rjaf: Regularized Joint Assignment Forest with Treatment Arm Clustering

October 27, 2024

Summary

Learning the assignment of treatments is an omnipresent problem in economics and public health. It arises, for example, from randomized controlled trials where a variety of behavioral nudges (treatments) are developed to enhance vaccination uptake against coronavirus disease 2019 (COVID-19) or influenza, especially among racially or ethnically underrepresented and socioeconomically disadvantaged populations (Milkman et al. 2021, 2022; Dai et al. 2021). Subject-specific covariates containing information regarding sociodemographics, clinical characteristics, and comorbid conditions, if available, can be harnessed to identify personalized treatment assignment schemes. The rjaf package provides a user-friendly implementation of the regularized joint assignment forest (RJAF) (Ladhania et al. 2023), a forest-based treatment assignment algorithm featuring greedy recursive partitioning (Athey, Tibshirani, and Wager 2019), treatment and covariate resampling in bootstrap tree aggregating (Breiman 1996), outcome residualization and regularization, and k-means treatment arm clustering (Hartigan and Wong 1979). Personalized treatment learning is achieved through optimizing a regularized empirical analogue of the expected outcome. The integration of R (R Core Team 2024) and C++ (Stroustrup 2013) substantially boosts the computational efficiency in tree partitioning and aggregating. This package is especially suitable in randomized controlled trial settings where a large number of treatment arms are present.

Statement of Need

There is an ever-growing literature in the intersection of machine learning and causal inference attempting to address the problem of optimal treatment assignment through heterogeneous treatment effect estimation (Athey and Imbens 2016; Wager and Athey 2018; Hitsch and Misra 2018; Athey, Tibshirani, and Wager 2019; Sverdrup et al. 2020; Athey and Wager 2021). Other methods focus on maximizing the benefit (empirical welfare) from treatment assignment (e.g., Kitagawa and Tetenov 2018), or the chance of assigning an individual to an optimal treatment arm (e.g., Murphy 2005; Zhou, Wang, and Zeng 2018). Most of these methods perform well with a limited number of treatment and control groups. As more arms are present, the estimation of arm-specific empirical welfare and the identification of individual-specific optimal arms become increasingly difficult. Commonly used implementations such as the multi-arm causal forest (Tibshirani et al. 2022) and random forest (Wright and Ziegler 2017) lead to significantly suboptimal assignment with insufficient levels of empirical welfare. By contrast, the RJAF yields elevated empirical welfare closer to the optimal level from the oracle assignment than the multi-arm causal forest and random forest. Despite the methodological advantage over existing approaches, the incorporation of machine learning and causal inference techniques such as recursive tree partitioning, bootstrap aggregating, and treatment arm clustering makes it challenging to implement the RJAF from scratch even for well-trained data scientists. The rjaf is an open-source software package in R and C++ that efficiently implements the RJAF, offering data scientists a user-friendly analytic toolbox for learning personalized treatment rules in real-world settings.

Workflow

Figure 1 outlines the workflow for using the rjaf package to perform personalized treatment assignment and honest outcome estimation. The process begins with partitioning the input data–consisting of outcomes, treatment arms, covariates, individual identifiers, and optional probabilities of treatment assignment–into two parts, one for training and estimation, and the other for validation. The rjaf function first checks whether residualization should be performed via the residualize function, using the resid argument. If resid is set to TRUE (the default), a new column of residualized outcomes is added to the input data and used for tree growing on the training subset to reduce baseline variation. Next, the rjaf function evaluates whether treatment clustering should be performed on the training and estimation set during tree growing using the clus.tree.growing argument. If clus.tree.growing is TRUE, the rjaf_cpp function is employed to estimate cross-validated counterfactual outcomes, which are then subjected to k-means clustering. The optimal number of treatment clusters is determined using the elbow method. Lastly, the rjaf_cpp function is applied to the pre-processed data to obtain optimal treatment arms or clusters and predicted outcomes for all individuals in the validation set.

