Spatial Analysis and Modeling

Spatial Cross-Validation

Corso di formazione su ML e DL
Fondazione LINKS
25/09/2025

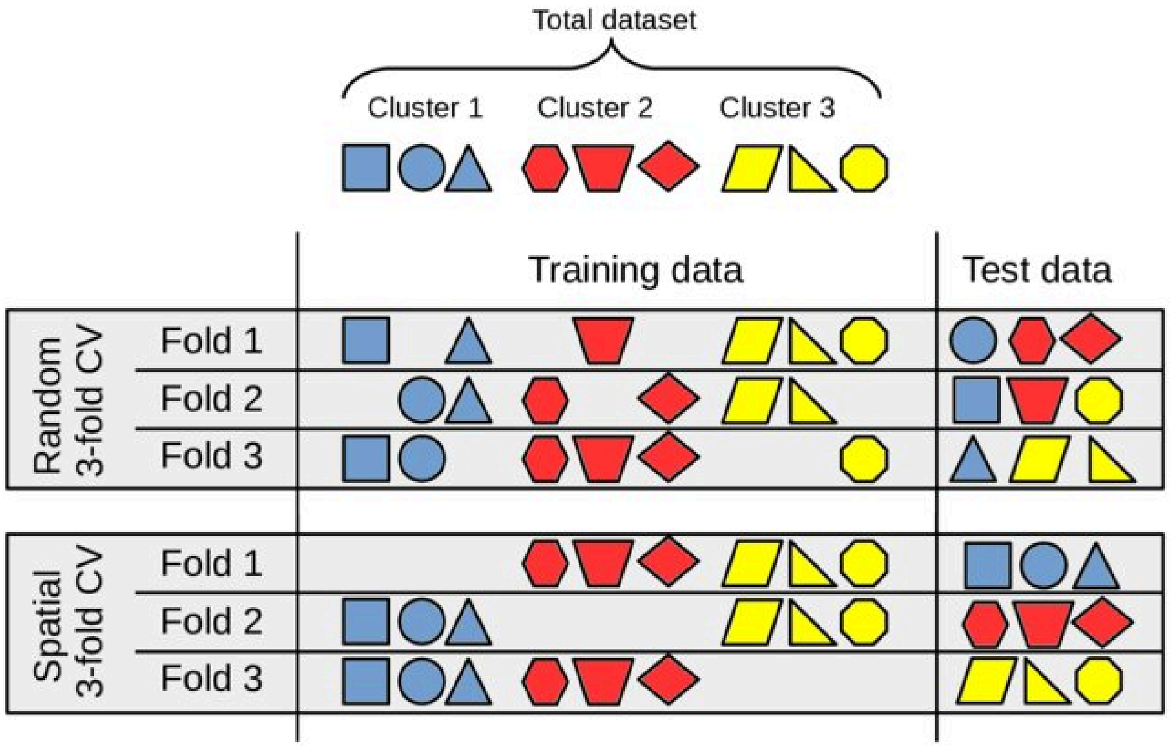
ISI Foundation

Why Standard CV Fails in Spatial Data

The Core Problem: Information Leakage

When your data has spatial autocorrelation, random K-fold CV creates a fundamental problem:

- **Train and test folds contain nearby locations** that share information
- This makes predictions appear **easier than they really are**
- Your model gets "hints" about test data through spatially correlated neighbors
    - Violates the **independence assumption** that CV relies on



Block Spatial Cross-Validation

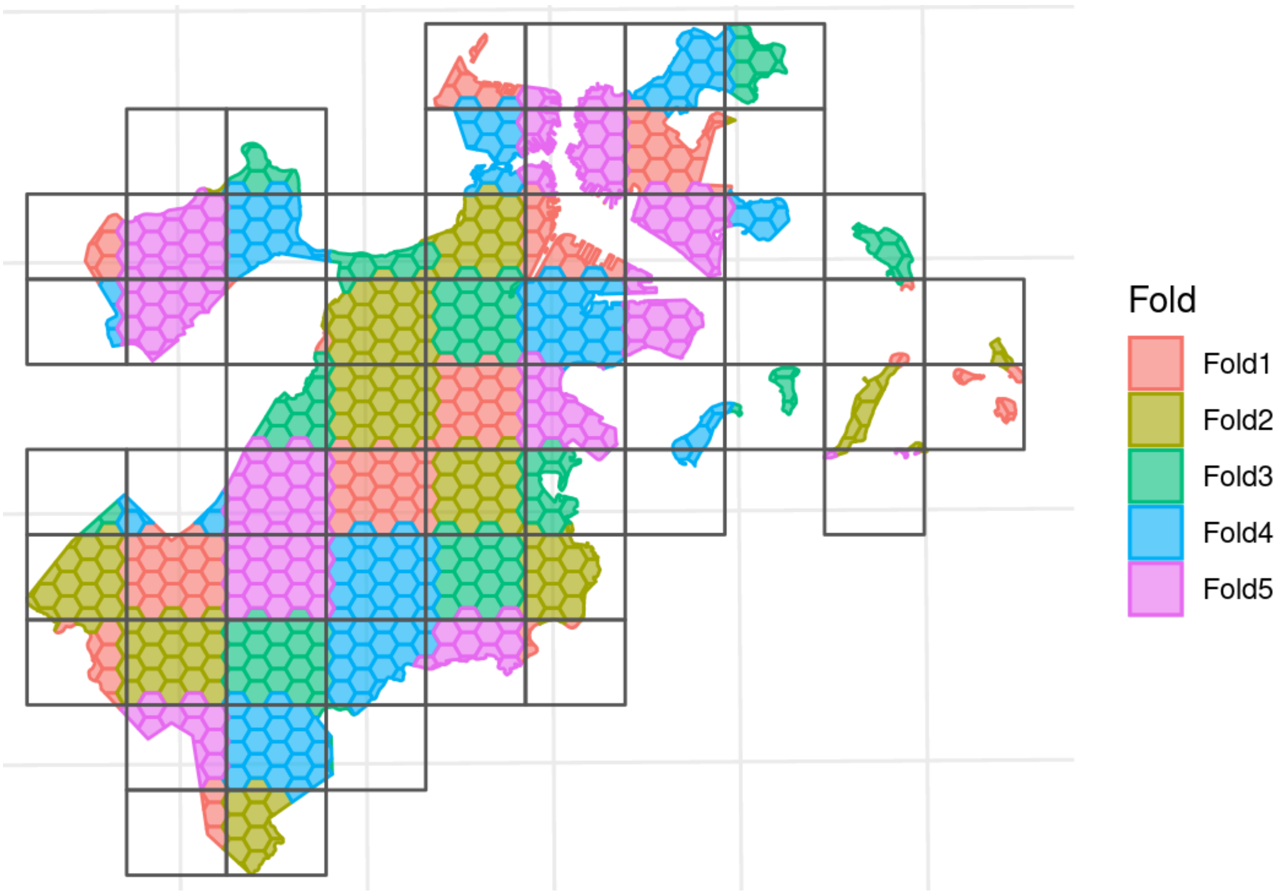
Concept: Partition space into contiguous blocks

