`rjaf`: Regularized Joint Assignment Forest with Treatment Arm Clustering

January 29, 2025

Summary

Learning optimal assignment of treatments—determining what works best for whom—is an important problem in economics, public health, and related fields, particularly when faced with a variety of treatment strategies. The problem arises, for example, in settings where randomized controlled trials (RCT) are conducted to evaluate various behavioral science-informed interventions aimed at fostering behavior change (Milkman, Gromet, et al. 2021). Such interventions have been studied across diverse domains, including encouraging gym attendance and increasing vaccine uptake for influenza or COVID-19 (Milkman, Gromet, et al. 2021; Milkman, Patel, et al. 2021; Dai et al. 2021; Milkman et al. 2022). While most studies focus on identifying interventions that perform best on average, this approach often overlooks effect heterogeneity. Ignoring heterogeneity can be a missed opportunity to tailor interventions for maximum effectiveness and may even exacerbate disparities (Bryan, Tipton, and Yeager 2021). Subject-specific covariates, such as sociodemographics, past behavior, clinical characteristics, and comorbidities, can be harnessed to identify which interventions work best for different segments of the population, allowing for more impactful intervention assignments. The `rjaf` package provides a user-friendly implementation of the regularized joint assignment forest (RJAF) (Ladhania et al. 2023), a regularized forest-type assignment algorithm based on greedy recursive partitioning (Athey, Tibshirani, and Wager 2019) that shrinks effect estimates across treatment arms. The algorithm is augmented by outcome residualization to reduce baseline variation, and employs a clustering scheme (Hartigan and Wong 1979) that combines treatment arms with consistently similar outcomes. Personalized treatment learning is achieved by optimizing a regularized empirical analogue of the expected outcome. The integration of `R` (R Core Team 2024) and `C++` (Stroustrup 2013) substantially boosts computational efficiency in tree partitioning and aggregating. It is especially suitable in RCT settings with numerous treatment arms and constrained sample sizes, making it a powerful tool for learning personalized intervention strategies.

Statement of Need

There is an ever-growing literature at 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 approaches perform well with a limited number of treatment and control groups. A large number of arms renders the identification of best arm increasingly difficult, and assignments based on separate arm-wise estimation are inefficient. Commonly used implementations such as the multi-arm causal forest (Tibshirani et al. 2022) and random forest (Wright and Ziegler 2017) might lead to suboptimal assignment, particularly in settings with high noise. By contrast, the RJAF (Ladhania et al. 2023) yields elevated empirical outcome estimates closer to the optimal level from the oracle assignment than the multi-arm causal forest approach in high noise settings, and performs at the same level in low-noise ones. 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 model training and estimation, and the other is the held-out set on which personalized assignment rules are obtained. The rjaf function first checks whether outcome residualization for reducing baseline variation 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 set. Next, the rjaf function evaluates whether treatment clustering should be performed on the training-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 for the $K+1$ treatment arms, after which k-means clustering is used to learn $M+1$ treatment arm clusters. The optimal number of treatment clusters is determined using the elbow method. After clustering, the rjaf_cpp function is reapplied to the preprocessed data, with assignment forest fitted on $M+1$ treatment clusters and counterfactual outcomes estimated for the original $K+1$ arms. If clus.tree.growing is FALSE, the rjaf_cpp function is employed to estimate counterfactual outcomes for the $K+1$ arms. Lastly, rjaf_cpp function is used to obtain optimal treatment arms and predicted counterfactual outcomes under all treatment arms for individuals in the held-out set.

Figure 2 provides a detailed description of the rjaf_cpp function, which grows a specified number of trees using the growTree function. growTree begins by taking the Training-Estimation data set as input, randomly splitting it into 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 specified by the lambda1 parameter can be performed along with IPW to calculate weighted average outcomes by 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 eps 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 from the parent node. Recursive partitioning ends when no potential split meets these criteria. Once terminal nodes are determined on the training subset, the learnt splitting rules are then applied to the estimation subset to obtain its terminal nodes. Outcomes from units in the estimation subset are then used to calculate treatment-arm-specific average outcomes for each terminal node, with optional regularization specified by the lambda2 parameter and imputation controlled by impute. On the held-out data set, treatment-arm-specific outcomes from the estimation subset are 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., eps, lambda1, and lambda2) is not provided in the rjaf package. Interested users are referred to the caret package (Kuhn 2008) for details.