Figure 2 provides a detailed description of the rjaf_cpp function, which grows a large number of trees through the growTree function. growTree begins by taking the training and estimation set as input, splitting it into two separate training and estimation subsets proportionally by treatment arms or clusters. Initially, utility is set at the root node, where optional inverse probability weighting (IPW) can be applied. A tree is then grown via recursive partitioning of the training subset based on covariate splits. Each potential split is generated by the splitting function, where regularization tuned by the lambda1 parameter can be performed along with IPW to calculate average outcomes across treatment arms or clusters. A potential split is retained if it meets three criteria: (1) each child node contains at least the minimum number of units specified by the nodesize argument, (2) the utility gain is at least epi times the empirical standard deviation of outcomes in the entire input data, and (3) the child nodes have different optimal treatment arm or cluster assignments. Recursive partitioning ends when no potential split meets these criteria. Once terminal nodes are determined on the training subset, the same splitting rules applied to the estimation subset to obtain its terminal nodes. Outcomes from the estimation subset are then used to calculate treatment-specific average outcomes in each terminal node, with optional regularization tuned by lambda2 and imputation controlled by impute. The validation set undergoes the same splitting, with treatment-specific outcomes from the estimation subset assigned to corresponding terminal nodes to achieve honest outcome estimates, thus concluding the growTree function. The final step in rjaf_cpp is bootstrap aggregation of a large number of trees grown via growTree, where the total number of trees is set by the ntree parameter of the rjaf_cpp function.

As in the implementations of other forest-based methods, built-in hyper-parameter tuning (e.g., epi, lambda1, and lambda2) is not provided in the rjaf package. Interested users are referred to the caret package (Kuhn 2008) for details.

Availability

The rjaf package is publicly available on GitHub, where the use of the rjaf package, including installation instructions and an example, has been documented in the README.md file. The package is also available on the Comprehensive R Archive Network.

Acknowledgments

Wenbo Wu and Rahul Ladhania were partially supported by a research grant from the Robert Wood Johnson Foundation titled *Informing Strategies to Increase Use of COVID-19 and Flu Vaccines by Different Racial and Ethnic Groups to Improve Health Equity during Health Crises* (award number 78416).

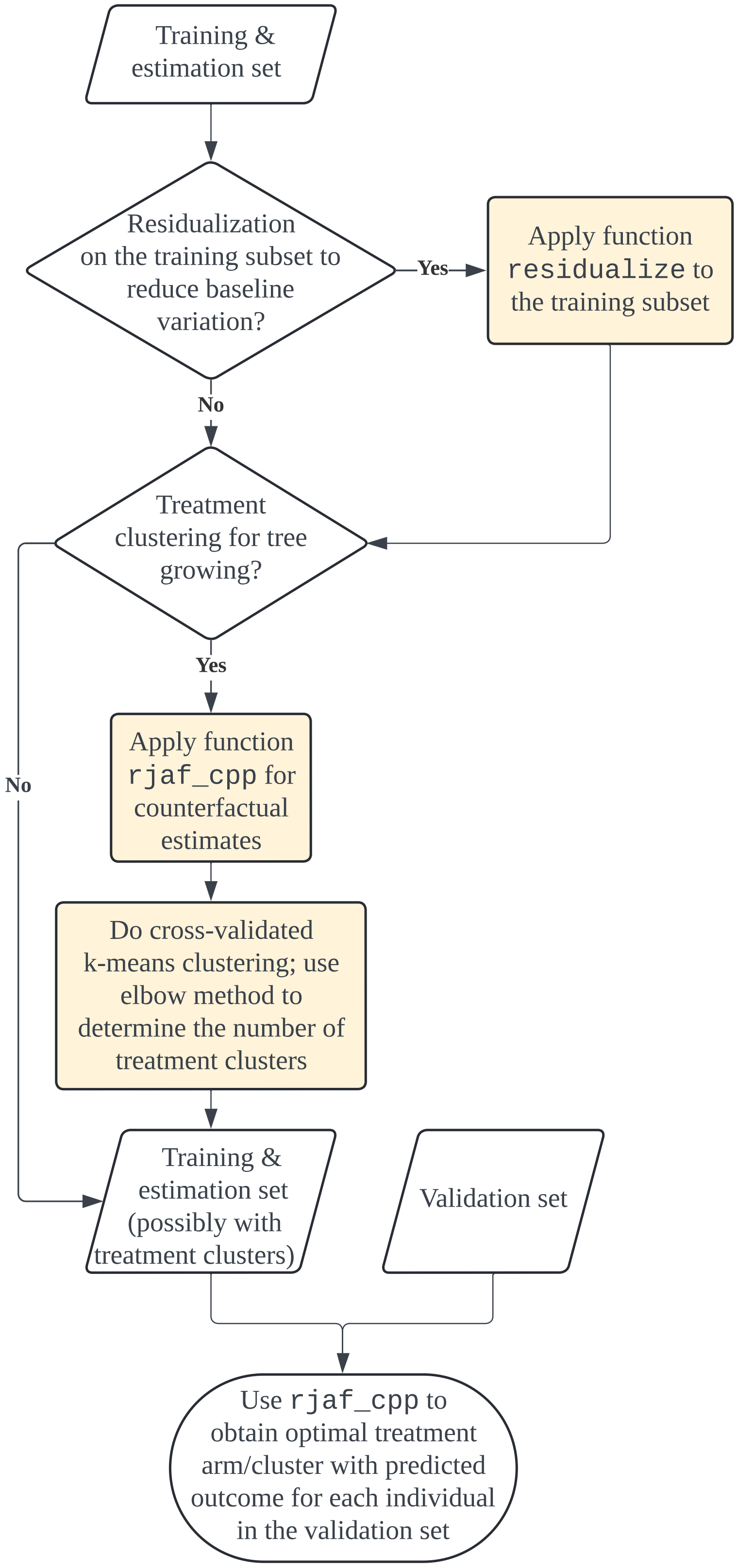
Training & estimation set
Residualization on the training subset to reduce baseline variation?
Yes
Apply function residualize to the training subset
No
Treatment clustering for tree growing?
Yes
No
Apply function rjaf_cpp for counterfactual estimates
Do cross-validated k-means clustering; use elbow method to determine the number of treatment clusters
Training & estimation set (possibly with treatment clusters)
Validation set
Use rjaf_cpp to obtain optimal treatment arm/cluster with predicted outcome for each individual in the validation set