Advantages:

- Preserves spatial structure
- Realistic evaluation scenario
- Simple to implement

Disadvantages:

- Unequal block sizes possible
- Edge effects between blocks



Clustering-based Spatial Cross-Validation

Concept:

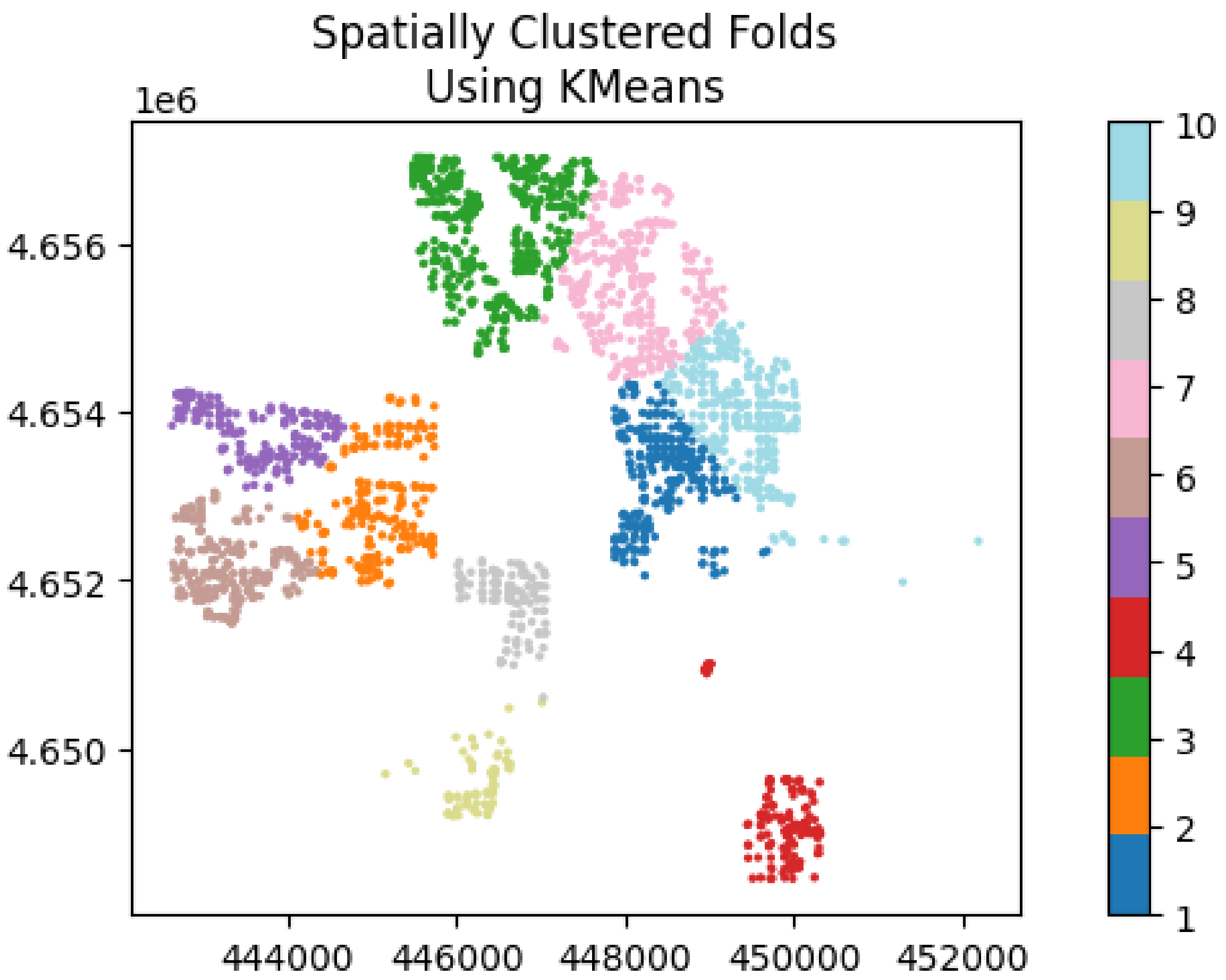
- Instead of regular spatial blocks, you can use **clustering** to define folds.
- Clusters can be formed using spatial coordinates or additional covariates (e.g., elevation, land cover, climate).
- Each cluster is treated as a test fold, with the remaining clusters as training data.

