

Package: spatialcvR (via r-universe)

October 11, 2026

Title Spatial Cross-Validation for Machine Learning

Version 0.1.0

Description Spatial cross-validation and model evaluation for geospatial machine learning applications. Addresses spatial dependence in observations by implementing spatial block, buffered, and clustering cross-validation methods. Includes spatial leakage detection, model performance metrics, and spatial residual diagnostics for assessing model generalization across geographic space. Methods based on Brenning (2012) <[doi:10.1016/j.cageo.2012.02.001](https://doi.org/10.1016/j.cageo.2012.02.001)>, Pohjankukka et al. (2017) <[doi:10.1016/j.isprsjprs.2017.07.001](https://doi.org/10.1016/j.isprsjprs.2017.07.001)>, and Roberts et al. (2017) <[doi:10.1111/ecog.02881](https://doi.org/10.1111/ecog.02881)>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Imports sf (>= 1.0-0)

Suggests dplyr (>= 1.0.0), ggplot2 (>= 3.4.0), tidymodels (>= 1.0.0), rsample (>= 1.0.0), testthat (>= 3.0.0), knitr (>= 1.40), rmarkdown (>= 2.14), terra (>= 1.5.0)

VignetteBuilder knitr

URL <https://sowsalim01.github.io/spatialcvR/>,
<https://github.com/sowsalim01/spatialcvR>

BugReports <https://github.com/sowsalim01/spatialcvR/issues>

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository <https://sowsalim01.r-universe.dev>

Date/Publication 2026-09-26 16:48:50 UTC

RemoteUrl <https://github.com/sowsalim01/spatialcvr>

RemoteRef HEAD
RemoteSha eab19c88a0c5f65156923b388f6fcc2dc4851664

Contents

compare_cv	2
detect_spatial_leakage	3
plot_spatial_folds	5
plot_spatial_residuals	6
print.cv_comparison	7
print.spatial_distance	7
print.spatial_folds	8
print.spatial_leakage_result	8
print.spatial_metrics	9
print.spatial_residuals	9
sample_spatial_data	10
spatial_block_folds	11
spatial_buffer_folds	12
spatial_cluster_folds	13
spatial_distance	15
spatial_folds	16
spatial_metrics	17
spatial_residuals	18
spatial_split	20
Index	21

compare_cv	<i>Compare Cross-Validation Methods</i>
------------	---

Description

Compares performance metrics across different cross-validation methods to assess the impact of spatial dependence on model evaluation.

Usage

```
compare_cv(results_list, methods = NULL)
```

Arguments

- results_list Named list of spatial_metrics objects from different CV methods
- methods Character vector of method names (optional, uses names of results_list)

Details

The function compares metrics across different CV methods:

- Creates summary table with mean and SD of metrics per method
- Identifies best performing method
- Provides fold-level comparison
- Calculates performance differences between methods

This is useful for comparing spatial CV methods against random CV to demonstrate the impact of spatial dependence.

Value

An object of class "cv_comparison" containing:

summary Data frame with summary statistics for each method

by_fold Data frame with fold-level results

best_method Method with best performance according to RMSE

comparison Performance comparison between methods

See Also

Other model evaluation functions: [spatial_metrics\(\)](#)

Examples

```
# Train model with different CV methods
results_random <- list(RMSE = 0.5, MAE = 0.4, R2 = 0.8)
results_block <- list(RMSE = 0.7, MAE = 0.6, R2 = 0.6)
results_cluster <- list(RMSE = 0.6, MAE = 0.5, R2 = 0.7)

comparison <- compare_cv(
  list(random = results_random, block = results_block, cluster = results_cluster)
)
print(comparison)
```

detect_spatial_leakage

Detect Spatial Leakage in Cross-Validation Folds

Description

Analyzes the proximity between training and test observations to detect potential spatial leakage, where training and test sets are too close spatially, leading to over-optimistic performance estimates.

