

DataSplits.jl: Data splitting strategies for machine learning in Julia

Davide Crucitti ^{1,2}, Carlos Pérez Míguez ¹, and Adrián Mosquera Orgueira ^{1,3}

1 Group of Computational Genomics and Hematology (GrHeCoXen), Health Research Institute of Santiago de Compostela (IDIS), Santiago de Compostela, Spain **2** Departamento de Farmacología, Farmacia y Tecnología Farmacéutica, Universidade de Santiago de Compostela, Santiago de Compostela, Spain **3** Division of Hematology, Complexo Hospitalario Universitario de Santiago de Compostela, SERGAS, Santiago de Compostela, Spain

DOI: [10.21105/joss.11066](https://doi.org/10.21105/joss.11066)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Matt Graham](#)  

Reviewers:

- [@MohamedLaghdafHABIBOULLAH](#)
- [@jonocarroll](#)

Submitted: 02 June 2026

Published: 30 October 2026

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

DataSplits.jl is a Julia package ([Bezanson et al., 2017](#)) for constructing training, validation, test, and cross-validation partitions in scientific machine learning workflows. While random train/test splits and standard k-fold cross-validation are widely available in machine learning libraries, many scientific applications require more controlled splitting procedures. The package exposes a single entry point:

```
res = partition(data, strategy; kwargs...)
```

Supported strategies include distance-based representative sample selection for calibration modelling, group-aware validation to avoid leakage between related observations, and time-aware validation for temporally ordered data. The package is lightweight and interoperable with MLUtils.jl, Flux.jl, and MLJ.jl. It can be installed from the Julia General Registry with] add DataSplits.

Statement of need

The choice of data partitioning strategy can strongly affect the reliability of model evaluation, particularly in small-data scientific domains such as chemometrics, spectroscopy, metabolomics, materials informatics, and quantitative structure–property (QSPR) or structure–activity (QSAR) modelling. In these settings, independent and identically distributed (i.i.d.) random splits may be inappropriate: samples can be highly structured, class distributions may be imbalanced, repeated measurements may induce group leakage, and calibration sets may need to span the relevant feature and response spaces ([Kohavi, 1995](#)). Representative sample selection algorithms such as Kennard–Stone/CADEX ([Kennard & Stone, 1969](#)) and SPXY/MDKS ([Galvão et al., 2005](#); [Sapto et al., 2012](#)) are commonly used in chemometrics to construct calibration and validation sets with broad coverage of the experimental domain. Related approaches include the Duplex algorithm ([Snee, 1977](#)), which selects both training and test sets simultaneously using the maximin criterion; OptiSim and dissimilarity-based selection methods ([Clark, 1997](#)); the Onion method ([Gallagher & O’Sullivan, 2020](#)) and its joint X–y extension, XYOnion ([Ezenarro, 2025](#)); electrostatic field-strength selection ([He et al., 2026](#)); spectral clustering-based splits ([Ng et al., 2001](#)); and classification-oriented adaptations such as the Morais–Lima–Martin algorithm ([Lelis Medeiros de Morais et al., 2019](#)). Time-series and financial machine learning workflows call for chronological splitting strategies such as purged k-fold cross-validation and combinatorial purged cross-validation ([López de Prado, 2018](#)), and blocked cross-validation ([Bergmeir & Benítez, 2012](#); [Roberts et al., 2017](#)). Target-aware

cross-validation strategies such as Venetian blinds (Næs et al., 2002) assign samples sorted by response value across folds in round-robin order, guaranteeing uniform target coverage without binning. To our knowledge, these methods are not available together in any general-purpose Julia library.

DataSplits.jl collects them in a single package with a consistent interface, for researchers who need reproducible, inspectable, and domain-appropriate partitions. For example, Kennard–Stone selection constructs a training set that spans the descriptor space, which random splits cannot guarantee (Figure 1).



Figure 1: Random and Kennard–Stone training-set selection on a synthetic dataset with a dense centre and a sparse boundary. Gray points denote all samples ($N=120$) and blue points the training set ($n=35$). Kennard–Stone covers the boundary of the experimental domain, whereas random selection concentrates in the centre.

