# UBayFS: An R Package for User Guided Feature Selection

**Anna Jenul** [1*] **and Stefan Schrunner** [1*]

**1** Norwegian University of Life Sciences, Ås, Norway * These authors contributed equally.

**DOI:** 10.21105/joss.04848

**Software**

- Review
- Repository
- Archive

**Editor:** Øystein Sørensen

**Reviewers:**

- @dhvalden
- @aaronpeikert
- @EugeneHao

**Submitted:** 01 October 2022
**Published:** 27 January 2023

## Statement of Need

Feature selection, also known as variable selection in statistics, is the process of selecting important variables (features) from a list of variables in a dataset. When training predictive models, the intention behind removing the least informative features from a dataset beforehand is (a) to reduce the computational burden and mathematical limitations associated with the curse of dimensionality, and (b) to increase interpretability of the model by allowing the user to obtain insights into the relevant input variables. In particular, feature selection speeds up the training process of machine learning models, especially when the dataset is high-dimensional.

The R (R Core Team, 2022) package UBayFS implements the user-guided framework for feature selection proposed in Jenul et al. (2022), which incorporates information from the data and prior knowledge from domain experts. Figure 1 demonstrates the framework. Different approaches for integrating prior knowledge in feature selection exist, though there is a lack of general and sophisticated frameworks that deliver stable and reproducible feature selection along with implementations. With its generic setup and the possibilities to specify prior weights as well as side constraints, UBayFS shows the flexibility to be applied in a broad range of application scenarios, which exceed the capabilities of conventional feature selectors while preserving large model generality. Besides side constraints, such as the option to specify a maximum number of features, the user can add must-link constraints (features must be selected together) or cannot-link constraints (features must not be selected together). In addition, constraints can be defined on feature-block level, as well. Thus, UBayFS is also capable of solving more general problems such as block feature selection. A parameter $\rho$ regulates the shape of a penalty term accounting for side constraints, where feature sets that violate constraints lead to a lower target value. State-of-the-art methods do not cover such scenarios.

The presented R package UBayFS provides an implementation along with an interactive Shiny dashboard, which makes feature selection available to R-users with different levels of expertise. The implementation allows the user to define their own feature selectors via a function interface or to use one out of three state-of-the-art feature selectors for building the generic ensemble of feature selectors covering the data-driven component of UBayFS. State-of-the-art choices include:

- Laplacian score
- Fisher score
- mRMR

R offers multiple packages implementing feature selection methodology. To name a few, `caret` (Kuhn, 2022) is an essential machine learning repository, containing models with built-in feature selection such as tree based methods (for instance `rpart2`), regularized approaches like `lasso`, and non-integrated feature selectors such as recursive feature elimination `rfe`. Other examples are the `Boruta` (Kursa & Rudnicki, 2010) package implementing the `Boruta` feature selector or the `GSelection` (Majumdar et al., 2019) package containing `hsic lasso` feature selection. All feature selectors available in R can be used as underlying ensemble feature selectors in UBayFS. Prior weights can be specified for single features or whole blocks as weight vectors. Linear side constraints are implemented via a matrix $A$ and a right side $b$ or with a customized function for specific constraint types. Hence, the sophisticated statistical model is summarized in a user-friendly and easy-to-use package.