Figure 1: A sketch of the `rjaf` package.

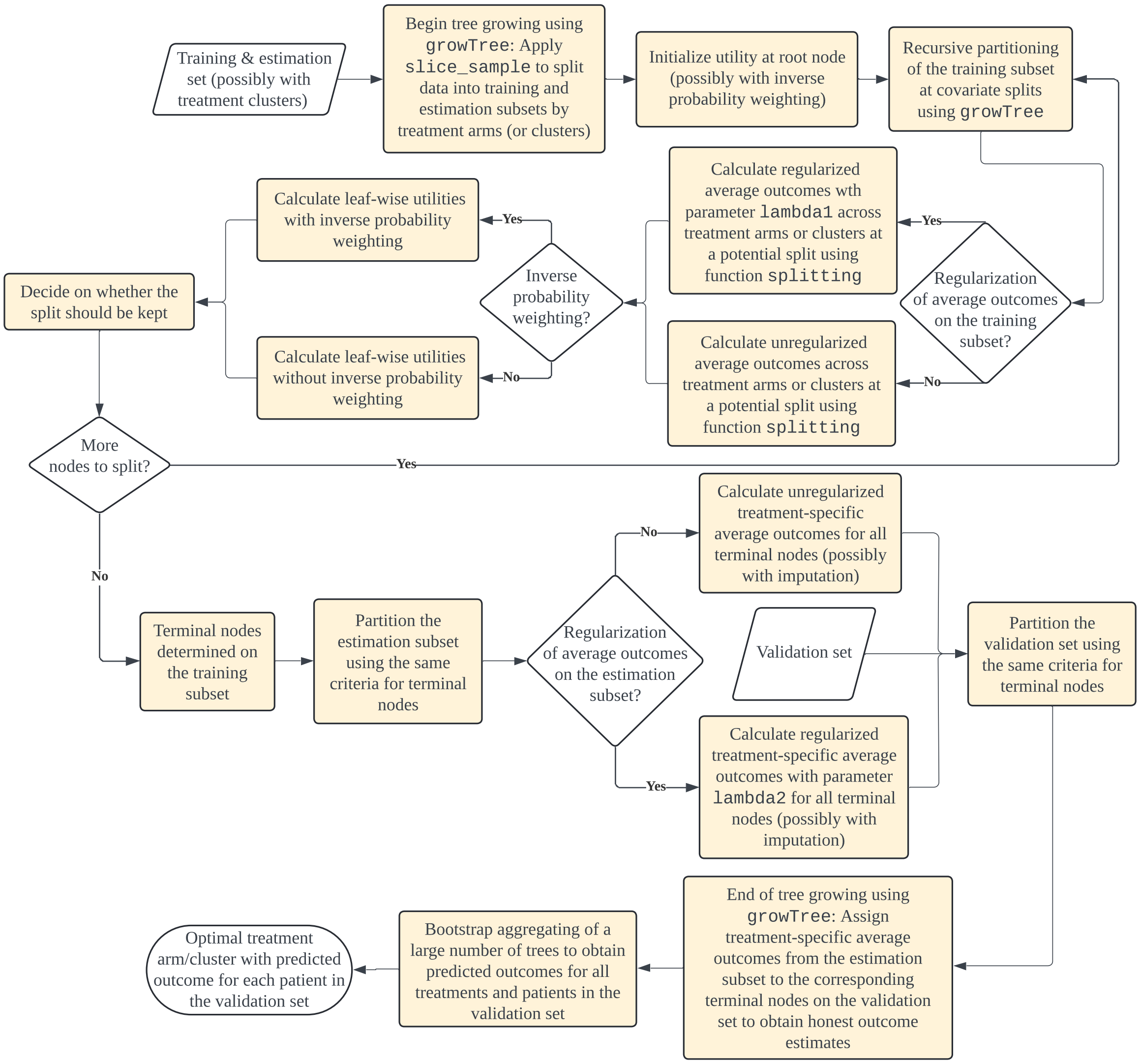
Training & estimation set (possibly with treatment clusters)
Begin tree growing using growTree: Apply slice_sample to split data into training and estimation subsets by treatment arms (or clusters)
Initialize utility at root node (possibly with inverse probability weighting)
Recursive partitioning of the training subset at covariate splits using growTree
Regularization of average outcomes on the training subset?
Yes
Calculate regularized average outcomes wth parameter lambda1 across treatment arms or clusters at a potential split using function splitting
No
Calculate unregularized average outcomes across treatment arms or clusters at a potential split using function splitting
Inverse probability weighting?
Yes
Calculate leaf-wise utilities with inverse probability weighting
No
Calculate leaf-wise utilities without inverse probability weighting
Decide on whether the split should be kept
More nodes to split?
Yes
No
Terminal nodes determined on the training subset
Partition the estimation subset using the same criteria for terminal nodes
Regularization of average outcomes on the estimation subset?
No
Calculate unregularized treatment-specific average outcomes for all terminal nodes (possibly with imputation)
Yes
Calculate regularized treatment-specific average outcomes with parameter lambda2 for all terminal nodes (possibly with imputation)
Validation set
Partition the validation set using the same criteria for terminal nodes
End of tree growing using growTree: Assign treatment-specific average outcomes from the estimation subset to the corresponding terminal nodes on the validation set to obtain honest outcome estimates
Bootstrap aggregating of a large number of trees to obtain predicted outcomes for all treatments and patients in the validation set
Optimal treatment arm/cluster with predicted outcome for each patient in the validation set

Figure 2: A description of the `rjaf_cpp` function.

References

Athey, Susan, and Guido Imbens. 2016. "Recursive Partitioning for Heterogeneous Causal Effects." *Proceedings of the National Academy of Sciences* 113 (27): 7353–60. https://doi.org/10.1073/pnas.1510489113.

Athey, Susan, Julie Tibshirani, and Stefan Wager. 2019. "Generalized Random Forests." *Annals of Statistics* 47 (2): 1148–78. https://doi.org/10.1214/18-AOS1709.

Athey, Susan, and Stefan Wager. 2021. "Policy Learning with Observational Data." *Econometrica* 89 (1): 133–61. https://doi.org/10.3982/ECTA15732.

Breiman, Leo. 1996. "Bagging Predictors." *Machine Learning* 24 (2): 123–40. https://doi.org/10.1007/BF00058655.