Advantages:

- Can create more ecologically meaningful folds, especially when covariates are used.
- Flexible: clusters can be spatially contiguous or disjoint, depending on the algorithm and input features.
- Useful for testing model transferability across environmental gradients.

Disadvantages:

- Fold sizes may be uneven, especially with natural clusters.
- Clusters may not be spatially contiguous, which can complicate interpretation.
- Choice of clustering algorithm and covariates can strongly affect results.
- May not fully eliminate spatial autocorrelation between folds if clusters are too small or too close.


Spatially Clustered Folds
Using KMeans
1e6
4.656
4.654
4.652
4.650
444000
446000
448000
450000
452000
10
9
8
7
6
5
4
3
2
1

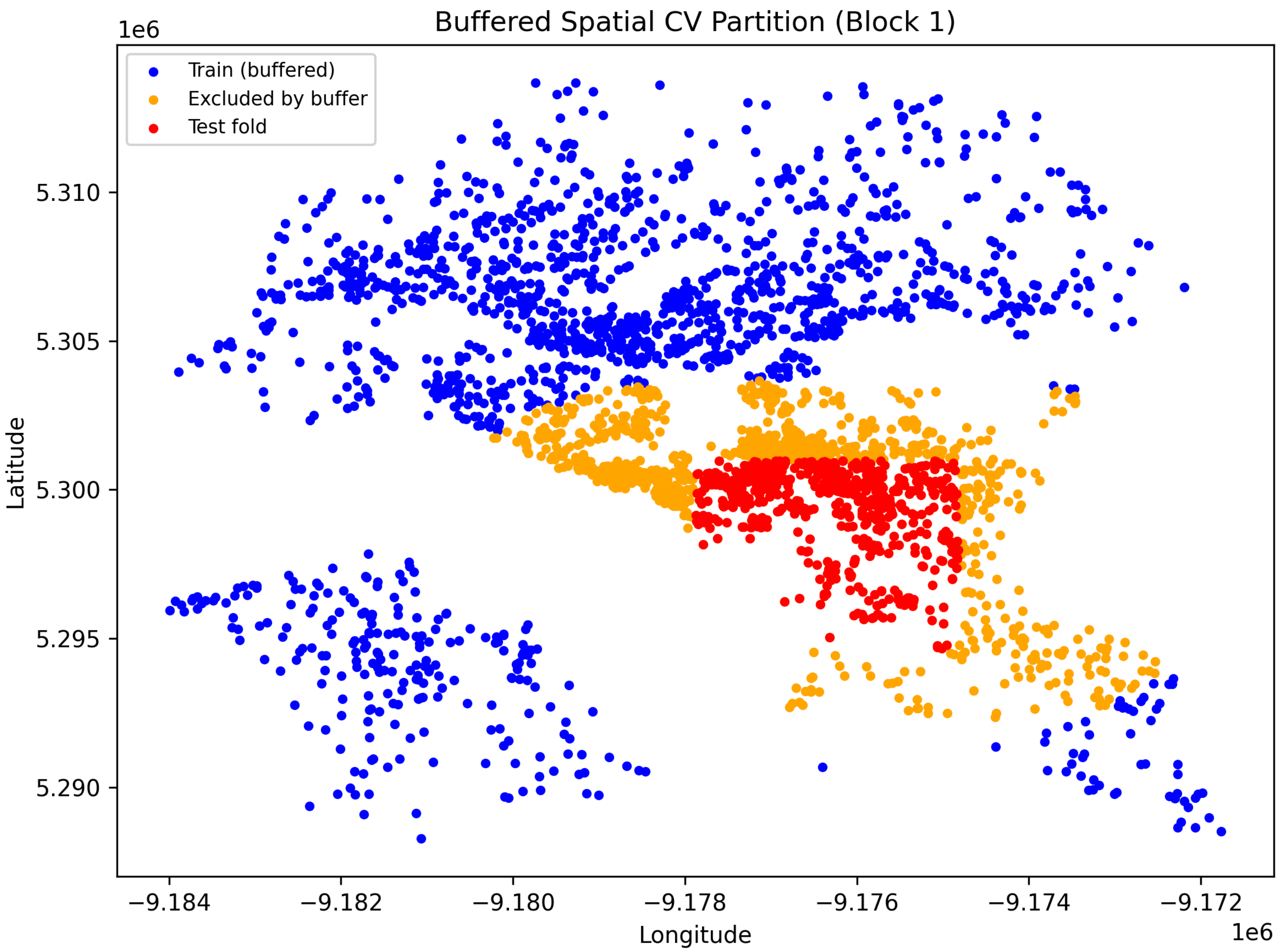
Buffered Spatial Cross-Validation

Concept:

- For each test location, exclude all training points within a **buffer** distance (e.g., based on spatial autocorrelation range).
- Ensures strict spatial separation between train and test sets.
- **Buffer size controls the minimum distance between test and training samples**.
- Especially useful for point data with spatial clustering or autocorrelation.