State of the field

The closest Python equivalents are scikit-learn (Pedregosa et al., 2018) and astartes (Burns et al., 2023). Scikit-learn provides group-aware and time-series CV strategies (GroupKFold, TimeSeriesSplit, StratifiedGroupKFold) but does not implement distance-based selection methods such as Kennard–Stone, SPXY, or OptiSim. Astartes fills that gap for molecular and materials data, offering Kennard–Stone and several dissimilarity-based splitters, but does not cover group-aware CV, time-series CV, or nested cross-validation. DataSplits.jl covers both families: distance-based selection and a broad catalogue of standard CV strategies. On the samplers the two packages share, DataSplits.jl is 1.2 to 27 times faster than astartes and uses less peak memory (Table 1). Both packages precompute the full distance matrix for distance-based samplers, which limits the dataset size that fits in memory; DataSplits.jl also provides lazy variants that allow working with data of any size.

Table 1: Median runtime and peak memory of DataSplits.jl and astartes on identical synthetic inputs with 20 features. Scripts and results are in benchmark/experiment/.

Sampler	N	DataSplits.jl (ms)	astartes (ms)	Speed- up	DataSplits.jl (GiB)	astartes (GiB)
Kennard–Stone	10,000	389	1,126	2.9	0.75	1.58
SPXY	10,000	987	1,555	1.6	1.48	2.33

Sampler	N	DataSplits.jl (ms)	astartes (ms)	Speed- up	DataSplits.jl (GiB)	astartes (GiB)
OptiSim	10,000	329	1,098	3.3	0.75	1.12
Sphere exclusion	10,000	545	676	1.2	0.75	1.49
Random	100,000	0.45	12.1	27	0.02	0.04
Target property (high)	100,000	2.9	56.2	19	0.02	0.05
Target property (low)	100,000	3.0	51.1	17	0.02	0.05
Time-based	100,000	6.7	21.1	3.2	0.02	0.04

Within Julia, MLUtils.jl provides an observation interface based on numobs, getobs, and lazy observation views, together with data loaders and basic splitting utilities (JuliaML, 2022). Flux.jl re-exports MLUtils data-loading functionality for neural network training (Innes, 2018). MLJ.jl provides a model-evaluation framework with holdout and cross-validation resampling, and also accepts explicit train/test row pairs (Blaom et al., 2020). None of these three implement distance-based, group-stratified, or purged time-series splitting strategies. The closest Julia package in scope is ChemometricsTools.jl (Kneale, 2021), a broad chemometrics toolkit offering Kennard–Stone selection, two-way SplitByProportion and VenetianBlinds splits, and KFoldValidation/LeaveOneOut iterators, all returning materialised arrays, but no group-stratified or purged time-series splitting.

DataSplits.jl complements these packages, its core contribution is split construction rather than model evaluation or data loading; keeping it lightweight lets the same split definitions be reused across MLJ, Flux, MLUtils, and custom workflows.

Software design

The central abstraction in DataSplits.jl is a splitting strategy. Each strategy is represented by a concrete subtype of AbstractSplitStrategy, while cross-validation strategies subtype AbstractCVStrategy. Users call partition(data, strategy; kwargs...) to produce an AbstractSplitResult (a two-way, three-way, or per-fold cross-validation split). A single entry point lets user code switch strategies without changing call sites. Table 2 lists the strategy families.

Table 2: Splitting strategy families implemented in DataSplits.jl.

Strategy family	Examples	Typical use
Distance-based	KennardStoneSplit, SPXYSplit, OptiSimSplit, DuplexSplit, OnionSplit, FieldStrengthSplit, SpectralSplit	Calibration/test design in chemometrics and descriptor spaces
Target-aware	TargetPropertySplit, StratifiedKFold, StratifiedShuffleSplit, XYOnionSplit, VenetianBlindsCV	Class-balanced or response-aware validation
Group-aware	GroupShuffleSplit, GroupKFold, StratifiedGroupKFold	Avoiding leakage across related observations
Time-aware	TimeSplit, BlockedCV, PurgedKFold, CombinatorialPurgedKFold	Temporally ordered validation
Resampling	BootstrapSplit, ShuffleSplit, RepeatedKFold, NestedCV	Repeated performance estimation and hyperparameter tuning

The package exposes stable accessors (e.g. trainindices), so downstream code does not need to depend on internal result fields. Exposing indices rather than only data subsets lets the

same split be applied to multiple aligned arrays (features, labels, metadata) without re-running the selection algorithm. Data can be extracted either as lazy MLUtils-compatible observation views or as materialised subsets.