Usage

```
detect_spatial_leakage(
  data,
  folds,
  x = NULL,
  y = NULL,
  threshold = NULL,
  risk_levels = NULL
)
```

Arguments

<code>data</code>	Spatial observations (data.frame or sf object)
<code>folds</code>	A spatial_folds object
<code>x</code>	Name of the x coordinate column (required for data.frame, ignored for sf)
<code>y</code>	Name of the y coordinate column (required for data.frame, ignored for sf)
<code>threshold</code>	Distance threshold for considering observations "too close" (default: NULL, auto-calculated)
<code>risk_levels</code>	Custom risk level thresholds as list(min, moderate, high) (default: NULL)

Details

The function analyzes spatial distances between training and test observations:

- Calculates minimum, mean, and median distances per fold
- Identifies observations within threshold distance
- Assesses overall risk level
- Provides recommendations for improvement

Risk levels are based on the proportion of train/test pairs that are too close:

- "low": < 10% of pairs below threshold
- "moderate": 10-30% of pairs below threshold
- "high": > 30% of pairs below threshold

Value

An object of class "spatial_leakage_result" containing:

method CV method used

fold_distances Distance analysis for each fold

summary_statistics Overall summary across all folds

risk_level Overall risk assessment: "low", "moderate", or "high"

recommendations Text recommendations based on analysis

threshold Threshold used for risk assessment

See Also

Other spatial analysis functions: [spatial_distance\(\)](#)

Examples

```
data(sample_spatial_data)
folds <- spatial_folds(sample_spatial_data, "longitude", "latitude", k = 5)
leakage <- detect_spatial_leakage(sample_spatial_data, folds, "longitude", "latitude")
print(leakage)
```

plot_spatial_folds	<i>Plot Spatial Cross-Validation Folds</i>
--------------------	--

Description

Creates a visual representation of spatial cross-validation folds, showing the spatial distribution of training and test observations.

Usage

```
plot_spatial_folds(folds, data, x = NULL, y = NULL, fold = 1, main = NULL, ...)
```

Arguments

folds	A spatial_folds object
data	Spatial observations (data.frame or sf object)
x	Name of the x coordinate column (required for data.frame, ignored for sf)
y	Name of the y coordinate column (required for data.frame, ignored for sf)
fold	Fold number to plot (default: 1, or "all" for all folds)
main	Plot title (default: NULL, auto-generated)
...	Additional graphical parameters passed to plot()

Details

Creates a scatter plot showing training and test observations with different colors. For spatial block CV, also shows block boundaries when available.

Value

Invisibly returns the fold object

See Also

Other visualization functions: [plot_spatial_residuals\(\)](#)

Examples

```
data(sample_spatial_data)
folds <- spatial_folds(sample_spatial_data, "longitude", "latitude", k = 5)
plot_spatial_folds(folds, sample_spatial_data, "longitude", "latitude", fold = 1)
```

plot_spatial_residuals

Plot Spatial Residuals

Description

Creates visualizations of spatial residuals to analyze the spatial distribution of model errors.

Usage

```
plot_spatial_residuals(residuals_obj, type = "scatter", main = NULL, ...)
```

Arguments

residuals_obj	A spatial_residuals object
type	Type of plot: "scatter", "histogram", or "qq" (default: "scatter")
main	Plot title (default: NULL, auto-generated)
...	Additional graphical parameters passed to plot()

Details

Creates different types of residual plots:

- "scatter": Spatial scatter plot of residuals colored by magnitude
- "histogram": Histogram of residual values
- "qq": Q-Q plot for normality assessment

Value

Invisibly returns the residuals object

See Also

Other visualization functions: [plot_spatial_folds\(\)](#)

Examples

```
observed <- c(1, 2, 3, 4, 5)
predicted <- c(1.1, 2.2, 2.8, 4.1, 4.9)
coords <- cbind(x = c(0, 1, 2, 3, 4), y = c(0, 1, 2, 3, 4))
residuals <- spatial_residuals(observed, predicted, coords)
plot_spatial_residuals(residuals, type = "scatter")
```

print.cv_comparison	<i>Print method for cv_comparison objects</i>
---------------------	---

Description

Print method for cv_comparison objects

Usage

```
## S3 method for class 'cv_comparison'
print(x, ...)
```

Arguments

x	A cv_comparison object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.spatial_distance	<i>Print method for spatial_distance objects</i>
------------------------	--

Description

Print method for spatial_distance objects

Usage

```
## S3 method for class 'spatial_distance'
print(x, ...)
```

Arguments

x	A spatial_distance object
...	Additional arguments (ignored)