Dai, Hengchen, Silvia Saccardo, Maria A Han, Lily Roh, Naveen Raja, Sitaram Vangala, Hardikkumar Modi, Shital Pandya, Michael Sloyan, and Daniel M Croymans. 2021. "Behavioural Nudges Increase COVID-19 Vaccinations." *Nature* 597 (7876): 404–9. https://doi.org/10.1038/s41586-021-03843-2.

Hartigan, John A, and Manchek A Wong. 1979. "Algorithm AS 136: A k-Means Clustering Algorithm." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28 (1): 100–108. https://doi.org/10.2307/2346830.

Hitsch, Günter J, and Sanjog Misra. 2018. "Heterogeneous Treatment Effects and Optimal Targeting Policy Evaluation." *Available at SSRN*. https://doi.org/10.1007/s11129-023-09278-5.

Kitagawa, Toru, and Aleksey Tetenov. 2018. "Who Should Be Treated? Empirical Welfare Maximization Methods for Treatment Choice." *Econometrica* 86 (2): 591–616. https://doi.org/10.3982/ECTA13288.

Kuhn, Max. 2008. "Building Predictive Models in r Using the Caret Package." *Journal of Statistical Software* 28: 1–26. https://doi.org/10.18637/jss.v028.i05.

Ladhania, Rahul, Jann Spiess, Lyle Ungar, and Wenbo Wu. 2023. "Personalized Assignment to One of Many Treatment Arms via Regularized and Clustered Joint Assignment Forests." https://doi.org/10.48550/arXiv.2311.00577.

Milkman, Katherine L, Linnea Gandhi, Mitesh S Patel, Heather N Graci, Dena M Gromet, Hung Ho, Joseph S Kay, et al. 2022. "A 680,000-Person Megastudy of Nudges to Encourage Vaccination in Pharmacies." *Proceedings of the National Academy of Sciences* 119 (6): e2115126119. https://doi.org/10.1073/pnas.2115126119.

Milkman, Katherine L, Mitesh S Patel, Linnea Gandhi, Heather N Graci, Dena M Gromet, Hung Ho, Joseph S Kay, et al. 2021. "A Megastudy of Text-Based Nudges Encouraging Patients to Get Vaccinated at an Upcoming Doctor's Appointment." *Proceedings of the National Academy of Sciences* 118 (20): e2101165118. https://doi.org/10.1073/pnas.2101165118.

Murphy, Susan A. 2005. "A Generalization Error for q-Learning." *Journal of Machine Learning Research* 6 (37): 1073–97. https://www.jmlr.org/papers/volume6/murphy05a/murphy05a.pdf.

R Core Team. 2024. "R: A Language and Environment for Statistical Computing." Vienna, Austria: R Foundation for Statistical Computing; `https://www.R-project.org`.

Stroustrup, Bjarne. 2013. *The C++ Programming Language*. Fourth. Pearson Education. https://www.stroustrup.com/4th.html.

Sverdrup, Erik, Ayush Kanodia, Zhengyuan Zhou, Susan Athey, and Stefan Wager. 2020. "Policytree: Policy Learning via Doubly Robust Empirical Welfare Maximization over Trees." *Journal of Open Source Software* 5 (50): 2232. https://doi.org/10.21105/joss.02232.

Tibshirani, Julie, Susan Athey, Rina Friedberg, Vitor Hadad, David Hirshberg, Luke Miner, Erik Sverdrup, Stefan Wager, and Marvin Wright. 2022. "grf: Generalized Random Forests." https://CRAN.R-project.org/package=grf. https://doi.org/10.32614/cran.package.grf.

Wager, Stefan, and Susan Athey. 2018. "Estimation and Inference of Heterogeneous Treatment Effects Using Random Forests." *Journal of the American Statistical Association* 113 (523): 1228–42. https://doi.org/10.1080/01621459.2017.1319839.

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

Zhou, Xuan, Yuanjia Wang, and Donglin Zeng. 2018. "Sequential Outcome-Weighted Multicategory Learning for Estimating Optimal Individualized Treatment Rules." http://www.columbia.edu/~yw2016/SOM_ITR.pdf.