Buffer Size Selection:

- **Rule of thumb:** 2-3× correlation range
- **Empirical:** Test multiple buffer sizes
- **Theoretical:** Based on variogram range parameter


Buffered Spatial CV Partition (Block 1)
1e6
Train (buffered)
Excluded by buffer
Test fold
5.310
5.305
5.300
5.295
5.290
Latitude
−9.184
−9.182
−9.180
−9.178
−9.176
−9.174
−9.172
1e6
Longitude

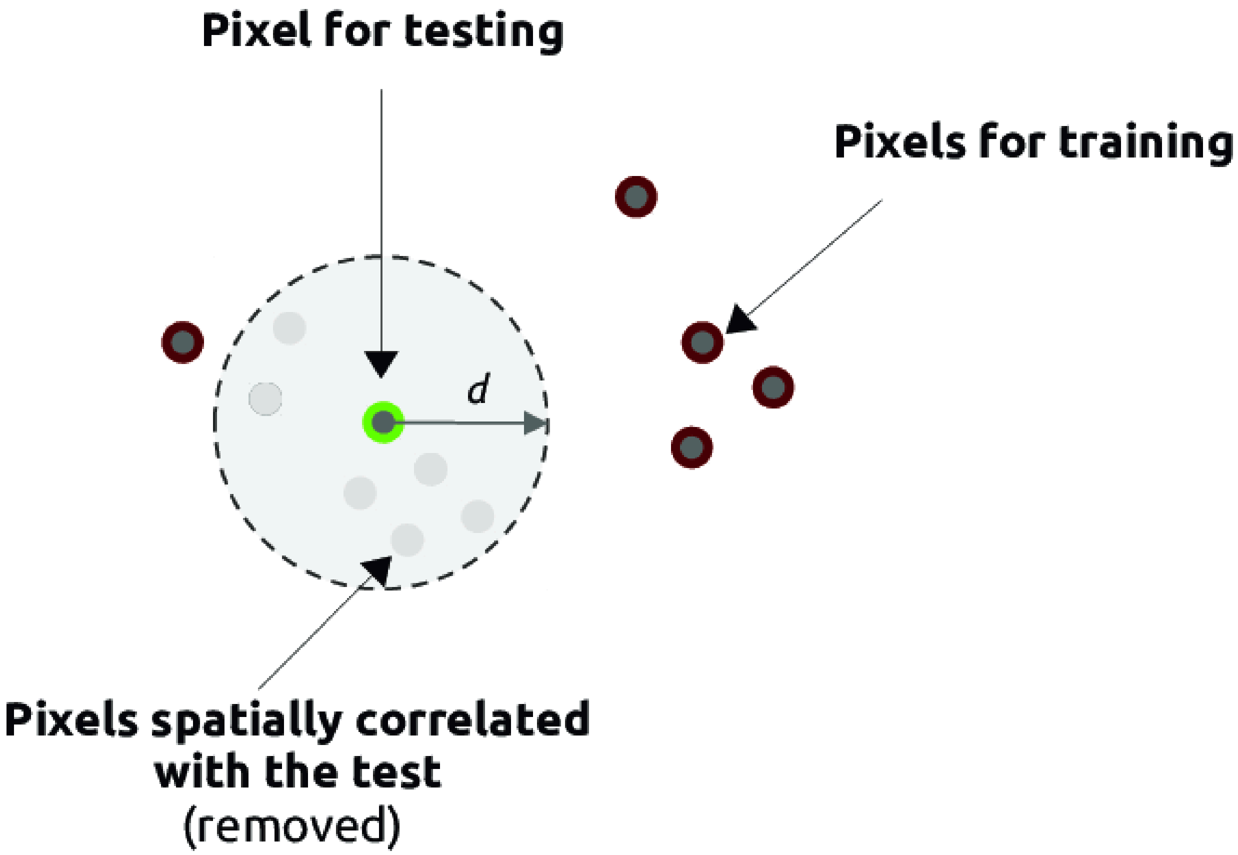
Leave-One-Out Spatial Cross-Validation (LOO-SCV)

Concept:

- For each observation, use it as the test set and all other observations as the training set.
- Strict independence: no overlap between train and test.

Spatial LOO Variant:

- Exclude not only the test point, but also **all points within a buffer distance** (see buffered CV).
- Useful for dense spatial samples and assessing model generalization at individual locations.



Leave-Location-Group-Out CV (LLGO)

Why LLGO?

- Standard Leave-One-Out (LOO) spatial CV tests model generalization at individual locations, but may not account for hierarchical or grouped spatial structure (e.g., cities, watersheds, farms).
- LLGO is designed for situations where data are naturally grouped by location, administrative unit, or other spatial hierarchy.
- Instead of leaving out single points, LLGO leaves out entire spatial groups, testing the model's ability to generalize to new regions or units.

How is LLGO different from LOO spatial CV?

- **LOO spatial CV:** Each test fold contains a single observation (or buffered area around it); evaluates point-level independence.
- **LLGO:** Each test fold contains all observations from a spatial group (e.g., all data from a city, watershed, or farm); evaluates group-level independence and transferability.
- LLGO is especially important when predictions will be made for new, unseen groups, not just new points within known groups.

Practical Recommendations Summary

For Areal Data:

- Use **contiguous block CV** or **leave-location-group-out**
- Buffer size: 1-2 neighboring units
- Report Moran's I on residuals

For Point Data:

- Use **buffered CV** with correlation range-based buffers
- Implement **target-oriented sampling** for clustered designs
- Check directional variograms

For Network Data:

- Use **leave-subgraph-out** CV
- Consider **network distance** in buffer calculations
- Evaluate **connectivity-based** accuracy metrics

Key Takeaways:

- **Use random CV** for spatial data only when sample is random
- **Always check residual spatial structure**

How to present results (recommended)

- Map residuals and local RMSE side-by-side with histogram / boxplot of residuals.
- Report Moran's I with p-value and a short interpretation sentence.
- Show empirical variogram with fitted model; annotate the range $\longrightarrow$ explain chosen buffer/block size.
- Report distribution (mean, median, SD, IQR) of the metric across folds (not only the mean).
- Show PICP and PINAW overall and by region (table + map).
    - PICP (Prediction Interval Coverage Probability): fraction of observations whose true value falls inside the predicted interval.

$$\mathrm{PICP} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i \in [\hat{y}_i^{\mathrm{lower}},\, \hat{y}_i^{\mathrm{upper}}]\}$$

Interpretation: PICP close to the nominal coverage (e.g., 0.95) indicates well-calibrated intervals.