Quick Start

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. This section provides an introduction to the basics of rjaf. One can install and load the rjaf package by executing the following R commands:

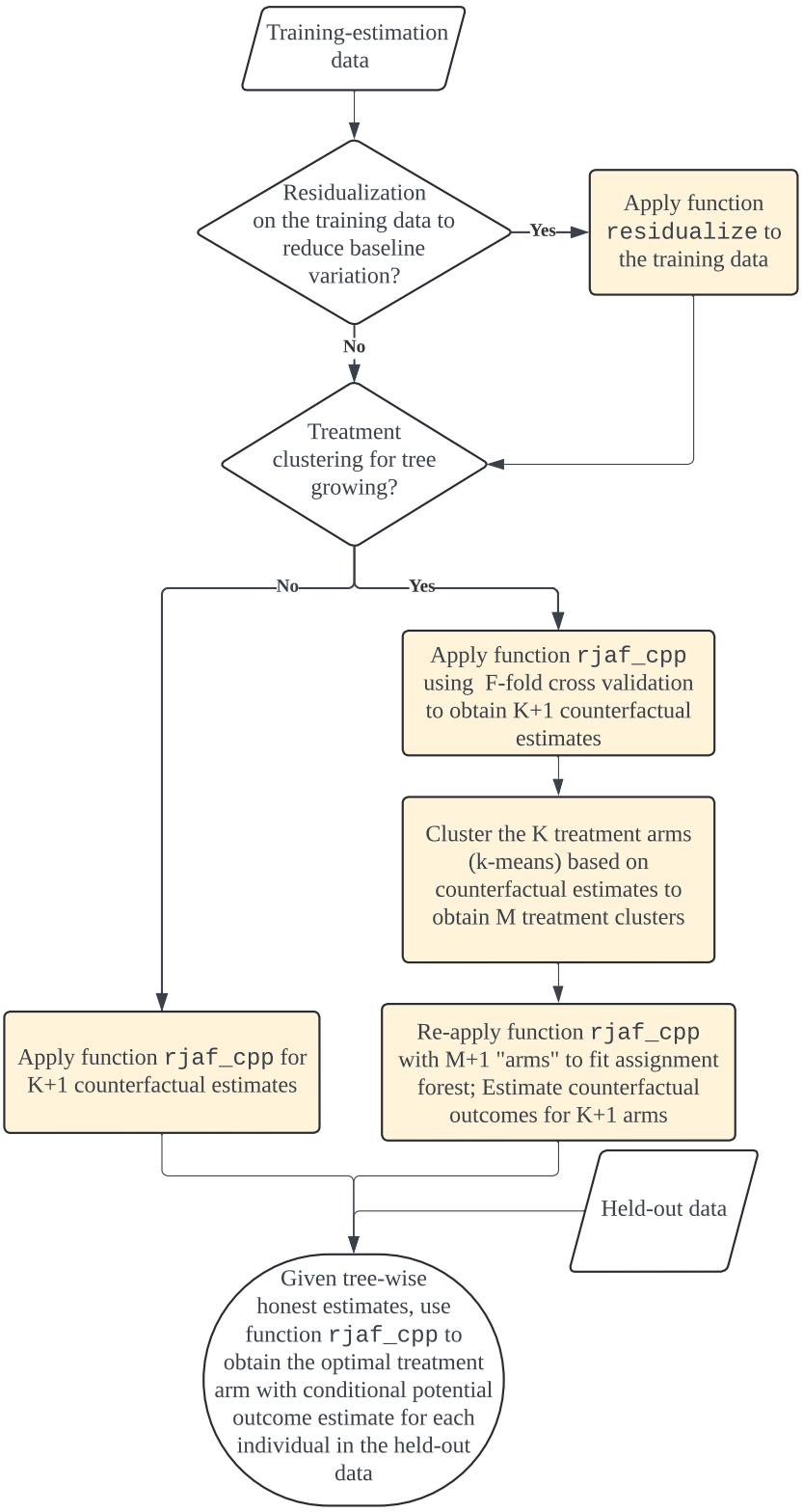

Figure 1: A sketch of the `rjaf` package.