Value

Invisibly returns the object

```
print.spatial_folds
```

Print method for spatial_folds objects

Description

Print method for spatial_folds objects

Usage

```
## S3 method for class 'spatial_folds'
print(x, ...)
```

Arguments

x	A spatial_folds object
...	Additional arguments (ignored)

Value

Invisibly returns the object

```
print.spatial_leakage_result
```

Print method for spatial_leakage_result objects

Description

Print method for spatial_leakage_result objects

Usage

```
## S3 method for class 'spatial_leakage_result'
print(x, ...)
```

Arguments

x	A spatial_leakage_result object
...	Additional arguments (ignored)

Value

Invisibly returns the object

`print.spatial_metrics` *Print method for spatial_metrics objects*

Description

Print method for spatial_metrics objects

Usage

```
## S3 method for class 'spatial_metrics'
print(x, ...)
```

Arguments

<code>x</code>	A spatial_metrics object
<code>...</code>	Additional arguments (ignored)

Value

Invisibly returns the object

`print.spatial_residuals`
Print method for spatial_residuals objects

Description

Print method for spatial_residuals objects

Usage

```
## S3 method for class 'spatial_residuals'
print(x, ...)
```

Arguments

<code>x</code>	A spatial_residuals object
<code>...</code>	Additional arguments (ignored)

Value

Invisibly returns the object

sample_spatial_data	<i>Sample Spatial Data for Demonstration</i>
---------------------	--

Description

A synthetic dataset containing 200 spatial observations with coordinates and variables for demonstration and testing purposes.

Usage

```
data(sample_spatial_data)
```

Format

A data frame with 200 rows and 6 columns:

id Unique identifier for each observation (integer)
longitude X coordinate in projected CRS (numeric)
latitude Y coordinate in projected CRS (numeric)
variable1 First predictor variable with spatial structure (numeric)
variable2 Second predictor variable with spatial structure (numeric)
target Target variable for prediction (numeric)

Details

The data was generated to simulate spatial autocorrelation:

- Coordinates are in a UTM-like projected system (0-1000 range)
- Variables show gradients in X and Y directions
- Target variable combines predictors with spatial structure
- Some missing values are included for testing NA handling

Source

Synthetic data generated for package demonstration

Examples

```
data(sample_spatial_data)  
head(sample_spatial_data)  
summary(sample_spatial_data)
```

Description

Creates spatial cross-validation folds by dividing the study area into rectangular blocks and assigning observations to folds based on their block membership. This method ensures spatial separation between training and test sets.

Usage

```
spatial_block_folds(
  data,
  x = NULL,
  y = NULL,
  k = 5,
  block_size = NULL,
  n_blocks = NULL,
  assignment = "systematic",
  seed = NULL
)
```

Arguments

data	Spatial observations (data.frame or sf object)
x	Name of the x coordinate column (required for data.frame, ignored for sf)
y	Name of the y coordinate column (required for data.frame, ignored for sf)
k	Number of folds (default: 5)
block_size	Size of blocks as c(width, height) in coordinate units
n_blocks	Number of blocks in x and y directions as c(nx, ny)
assignment	Strategy for assigning blocks to folds: "systematic" or "random"
seed	Random seed for reproducibility (default: NULL)

Details

The spatial block method divides the spatial extent into a grid of rectangular blocks. Two approaches are available:

- Fixed block size: Specify block_size to control block dimensions
- Fixed number of blocks: Specify n_blocks to control grid resolution

If neither is specified, the function attempts to create approximately $\sqrt{k}$ blocks in each direction. Assignment strategies:

- "systematic": Assigns blocks to folds in a systematic pattern
- "random": Randomly assigns blocks to folds

Value

An object of class "spatial_folds" containing fold assignments

See Also

Other spatial cross-validation functions: [spatial_buffer_folds\(\)](#), [spatial_cluster_folds\(\)](#), [spatial_folds\(\)](#), [spatial_split\(\)](#)

Examples

```
data(sample_spatial_data)
folds <- spatial_block_folds(
  data = sample_spatial_data,
  x = "longitude",
  y = "latitude",
  k = 5,
  block_size = c(200, 200)
)
```

spatial_buffer_folds *Spatial Buffered Cross-Validation*

Description

Creates spatial cross-validation folds by excluding training observations within a specified buffer radius around test observations. This method ensures a minimum spatial separation between training and test sets.