- PINAW (Prediction Interval Normalized Average Width): average interval width normalized by the data range (smaller is tighter).

$$\mathrm{PINAW} = \frac{1}{n(y_{\max} - y_{\min})} \sum_{i=1}^{n} \left(\hat{y}_i^{\text{upper}} - \hat{y}_i^{\text{lower}}\right)$$

  Interpretation: trade-off between interval width and coverage — report both PICP and PINAW together.

How to make them spatial

- By-region (administrative units / strata)
  - Compute PICP and PINAW per region; show a table + choropleth map.
- Per-fold (CV fold) or per-cluster
  - Report PICP/PINAW distributions across folds/clusters to check transferability.
- Local / moving-window
  - Compute PICP/PINAW in a moving window or k-nearest neighborhood to map local calibration / sharpness.
- Map coverage flags (covered = 0/1) and test spatial autocorrelation (Moran's I) of coverage failures.
- Inspect PINAW spatially (map interval widths) to see where uncertainty is large.

Practical tips

- Always show both PICP (calibration) and PINAW (sharpness). High PICP + huge PINAW is not useful; low PINAW + poor PICP is overconfident.
- For region-level PICP, include confidence intervals (binomial) so you know whether deviations from nominal are significant.
- If coverage failures cluster spatially, investigate covariates / missing processes and consider location-dependent uncertainty models (heteroskedastic models, spatial random effects, spatial conformal methods).

blockCV R package

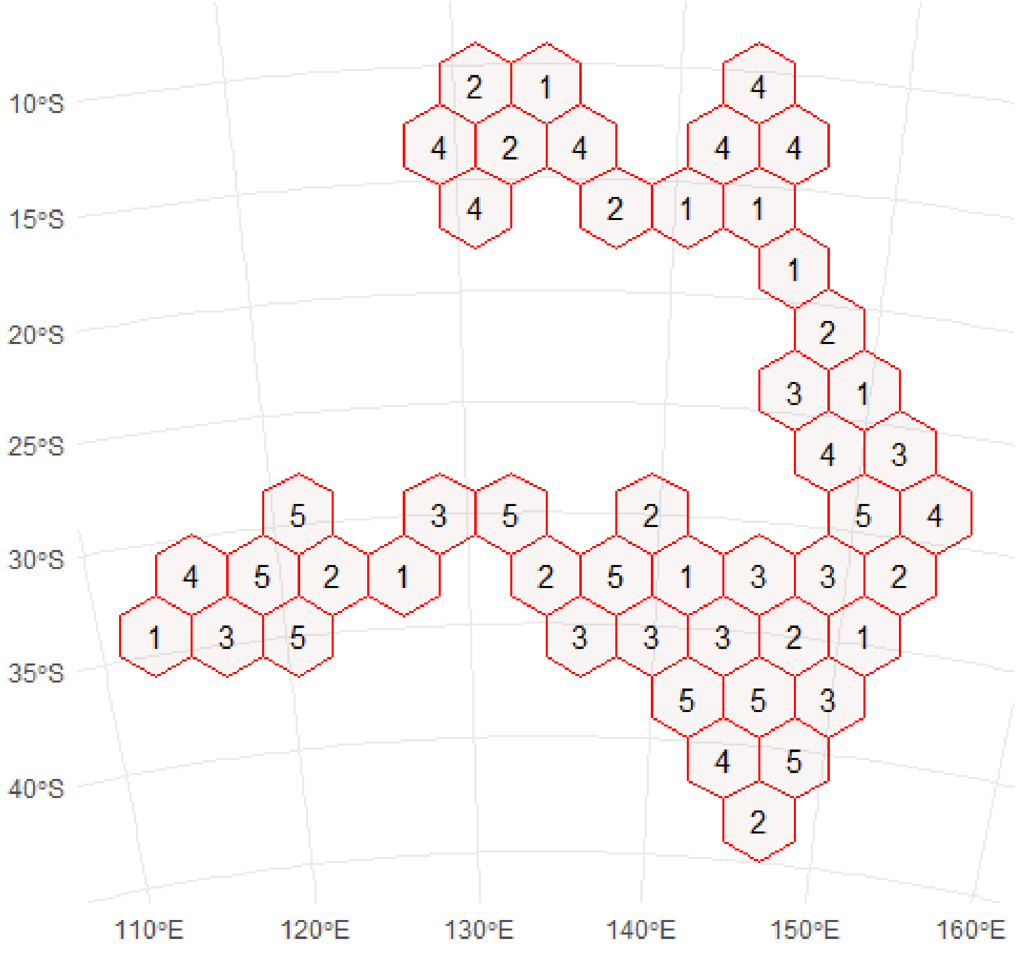
1. Why spatial cross-validation (SCV)?
2. The blockCV toolbox at a glance
3. Methods
    - Spatial blocks (`cv_spatial`)
    - Spatial/Environmental clustering (`cv_cluster`)
    - Buffered spatial LOO (`cv_buffer`)
    - NNDM LOO (`cv_nndm`)
4. Choosing a block/buffer size (`cv_spatial_autocor`, `cv_block_size`)
5. Plotting & diagnostics (`cv_plot`, `cv_similarity`)
6. Practical tips & references