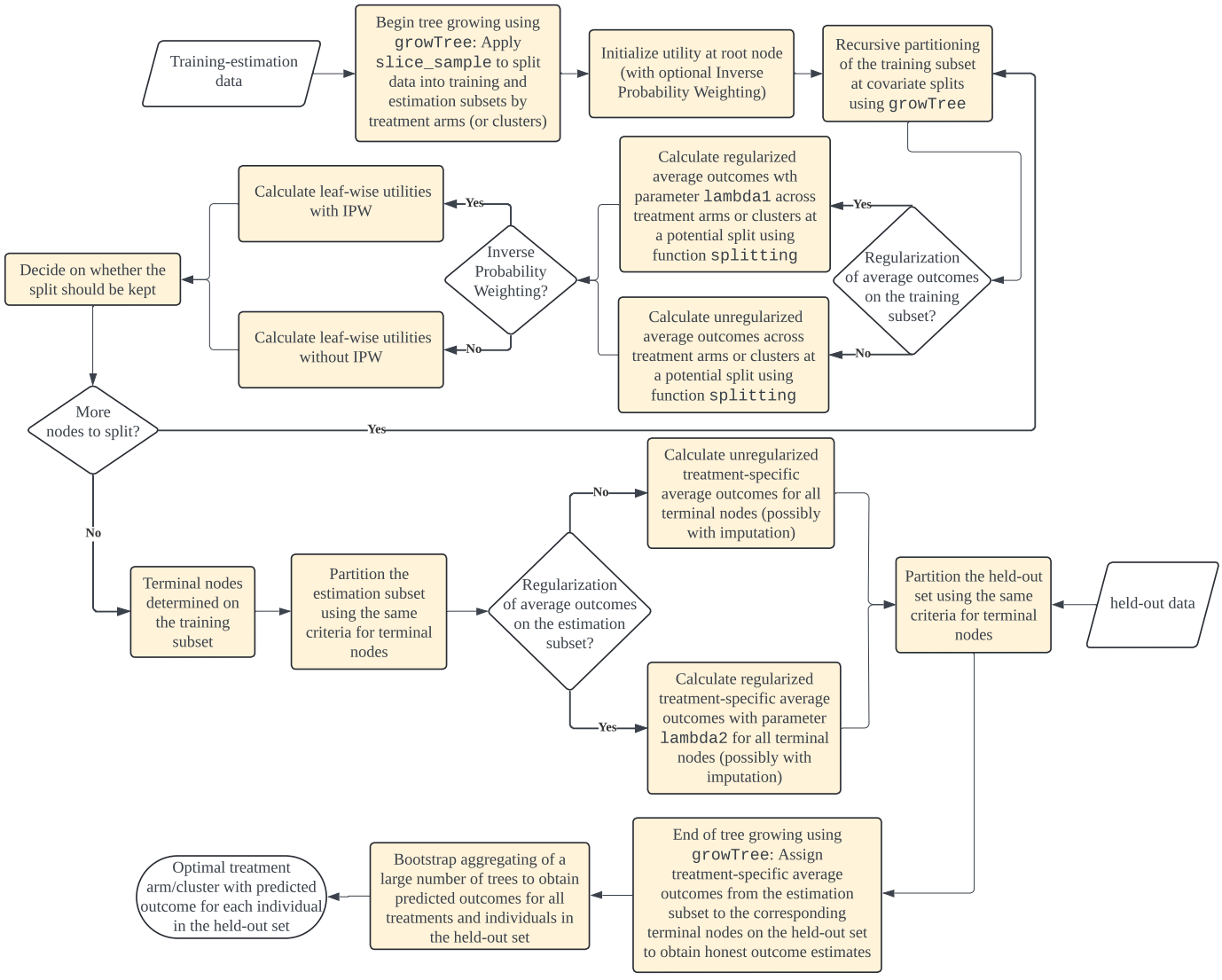

Figure 2: A description of the `rjaf_cpp` function.

```
install.packages('rjaf')
library(rjaf)
```