All distance-based strategies have a lazy counterpart that computes distances on-the-fly in $O(N)$ peak memory instead of precomputing the full $O(N^2)$ distance matrix. This makes the strategies applicable to datasets where the full matrix would not fit in RAM, at the cost of increased runtime. At $N = 10,000$ the lazy variants allocate 236 KiB where the eager ones allocate 763 MiB, for a 1.4 to 2.4 times increase in runtime. Under a 4 GiB budget this raises the largest feasible sample count from 10,000 to at least 100,000.

The package is designed to be extensible. Adding a custom strategy requires subtyping `AbstractSplitStrategy`, declaring which auxiliary slots (`:data`, `:target`, `:time`, `:groups`) the strategy reads via the `consumes` trait, and implementing a single `_partition` method. The partition entry point handles cohort-size resolution, slot validation, and feature-matrix conversion automatically.

Ecosystem integration is kept lightweight: lazy views can be passed directly to Flux/MLUtils data loaders, and `rowpairs` converts any split result into the `(train_indices, test_indices)` pairs accepted by MLJ's `evaluate!`, without a dependency on MLJ.jl.

Research impact statement

DataSplits.jl has been developed to support benchmarking and validation workflows in scientific machine learning where split construction directly affects reported performance estimates. The package underpins a published companion study ([Crucitti & Mosquera Orgueira, 2026](#)), which demonstrates empirically that split strategy choice can dominate differences between modelling algorithms in structured chemical datasets, with random splits systematically producing more optimistic performance estimates than distance-based and group-aware alternatives. A reproducible tutorial (`notebooks/tutorial.jl`) compares random, Kennard–Stone, SPXY, and group-aware splits on controlled datasets, quantifying feature-space coverage and group leakage, and generates [Figure 1](#). A benchmark suite (`benchmark/benchmarks.jl`) tracks runtime scaling of all strategies. DataSplits.jl will be maintained by the GrHeCo-Xen group through the public GitHub repository, where users can report issues, request features, and contribute new splitting strategies.

AI usage disclosure

Generative AI tools, including Claude Code (Anthropic), were used to assist with software development — specifically in the integration of some algorithms — documentation drafting, and drafting and language editing of this paper. AI tools were not used to make autonomous design, implementation, or scientific decisions. All AI-assisted code was thoroughly verified against independent reference implementations and covered by the test suite. All AI-assisted text was reviewed and edited by the authors, and technical claims were checked against the implementation, tests, documentation, and cited literature. The authors are responsible for the final content of the manuscript, code, and documentation.

Acknowledgements

This research project was made possible through the access granted by the Galician Supercomputing Center (CESGA) to its supercomputing infrastructure. The supercomputer FinisTerae III and its permanent data storage system have been funded by the NextGeneration EU 2021 Recovery, Transformation and Resilience Plan, ICT2021-006904, and also from the Pluriregional Operational Programme of Spain 2014-2020 of the European Regional

Development Fund (ERDF), ICTS-2019-02-CESGA-3, and from the State Programme for the Promotion of Scientific and Technical Research of Excellence of the State Plan for Scientific and Technical Research and Innovation 2013-2016 State subprogramme for scientific and technical infrastructures and equipment of ERDF, CESG15-DE-3114.