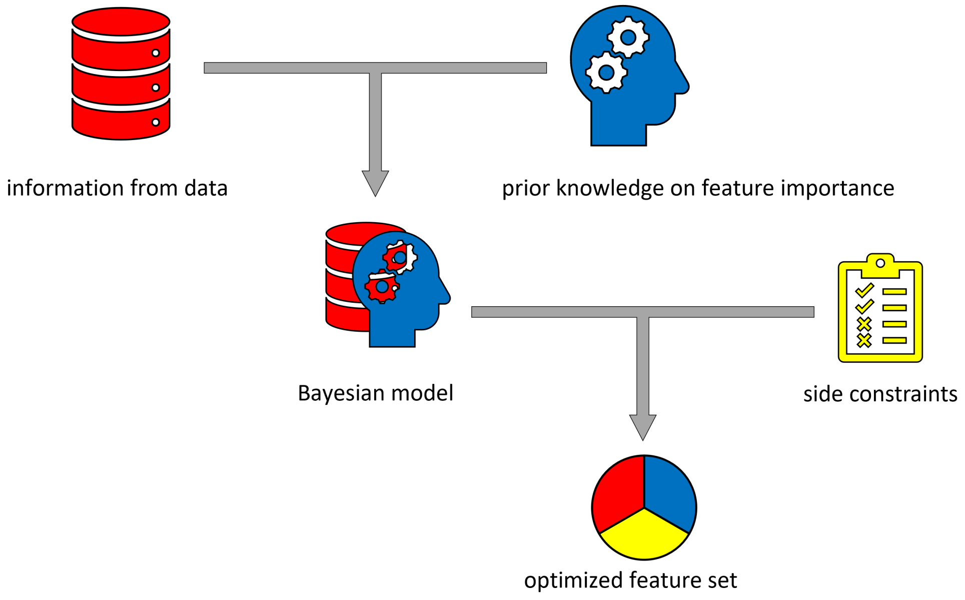


**Figure 1:** At first, UBayFS elaborates information directly from data via ensemble feature selection. This information is merged with prior expert knowledge (a-priori feature weights) in a Bayesian model framework. Additionally, the user can include further side constraints such as a maximum number of features or cannot-link constraints between features. The final step comprises the optimization with respect to the model's utility function, including the side constraints.

## Concept of UBayFS

As described in Jenul et al. (2022), UBayFS is a Bayesian ensemble feature selection framework. The methodology is based on quantifying a random variable $\theta$, representing feature importances, given evidence collected from the data, denoted as $y$. In particular, $y$ counts the number of elementary models in the generic ensemble of feature selectors, which select a particular feature. Statistically, we interpret the result from each elementary feature selector as a realization from a multinomial distribution with parameters $\theta$, where $\theta \in [0,1]^N$ defines the success probabilities of sampling each feature in an individual feature selection and thus the success probability in the ensemble. Both sources of information are combined using Bayes' Theorem:

$$p(\theta|y) \propto p(y|\theta) \cdot p(\theta).$$

In the framework of UBayFS, $p(y|\theta)$ represents the data-driven component (implemented via a multinomial likelihood), while $p(\theta)$ describes the user knowledge part modeled with a Dirichlet distribution. Due to the conjugate prior property of the Dirichlet distribution, the posterior parameter update has a tractable form and can be computed analytically. Side constraints are represented by a system of linear inequalities $A \cdot \delta - b \leq 0$, where $A \in \mathbb{R}^{K \times N}$, $b \in \mathbb{R}^K$, and $0 \in \mathbb{R}^K$ is the $K$-dimensional vector of zeros. $K$ is defined as the total number of constraints. The comparison is performed elementwise.

In UBayFS, a relaxed inadmissibility function $\kappa_{k,\rho}(\delta)$ is used as a penalization for the violation of a given side constraint $k = 1, ..., K$. The joint inadmissibility function $\kappa$ pursues the idea that $\kappa = 1$ (maximum penalization) if at least one $\kappa_{k,\rho} = 1$, while $\kappa = 0$ (no penalization) if all $\kappa_{k,\rho} = 0$. A more detailed description is provided in the original paper (Jenul et al., 2022).

To obtain an optimal feature set $\delta^{\star}$, we use a target function $U(\delta, \theta)$ which represents a posterior expected utility of feature sets $\delta$ given the posterior feature importance parameter $\theta$, regularized by the inadmissibility function $\kappa(.)$

$$\delta^{\star} = \max_{\delta \in \{0,1\}^N} \left( \mathbb{E}_{\theta|y}[U(\delta, \theta(y))] \right) = \max_{\delta \in \{0,1\}^N} \left( \delta^T \mathbb{E}_{\theta|y}[\theta(y)] - \lambda \kappa(\delta) \right).$$

The optimization is implemented via a genetic algorithm along with a greedy algorithm for initialization, suggested by Jenul et al. (2022) to find a proper start vector for the optimization.