Usage

```
spatial_buffer_folds(
  data,
  x = NULL,
  y = NULL,
  k = 5,
  buffer_radius = NULL,
  seed = NULL
)
```

Arguments

data	Spatial observations (data.frame or sf object)
x	Name of the x coordinate column (required for data.frame, ignored for sf)
y	Name of the y coordinate column (required for data.frame, ignored for sf)
k	Number of folds (default: 5)

buffer_radius Buffer radius in coordinate units
seed Random seed for reproducibility (default: NULL)

Details

The buffered method ensures that for each test observation, no training observation falls within the specified buffer radius. This is particularly useful when you need strict control over the minimum distance between training and test observations.

Note: This method requires a projected CRS for accurate distance calculations. Using geographic coordinates (longitude/latitude) will produce warnings.

Value

An object of class "spatial_folds" containing fold assignments

See Also

Other spatial cross-validation functions: [spatial_block_folds\(\)](#), [spatial_cluster_folds\(\)](#), [spatial_folds\(\)](#), [spatial_split\(\)](#)

Examples

```
data(sample_spatial_data)
folds <- spatial_buffer_folds(
  data = sample_spatial_data,
  x = "longitude",
  y = "latitude",
  k = 5,
  buffer_radius = 100
)
```

spatial_cluster_folds *Spatial Clustering Cross-Validation*

Description

Creates spatial cross-validation folds by clustering observations spatially and assigning clusters to folds. This method is useful for data with complex spatial structure.

Usage

```
spatial_cluster_folds(
  data,
  x = NULL,
  y = NULL,
  k = 5,
```

```
n_clusters = NULL,  
seed = NULL  
)
```

Arguments

<code>data</code>	Spatial observations (data.frame or sf object)
<code>x</code>	Name of the x coordinate column (required for data.frame, ignored for sf)
<code>y</code>	Name of the y coordinate column (required for data.frame, ignored for sf)
<code>k</code>	Number of folds (default: 5)
<code>n_clusters</code>	Number of spatial clusters (default: k)
<code>seed</code>	Random seed for reproducibility (default: NULL)

Details

The clustering method groups spatially proximate observations into clusters using k-means clustering on coordinates, then assigns clusters to folds. This ensures spatial coherence within folds while maintaining separation between folds.

Value

An object of class "spatial_folds" containing fold assignments

See Also

Other spatial cross-validation functions: [spatial_block_folds\(\)](#), [spatial_buffer_folds\(\)](#), [spatial_folds\(\)](#), [spatial_split\(\)](#)

Examples

```
data(sample_spatial_data)  
folds <- spatial_cluster_folds(  
  data = sample_spatial_data,  
  x = "longitude",  
  y = "latitude",  
  k = 5,  
  n_clusters = 10  
)
```

spatial_distance	<i>Calculate Spatial Distances Between Train and Test Observations</i>
------------------	--

Description

Computes distances between training and test observations for each fold in a spatial cross-validation setup. This helps assess the spatial separation between training and test sets.

Usage

```
spatial_distance(data, folds, x = NULL, y = NULL)
```

Arguments

<code>data</code>	Spatial observations (data.frame or sf object)
<code>folds</code>	A spatial_folds object
<code>x</code>	Name of the x coordinate column (required for data.frame, ignored for sf)
<code>y</code>	Name of the y coordinate column (required for data.frame, ignored for sf)

Details

For each fold, the function calculates Euclidean distances between all pairs of training and test observations. This provides a comprehensive view of spatial separation.

Note: Distance calculations use Euclidean distance on the provided coordinates. For accurate metric distances, ensure data is in a projected CRS. Geographic coordinates (longitude/latitude) will produce approximate distances.

