New York. doi:10.1007/978-0-387-98141-3.

Willmott CJ, Matsuura K (2006). "On the Use of Dimensioned Measures of Error to Evaluate the Performance of Spatial Interpolators." *International Journal of Geographical Information Science*, **20**(1), 89–102. doi:10.1080/13658810500286976.

Wright MN, Ziegler A (2017). "**ranger**: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software*, **77**(1), 1–17. doi:10.18637/jss.v077.i01.

Wu T, Luo J, Dong W, Gao L, Hu X, Wu Z, Sun Y, Liu J (2020). "Disaggregating County-Level Census Data for Population Mapping Using Residential Geo-Objects with Multisource Geo-Spatial Data." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **13**, 1189–1205. doi:10.1109/jstars.2020.2974896.

Zhang Y, Yang Y (2015). "Cross-Validation for Selecting a Model Selection Procedure." *Journal of Econometrics*, **187**(1), 95–112. doi:10.1016/j.jeconom.2015.02.006.

Zhu AX, Liu J, Du F, Zhang SJ, Qin CZ, Burt J, Behrens T, Scholten T (2015). "Predictive Soil Mapping with Limited Sample Data." *European Journal of Soil Science*, **66**(3), 535–547. doi:10.1111/ejss.12244.

Zurell D, Zimmermann NE, Gross H, Baltensweiler A, Sattler T, Wüest RO (2020). "Testing Species Assemblage Predictions from Stacked and Joint Species Distribution Models." *Journal of Biogeography*, **47**(1), 101–113. doi:10.1111/jbi.13608.

Affiliation:

Patrick Schratz, Alexander Brenning
Friedrich Schiller University Jena
Department of Geography
Geographic Information Science Group
E-mail: patrick.schratz@uni-jena.de, alexander.brenning@uni-jena.de

Marc Becker
Ludwig-Maximilians-Universität München
Department of Statistics
Statistical Learning and Data Science Group
E-mail: marc.becker@stat.uni-muenchen.de

Michel Lang
TU Dortmund University
Faculty of Statistics
E-mail: lang@statistik.tu-dortmund.de

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
November 2024, Volume 111, Issue 7 *Submitted:* 2022-06-23
doi:10.18637/jss.v111.i07 *Accepted:* 2024-04-28