## Package Summary

The function `build.UBaymodel()` initializes an S3 class object UBaymodel and computes the ensemble of elementary feature selectors. In the current version, linear feature selectors such as Fisher score, Laplacian score (You & Shung, 2022), and mRMR (Jay et al., 2013) are supported as integrated options. Any arbitrary feature selector can be defined manually and used as input. In addition, the number of elementary models $M$ is specified. The user can directly set prior weights inside the `build` function. Constraints are either provided as a matrix $A$ and a right side $b$, or built using the `buildConstraints()` function, which supports max-size, must-link, and cannot-link constraints on both feature and block level. UBayFS requires at least one constraint limiting the total number of features to be selected ("max-size"). The level of constraint-relaxation is steered with an input parameter $\rho$. In addition, the weights for single features or feature blocks are set with `setWeights()`.

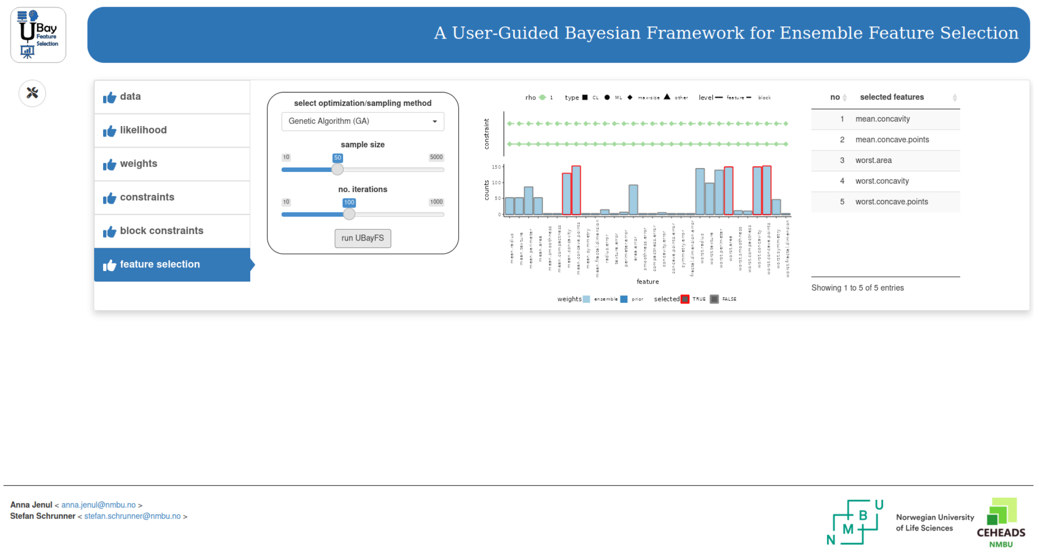
The function `admissibility()` allows the user to evaluate the penalty term for a given feature set under a set of constraints. After initializing the model and computing the ensembles, the `train.UBaymodel()` function optimizes the feature set via a genetic algorithm (Scrucca, 2013) with greedy initialization. According to empirical evaluations, the greedy initialization decreases the runtime and leads to faster convergence towards an optimal feature set. Finally, the package implements the generic functions `print.UBaymodel()`, `plot.UBaymodel()`, and `summary.UBaymodel()` as well as an evaluation function `evaluteFS()` to report and visualize results. Two vignettes guide the user through the package and demonstrate how the method can be deployed in common application scenarios, including how user knowledge is specified and how feature- and block-wise constraints are set.

## Interactive Shiny Dashboard

The function `runInteractive()` opens an interactive Shiny dashboard allowing the user to load and analyze data interactively. However, due to computational limitations, it is not recommended to use the HTML interface for larger datasets ($> 100$ features or $> 1000$ samples). Instead, functions should be called from the `R` console in such cases. Figure 2 shows the dashboard with the different tabs:

- **data**: Load the dataset and specify whether row names, column names, or a block structure is present. A demo dataset is ready to be loaded and used for a first touch on the package.
- **likelihood**: Select elementary feature selectors for ensemble feature selection, the number of models $M$, the number of features in each model, and the ratio of the train-test split. Further, the dashboard allows the user to mix different elementary feature selectors, although this option is not recommended due to limited stability (Seijo-Pardo et al., 2017).