Value

A list containing distance information for each fold:

- fold** Fold number
- min_distance** Minimum distance between train and test observations
- mean_distance** Mean distance between train and test observations
- median_distance** Median distance between train and test observations
- max_distance** Maximum distance between train and test observations
- sd_distance** Standard deviation of distances
- quantiles** Distance quantiles (25%, 50%, 75%)

See Also

Other spatial analysis functions: [detect_spatial_leakage\(\)](#)

Examples

```
data(sample_spatial_data)
folds <- spatial_folds(sample_spatial_data, "longitude", "latitude", k = 5)
distances <- spatial_distance(sample_spatial_data, folds, "longitude", "latitude")
print(distances)
```

spatial_folds

Create Spatial Cross-Validation Folds

Description

Creates spatially separated folds for model evaluation to address spatial dependence in observations. This function serves as a wrapper for different spatial cross-validation methods.

Usage

```
spatial_folds(
  data,
  x = NULL,
  y = NULL,
  k = 5,
  method = "block",
  seed = NULL,
  ...
)
```

Arguments

data	Spatial observations (data.frame or sf object)
x	Name of the x coordinate column (required for data.frame, ignored for sf)
y	Name of the y coordinate column (required for data.frame, ignored for sf)
k	Number of folds (default: 5)
method	Spatial CV method: "block", "buffer", "cluster", or "random" (default: "block")
seed	Random seed for reproducibility (default: NULL)
...	Additional parameters passed to specific methods

Details

The function validates inputs and dispatches to the appropriate spatial CV method:

- "block": Spatial block cross-validation (default)
- "buffer": Buffered cross-validation
- "cluster": Spatial clustering cross-validation
- "random": Random spatial split (baseline)

Value

An object of class "spatial_folds" containing:

- folds** List of train/test indices for each fold
- method** Method used for fold creation
- k** Number of folds
- parameters** List of parameters used
- coordinates** Coordinate matrix of observations
- crs** Coordinate reference system
- metadata** Additional metadata

See Also

Other spatial cross-validation functions: [spatial_block_folds\(\)](#), [spatial_buffer_folds\(\)](#), [spatial_cluster_folds\(\)](#), [spatial_split\(\)](#)

Examples

```
data(sample_spatial_data)
folds <- spatial_folds(
  data = sample_spatial_data,
  x = "longitude",
  y = "latitude",
  k = 5,
  method = "block"
)
```

spatial_metrics	<i>Spatial Model Evaluation Metrics</i>
-----------------	---

Description

Calculates standard model evaluation metrics for comparing observed and predicted values. These metrics are commonly used in machine learning and spatial modeling.

Usage

```
spatial_metrics(observed, predicted, na.rm = TRUE)
```

Arguments

- observed Numeric vector of observed values
- predicted Numeric vector of predicted values
- na.rm Logical, whether to remove NA values (default: TRUE)

Details

The function calculates the following metrics:

- RMSE: $\sqrt{\text{mean}((\text{observed} - \text{predicted})^2)}$
- MAE: $\text{mean}(\text{abs}(\text{observed} - \text{predicted}))$
- R2: $1 - \text{sum}((\text{observed} - \text{predicted})^2) / \text{sum}((\text{observed} - \text{mean}(\text{observed}))^2)$
- MAPE: $\text{mean}(\text{abs}((\text{observed} - \text{predicted}) / \text{observed})) * 100$ (only when no zeros in observed)

Value

An object of class "spatial_metrics" containing:

n Number of observations

RMSE Root Mean Square Error

MAE Mean Absolute Error

R2 R-squared (coefficient of determination)

MAPE Mean Absolute Percentage Error (when applicable)

See Also

Other model evaluation functions: [compare_cv\(\)](#)

Examples

```
observed <- c(1, 2, 3, 4, 5)
predicted <- c(1.1, 2.2, 2.8, 4.1, 4.9)
metrics <- spatial_metrics(observed, predicted)
print(metrics)
```

spatial_residuals

Spatial Residual Diagnostics

Description

Analyzes the spatial distribution of model residuals to detect spatial patterns in prediction errors. This helps identify whether model errors are spatially autocorrelated, which may indicate missing spatial predictors or inappropriate model specification.