Example

Next we present an example that illustrates the use of the package. A function `sim.data` used for simulating a synthetic data set with three covariates is provided below, where `n` indicates the sample size, `K` +1 represents the total number of treatment arms (including control), `gamma` denotes the strength of treatment effects, `sigma` is the noise level, and `probability` is a vector of sampling probabilities of treatment arms. This function depends on two widely used `R` packages `MASS` and `dplyr`. The output of the `sim.data` function is a data frame, containing a column (named `id`) of individual IDs, a column (named `Y`) of outcomes, three columns of covariates `X1`, `X2`, and `X3`, a column (named `trt`) of treatment arms, and a column (named `prob`) of probabilities of treatment assignment. Readers are referred to Ladhania et al. (2023, sec. 4.1) for more details about the simulation setup.

```
sim.data <- function(n, K, gamma, sigma, probability = rep(1,K+1)/(K+1)) {
  options(stringsAsFactors=FALSE)
  data <- left_join(data.frame(id=1:n,
                               trt=sample(0:K, n, replace=TRUE, probability),
                               mvrnorm(n, rep(0,3), diag(3))),
                    data.frame(trt=0:K, prob=probability), by="trt")
  data <- mutate(data, tmp1=10+20*(X1>0)-20*(X2>0)-40*(X1>0&X2>0),
                 tmp2=gamma*(2*(X3>0)-1)/(K-1),
                 tmp3=-10*X1^2,
                 Y=tmp1+tmp2*(trt>0)*(2*trt-K-1)+tmp3*(trt==0)+rnorm(n,0,sigma))

  Y.cf <- data.frame(sapply(0:K, function(t)
    mutate(data, Y=tmp1+tmp2*(t>0)*(2*t-K-1)+tmp3*(t==0))$Y))
  names(Y.cf) <- paste0("Y",0:K)
  return(mutate(bind_cols(dplyr::select(data, -c(tmp1,tmp2,tmp3)), Y.cf),
                across(c(id, trt), as.character)))
}
```

Using the `sim.data` function, we generate two data sets: one for training and estimation, and the other is held out for final assignment rules and outcome estimates, both of which have a sample size of 5000. The number of treatment arms is set to 30 (`K` = 29), with a treatment effect strength of `gamma` = 10 and a noise level of `sigma` = 20. Treatment arms are uniformly assigned to all individuals across both data sets. Calling the `rjaf` function returns a tidyverse tibble (Müller and Wickham 2023) containing individual IDs, optimal treatment arms, counterfactual outcomes, predicted outcomes, and treatment arm clusters.

```
library(MASS)
library(dplyr)
K <- 29; gamma <- 10; sigma <- 20; probability <- rep(1,K+1)/(K+1)
n.heldout <- n.trainest <- 5000
data.trainest <- sim.data(n.trainest, K, gamma, sigma, probability)
data.heldout <- sim.data(n.heldout, K, gamma, sigma, probability)
fit <- rjaf(data.trainest, data.heldout, y = "Y", id = "id", trt = "trt",
            vars = paste0("X", 1:3), prob = "prob", ntrt = K+1, nvar = 3,
            lambda1 = 0, lambda2 = 0.5, nodesize = 3, eps = 0.5, reg = TRUE,
            impute = FALSE, clus.tree.growing = TRUE, clus.max = 5)
head(fit)
# A tibble: 6 × 5
  id    trt.rjaf  Y.cf Y.rjaf clus.rjaf
  <chr> <chr>    <dbl>  <dbl>     <int>
```

```
1 1     29        20     11.6        1
2 2     29        40     11.5        1
3 3     3         18.6   10.3        3
4 4     3         38.6   10.4        3
5 5     3         18.6   10.4        3
6 6     3        -21.4   10.3        3
```

To demonstrate the advantage of treatment arm clustering, we conducted a series of simulated data experiments following the above setup. In each experiment, 500 simulated Training-Estimation datasets were generated, while the same held-out data set was used to arrive at the final outcome estimates under personalization for all Training-Estimation sets. The number of treatment arms was set to 10, 30, 50, and 100, respectively. Figure 3 displays boxplots of 500 simulations comparing the average outcome of the held-out set from unclustered and clustered RJAF. "Oracle Optimal Assignment" denotes the assignment strategy derived from the ground truth in simulations, ensuring the best possible outcomes. "Random Assignment" involves randomly distributing units across treatment arms in each simulation. "Global Best Assignment" refers to assigning all units to the treatment in a simulation that demonstrates the highest average performance. Across all settings, the clustered RJAF is associated with a higher average outcome than the unclustered RJAF; the RJAF (both clustered and unclustered) consistently outperforms "Random Assignment" and "Global Best Assignment," while closely approximating the performance of the "Oracle Optimal Assignment."