References

- Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191, 192–213. <https://doi.org/10.1016/j.ins.2011.12.028>
- Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1), 65–98. <https://doi.org/10.1137/141000671>
- Blaom, A., Kiraly, F., Lienart, T., Simillides, Y., Arenas, D., & Vollmer, S. (2020). MLJ: A Julia package for composable machine learning. *Journal of Open Source Software*, 5(55), 2704. <https://doi.org/10.21105/JOSS.02704>
- Burns, J. W., Spiekermann, K. A., Bhattacharjee, H., Vlachos, D. G., & Green, W. H. (2023). Machine Learning Validation via Rational Dataset Sampling with astartes. *Journal of Open Source Software*, 8(91), 5996. <https://doi.org/10.21105/JOSS.05996>
- Clark, R. D. (1997). OptiSim: An Extended Dissimilarity Selection Method for Finding Diverse Representative Subsets. *Journal of Chemical Information and Computer Sciences*, 37(6), 1181–1188. <https://doi.org/10.1021/CI970282V>
- Crucitti, D., & Mosquera Orgueira, A. (2026). Train-test splits matter more for evaluation than for performance. *Journal of Cheminformatics*. <https://doi.org/10.1186/s13321-026-01266-7>
- Ezenarro, J. (2025). XYOnion: A layer-based method for splitting datasets into calibration and validation subsets. *Analytica Chimica Acta*, 1364, 344229. <https://doi.org/10.1016/j.aca.2025.344229>
- Gallagher, N., & O’Sullivan, D. (2020). *Selection of Representative Learning and Test Sets Using the Onion Method*. <https://doi.org/10.13140/RG.2.2.25372.67202>
- Galvão, R. K. H., Araújo, C., José, G. E., Pontes, M. J. C., Silva, E. C., & Saldanha, T. C. B. (2005). A method for calibration and validation subset partitioning. *Talanta*, 67(4), 736–740. <https://doi.org/10.1016/J.TALANTA.2005.03.025>
- He, Z., Sun, J., Zhang, Y., & Zhang, X. (2026). Field Strength Distribution-Based Sample Selection Method. *Journal of Chemometrics*, 40(1), e70094. <https://doi.org/10.1002/cem.70094>
- Innes, M. (2018). Flux: Elegant machine learning with Julia. *Journal of Open Source Software*, 3(25), 602. <https://doi.org/10.21105/JOSS.00602>
- JuliaML. (2022). *MLUtils.jl: Machine learning utility functions for Julia*. <https://github.com/JuliaML/MLUtils.jl>
- Kennard, R. W., & Stone, L. A. (1969). Computer Aided Design of Experiments. *Technometrics*, 11(1), 137–148. <https://doi.org/10.1080/00401706.1969.10490666>
- Kneale, C. (2021). *ChemometricsTools.jl: A collection of tools for chemometrics and machine learning written in Julia* (Version v0.5.15). <https://github.com/caseykneale/ChemometricsTools.jl>
- Kohavi, R. (1995). *A study of cross-validation and bootstrap for accuracy estimation and model selection*. *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, 1137–1143. ISBN: 1-55860-363-8
- Lelis Medeiros de Moraes, C. de, Santos, M. C. D., Lima, K. M. G., & Martin, F. L. (2019).

- Improving data splitting for classification applications in spectrochemical analyses employing a random-mutation Kennard-Stone algorithm approach. *Bioinformatics*, 35(24), 5257–5263. <https://doi.org/10.1093/BIOINFORMATICS/BTZ421>
- López de Prado, M. M. (2018). *Advances in financial machine learning*. Wiley. ISBN: 978-1-119-48208-6
- Næs, T., Isaksson, T., Fearn, T., & Davies, T. (2002). *A user-friendly guide to multivariate calibration and classification*. NIR Publications. <https://doi.org/10.1255/978-1-906715-25-0>
- Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Proceedings of the 15th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 849–856. <https://proceedings.neurips.cc/paper/2001/hash/801272ee79cfde7fa5960571fee36b9b-Abstract.html>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2018). *Scikit-learn: Machine Learning in Python*. <https://doi.org/10.48550/arXiv.1201.0490>
- Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J. J., Schröder, B., Thuiller, W., Warton, D. I., Wintle, B. A., Hartig, F., & Dormann, C. F. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography*, 40(8), 913–929. <https://doi.org/10.1111/ECOG.02881>
- Sapto, A., Tadé, M. O., & Vuthaluru, H. (2012). A Modified Kennard-Stone Algorithm for Optimal Division of Data for Developing Artificial Neural Network Models. *Chemical Product and Process Modeling*, 7(1). <https://doi.org/10.1515/1934-2659.1645>
- Snee, R. D. (1977). Validation of Regression Models: Methods and Examples. *Technometrics*, 19(4), 415–428. <https://doi.org/10.1080/00401706.1977.10489581>