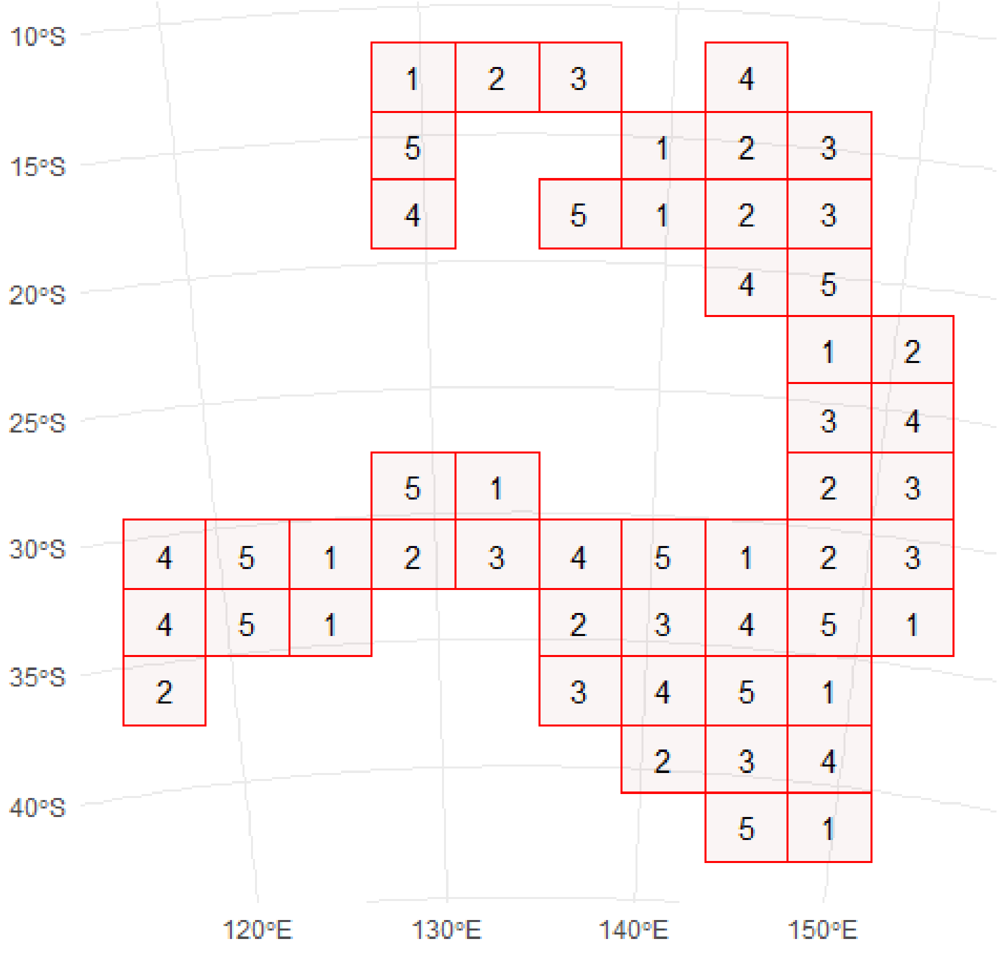
Method 1 — Spatial blocks (`cv_spatial`)

Idea. Cover the region with blocks (hex or square), then assign entire blocks to folds so train/test are spatially separated.
When to use. Classic k-fold SCV for gridded or regional studies; fair when blocks approximate the spatial scale of dependence.

Step-by-step:

1. **Choose block size** (metres) or grid (`rows_cols`).
2. **Make blocks**: hex (default) or squares.
3. **Assign folds** (k): random / systematic / checkerboard.
4. **Check balance** of train/test counts (tune `iteration` for better balance).
5. **Plot** with `cv_plot` to verify spatial separation.


10°S
15°S
20°S
25°S
30°S
35°S
40°S
110°E
120°E
130°E
140°E
150°E
160°E
2 1 4
4 2 4 4 4
4 2 1 1
1
2
3 1
4 3
5 3 5 2 5 4
4 5 2 1 2 5 1 3 3 2
1 3 5 3 3 3 2 1
5 5 3
4 5
2


10°S
15°S
20°S
25°S
30°S
35°S
40°S
120°E
130°E
140°E
150°E
1 2 3 4
5 1 2 3
4 5 1 2 3
4 5
1 2
3 4
5 1 2 3
4 5 1 2 3 4 5 1 2 3
4 5 1 2 3 4 5 1
2 3 4 5 1
2 3 4
5 1

`cv_spatial`: Tips & pitfalls

- **Scale matters**: set block `size` near the spatial autocorrelation range of your predictors or residuals.
- Use **random** with `iteration` to improve class balance in binary outcomes.
- If points cluster strongly, **systematic** or **checkerboard** can avoid empty test folds.
- Don’t mix coordinate units — `size` is interpreted in **metres**.

Method 2 — Spatial / Environmental clustering (`cv_cluster`)

Idea. Cluster points in **geographic** space (coords) or **environmental** space (rasters). Assign clusters to folds.

When to use.

- Spatial k-means: flexible shapes vs. rigid blocks.
- Environmental clustering: test extrapolation to novel environments, or de-bias by environment.

What you can cluster on

Coordinates of points (spatial clustering)

- If you do not supply raster covariates (r = NULL), then clustering is done in "spatial space", using the coordinates of your x sample points.
- Basically, cv_cluster(..., x = pa_data, k = something) clusters the x,y coordinates by k-means.