- **weights**: The prior feature weights are set by the user. For block feature selection, it is possible to set weights for blocks; otherwise, for a single feature.
- **constraints / block constraints**: In this task, the user sets different constraints (at least a max-size constraint). The penalty $\rho$ can be varied here as well.
- **feature selection**: In the dashboard's last step, an optimization procedure determines the final feature set. A plot of the final result is produced - also, the model can be saved as an Rdata file and loaded to the dashboard again.


A User-Guided Bayesian Framework for Ensemble Feature Selection
data
likelihood
weights
constraints
block constraints
feature selection
run UBayFS
rho 1 type CL ML max-size other level feature block
constraint
counts
feature
weights ensemble prior selected TRUE FALSE
Norwegian University of Life Sciences
CEHEADS NMBU

select optimization/sampling method: Genetic Algorithm (GA)

sample size: 50 (10 – 5000)

no. iterations: 100 (10 – 1000)

| no | selected features |
|---|---|
| 1 | mean.concavity |
| 2 | mean.concave.points |
| 3 | worst.area |
| 4 | worst.concavity |
| 5 | worst.concave.points |

Showing 1 to 5 of 5 entries

Anna Jenul < anna.jenul@nmbu.no >
Stefan Schrunner < stefan.schrunner@nmbu.no >

**Figure 2:** Illustration of the Shiny HTML dashboard.

## Ongoing research

Based on the present UBayFS package, ongoing work focuses on the implementation of even more types of expert constraints and elementary feature selection models. Moreover, a Python package with similar functionality is planned for the future.

## Acknowledgements

We would like to thank Prof. Jürgen Pilz (University of Klagenfurt) and Prof. Oliver Tomic (Norwegian University of Life Sciences), who contributed to the development of the methodology and supported us with ideas and fruitful discussions, as well as Kristian Hovde Liland (Norwegian University of Life Sciences) for testing the R package.

## References

Jay, N. D., Papillon-Cavanagh, S., Olsen, C., El-Hachem, N., Bontempi, G., & Haibe-Kains, B. (2013). mRMRe: An R package for parallelized mRMR ensemble feature selection. *Bioinformatics*, *29*(18), 2365–2368. https://doi.org/10.1093/bioinformatics/btt383

Jenul, A., Schrunner, S., Pilz, J., & Tomic, O. (2022). A user-guided Bayesian framework for ensemble feature selection in life science applications (UBayFS). *Machine Learning*, *111*(10), 3897–3923. https://doi.org/10.1007/s10994-022-06221-9

Kuhn, M. (2022). *Caret: Classification and regression training*. https://CRAN.R-project.org/package=caret

Kursa, M. B., & Rudnicki, W. R. (2010). Feature selection with the Boruta package. *Journal of Statistical Software*, *36*(11). https://doi.org/10.18637/jss.v036.i11

Majumdar, S. G., Rai, A., & Mishra, D. C. (2019). *GSelection: Genomic selection*. https://CRAN.R-project.org/package=GSelection

R Core Team. (2022). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Scrucca, L. (2013). GA: A package for genetic algorithms in R. *Journal of Statistical Software*, *53*(4). https://doi.org/10.18637/jss.v053.i04

Seijo-Pardo, B., Porto-Díaz, I., Bolón-Canedo, V., & Alonso-Betanzos, A. (2017). Ensemble feature selection: Homogeneous and heterogeneous approaches. *Knowledge-Based Systems*, *118*, 124–139. https://doi.org/10.1016/j.knosys.2016.11.017

You, K., & Shung, D. (2022). Rdimtools: An R package for dimension reduction and intrinsic dimension estimation. *Software Impacts*, *14*, 100414. https://doi.org/10.1016/j.simpa.2022.100414