Usage

```
spatial_residuals(observed, predicted, coordinates, x = NULL, y = NULL)
```

Arguments

<code>observed</code>	Numeric vector of observed values
<code>predicted</code>	Numeric vector of predicted values
<code>coordinates</code>	Coordinate matrix or data.frame with x and y columns
<code>x</code>	Name of x coordinate column if coordinates is a data.frame
<code>y</code>	Name of y coordinate column if coordinates is a data.frame

Details

The function calculates residuals and provides summary statistics:

- Mean, median, SD of residuals
- Quantiles of residuals
- Normality test statistics (if sufficient data)

Spatial autocorrelation analysis can be added in future versions when appropriate dependencies (e.g., `spdep`) are available.

Value

An object of class "spatial_residuals" containing:

residuals Residual values (observed - predicted)

observed Observed values

predicted Predicted values

coordinates Coordinate matrix

statistics Summary statistics of residuals

Examples

```
observed <- c(1, 2, 3, 4, 5)
predicted <- c(1.1, 2.2, 2.8, 4.1, 4.9)
coords <- cbind(x = c(0, 1, 2, 3, 4), y = c(0, 1, 2, 3, 4))
residuals <- spatial_residuals(observed, predicted, coords)
print(residuals)
```

spatial_split	<i>Random Spatial Split</i>
---------------	-----------------------------

Description

Creates a random spatial split serving as a baseline for comparison with spatial cross-validation methods. This represents traditional random cross-validation without spatial considerations.

Usage

```
spatial_split(data, x = NULL, y = NULL, k = 5, seed = NULL)
```

Arguments

data	Spatial observations (data.frame or sf object)
x	Name of the x coordinate column (required for data.frame, ignored for sf)
y	Name of the y coordinate column (required for data.frame, ignored for sf)
k	Number of folds (default: 5)
seed	Random seed for reproducibility (default: NULL)

Details

This method performs standard random k-fold cross-validation without any spatial constraints. It serves as a baseline to compare against spatial cross-validation methods and demonstrate the impact of spatial dependence on model evaluation.

Value

An object of class "spatial_folds" containing fold assignments

See Also

Other spatial cross-validation functions: [spatial_block_folds\(\)](#), [spatial_buffer_folds\(\)](#), [spatial_cluster_folds\(\)](#), [spatial_folds\(\)](#)

Examples

```
data(sample_spatial_data)
folds <- spatial_split(
  data = sample_spatial_data,
  x = "longitude",
  y = "latitude",
  k = 5
)
```

Index

- * **datasets**
 - sample_spatial_data, [10](#)
- * **model evaluation functions**
 - compare_cv, [2](#)
 - spatial_metrics, [17](#)
- * **spatial analysis functions**
 - detect_spatial_leakage, [3](#)
 - spatial_distance, [15](#)
- * **spatial cross-validation functions**
 - spatial_block_folds, [11](#)
 - spatial_buffer_folds, [12](#)
 - spatial_cluster_folds, [13](#)
 - spatial_folds, [16](#)
 - spatial_split, [20](#)
- * **spatial diagnostics functions**
 - spatial_residuals, [18](#)
- * **visualization functions**
 - plot_spatial_folds, [5](#)
 - plot_spatial_residuals, [6](#)

compare_cv, [2](#)
compare_cv(), [18](#)

detect_spatial_leakage, [3](#)
detect_spatial_leakage(), [15](#)

plot_spatial_folds, [5](#)
plot_spatial_folds(), [6](#)
plot_spatial_residuals, [6](#)
plot_spatial_residuals(), [5](#)
print.cv_comparison, [7](#)
print.spatial_distance, [7](#)
print.spatial_folds, [8](#)
print.spatial_leakage_result, [8](#)
print.spatial_metrics, [9](#)
print.spatial_residuals, [9](#)

sample_spatial_data, [10](#)
spatial_block_folds, [11](#)
spatial_block_folds(), [13](#), [14](#), [17](#), [20](#)

spatial_buffer_folds, [12](#)
spatial_buffer_folds(), [12](#), [14](#), [17](#), [20](#)
spatial_cluster_folds, [13](#)
spatial_cluster_folds(), [12](#), [13](#), [17](#), [20](#)
spatial_distance, [15](#)
spatial_distance(), [5](#)
spatial_folds, [16](#)
spatial_folds(), [12–14](#), [20](#)
spatial_metrics, [17](#)
spatial_metrics(), [3](#)
spatial_residuals, [18](#)
spatial_split, [20](#)
spatial_split(), [12–14](#), [17](#)