Environmental variables (covariates)

- If you supply a raster stack / SpatRaster object via r, then clustering is done in the space of environmental covariates.
- You can choose whether clustering is done using just the environmental values at the sample points, or across all raster cells ("raster space") – depending on the argument raster_cluster.

Method 3 — Buffered spatial LOO (`cv_buffer`)

Idea. For each target point, **buffer** by a distance and exclude all points in that buffer from training. Test on the target (optionally more).
When to use. Strict independence at a chosen distance; robust for dense or irregular samples; good for SDMs and map validation.

Step-by-step:

1. Pick buffer `size` (metres), ideally near the **autocorrelation range**.
2. For each record *i*:
   - **Test set** = record *i*.
   - **Train set** = all records **outside** the buffer around *i*.
3. Repeat for all records $\longrightarrow$ **LOO-like** folds (often many!).
4. Plot a subset of folds to inspect.

Method 4 — NNDM LOO (`cv_nndm`)

Idea. A fast C++ LOO that matches the **nearest-neighbour distance distribution** in train vs. test to mimic prediction settings (target domain).
When to use. When simple buffers don't reflect target sampling density or distribution; when you want **distribution-aware** separation.

Step-by-step:

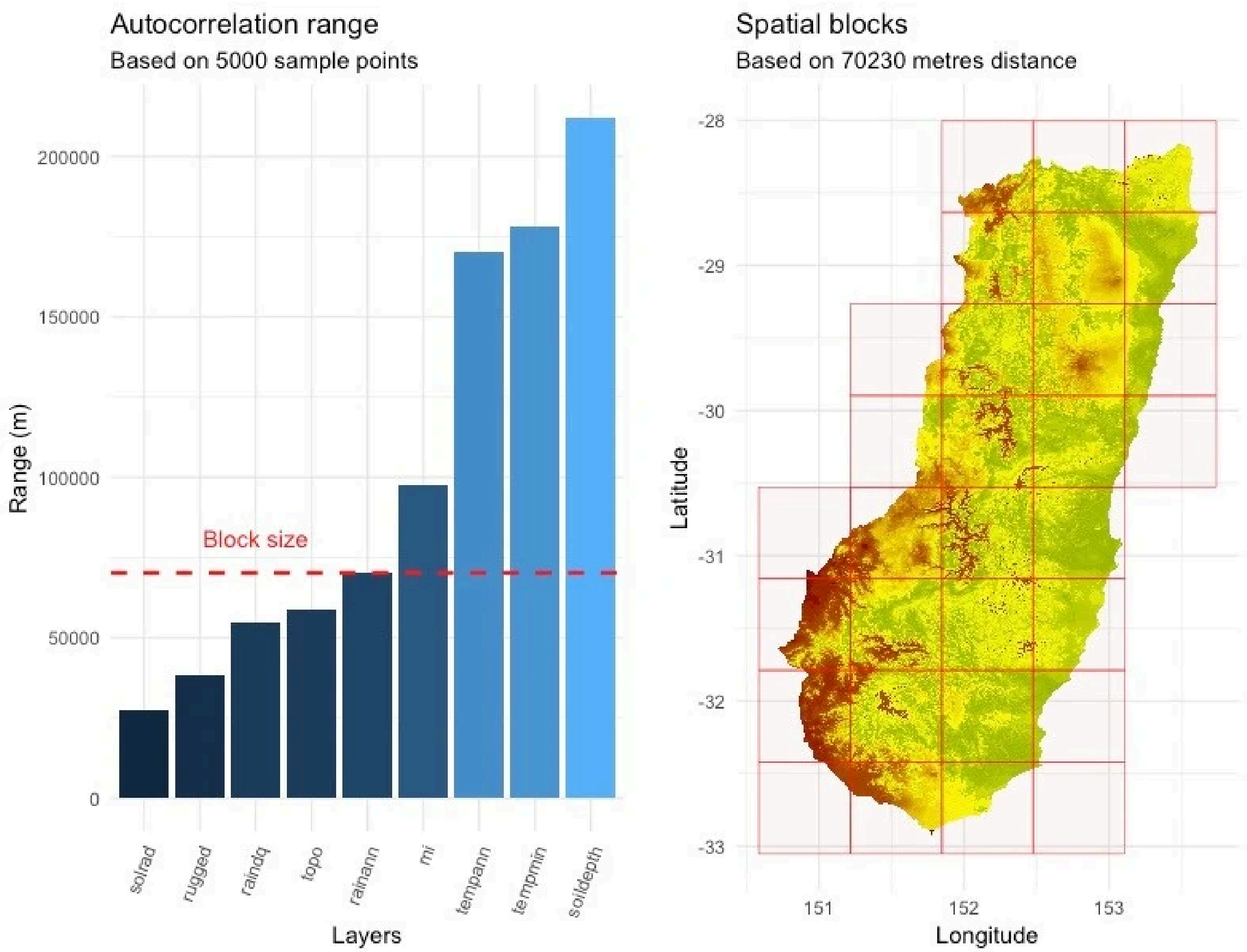
1. Provide sample `x` (and `column` for binary data).
2. Provide a **raster** or spatial extent representing the **prediction domain** (`r`).
3. Choose `size` (distance scale), `num_sample` (sampling points from domain), `sampling` ("regular"/"random"), and a minimum train fraction `min_train`.
4. Run and **plot** selected folds; evaluate.

Blocks vs. Clusters vs. Buffers vs. NNDM — quick contrasts

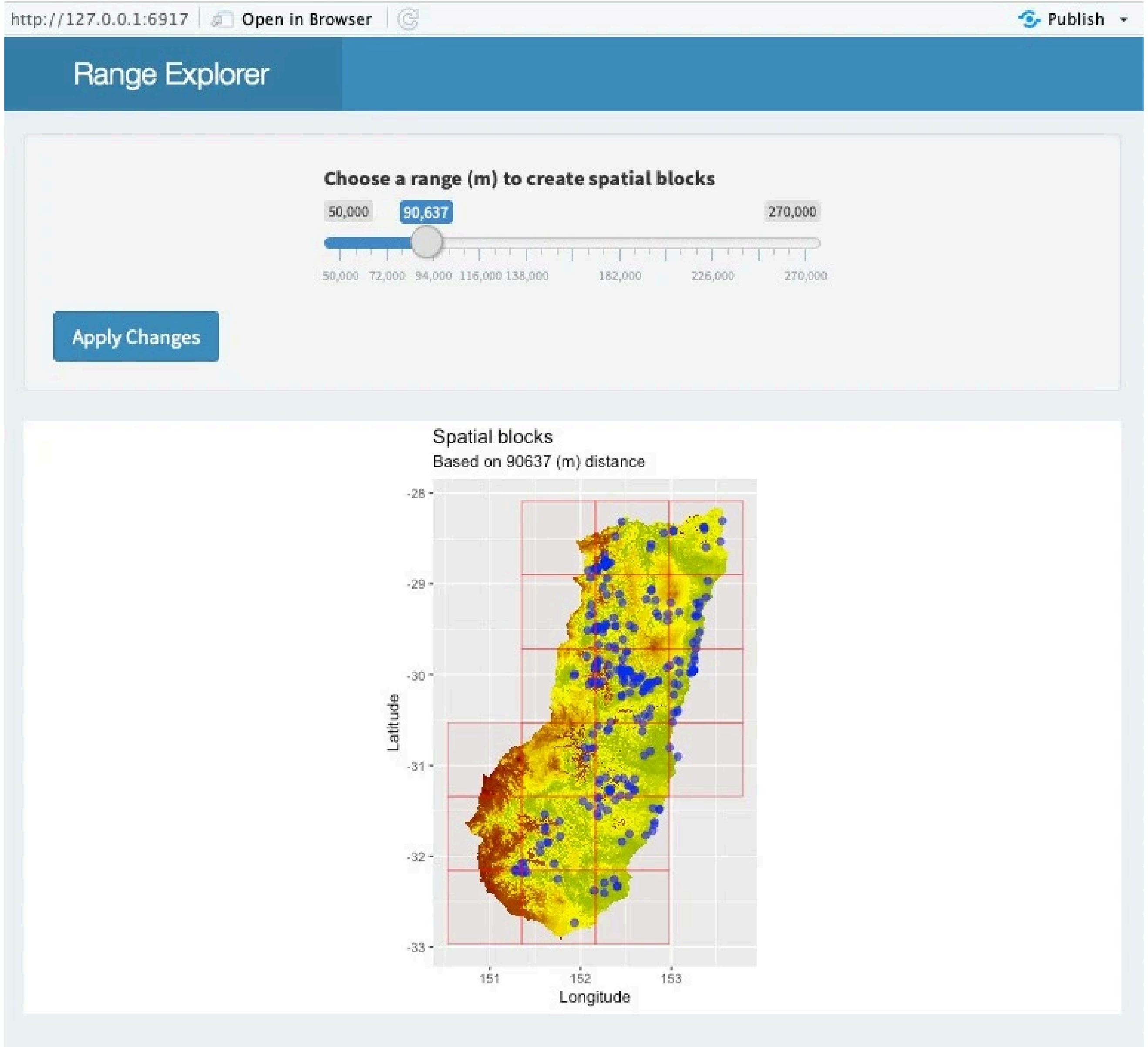
- **Blocks**: simple geometry; k folds; controlled **spatial scale** via `size`.
- **Clusters**: data-driven shapes; can be environmental; k folds; **balance** varies with `k`.
- **Buffers (LOO)**: strict separation at distance; **many** folds; heavy compute but principled.
- **NNDM (LOO)**: distance-distribution matching to target domain; **data- and domain-aware**.

Choosing a distance: `cv_spatial_autocor`

Goal. Estimate the **effective range of spatial autocorrelation** to inform block/buffer size.



Interactive block-size chooser: `cv_block_size`



Checking environmental similarity: `cv_similarity`

Why. Environmental novelty in test folds can bias evaluation.
What. Compute **MESS** or **L1/L2** distances between train and test environments.

```
cv_similarity(cv = ecv, x = pa_data, r = rasters, method = "MESS")
```

Negative MESS suggests extrapolation — report it and consider adjusting folds.

Practical setup & evaluation checklist

- **Project CRS** defined; distances in **metres**.
- **Pick a method** (blocks / clusters / buffer / NNDM) aligned to your task and domain.
- **Choose size** via autocorrelation or domain knowledge.
- **Balance**: tune `k` and `iteration` (blocks), or `k` (clusters).
- **Avoid leakage**: standardise features **within each train fold** only.
- **Aggregate metrics** across folds; report distributions, not just means.
- **Visualise & document**: include fold maps in your reports.

References

- **blockCV tutorials** (vignettes): introduction & SDM examples.
  - Tutorial 1: *How to create block cross-validation folds*.
  - Tutorial 2: *Block cross-validation for species distribution modelling*.
- Valavi R., Elith J., Lahoz-Monfort J., Guillera-Arroita G. (2019). *Methods in Ecology & Evolution* 10:225–232.
- Figures used on selected slides come from blockCV docs (README/tutorials) and the *Methods Blog* post introducing blockCV.