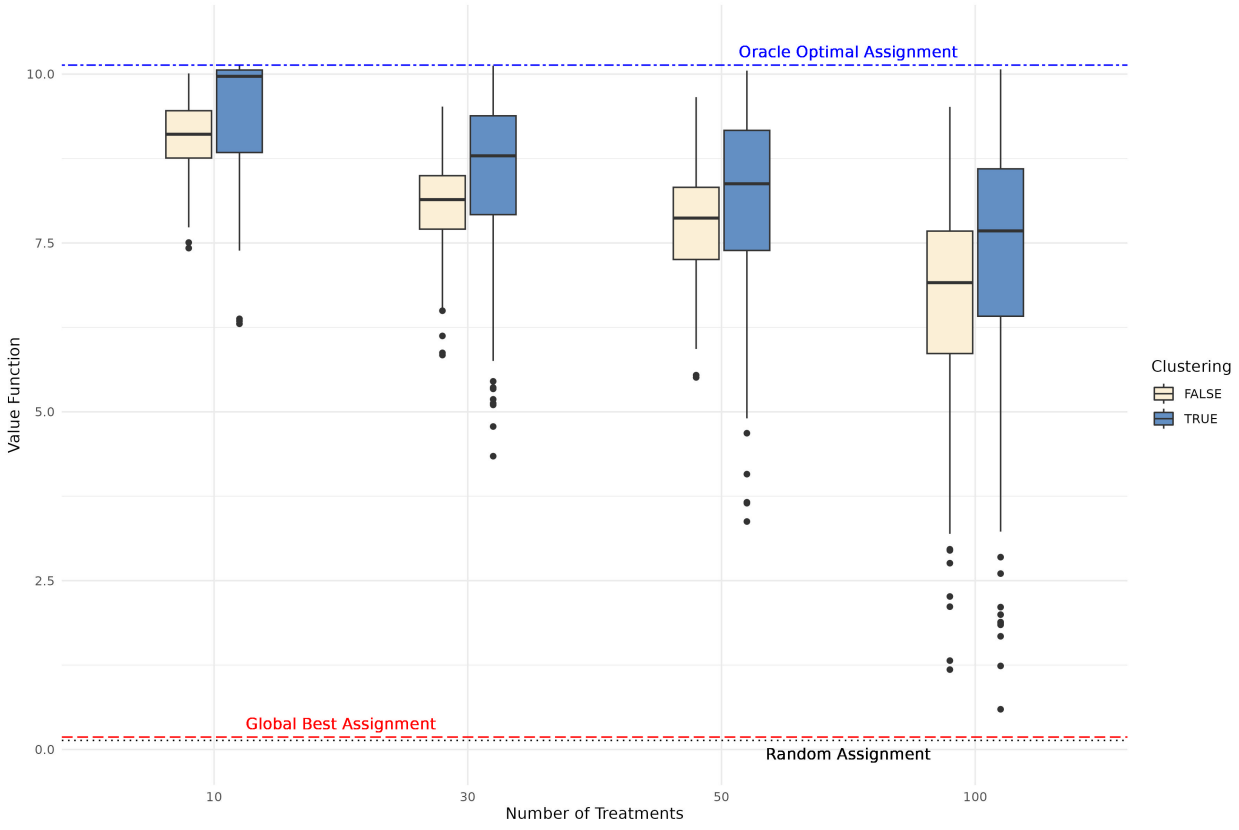

Figure 3: Boxplots of 500 simulations comparing the average outcome of the held-out set from unclustered and clustered RJAF. "Oracle Optimal Assignment" denotes the assignment strategy derived from the ground truth in simulations, ensuring the best possible outcomes. "Random Assignment" involves randomly distributing units across treatment arms in each simulation. "Global Best Assignment" refers to assigning all units to the treatment in a simulation that demonstrates the highest average performance.

Acknowledgments

Wenbo Wu and Xinyi Zhang contributed equally to the project. 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).

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.

Bryan, Christopher J, Elizabeth Tipton, and David S Yeager. 2021. "Behavioural Science Is Unlikely to Change the World Without a Heterogeneity Revolution." *Nature Human Behaviour* 5 (8): 980–89. https://doi.org/10.1038/s41562-021-01143-3.

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, Dena Gromet, Hung Ho, Joseph S Kay, Timothy W Lee, Pepi Pandiloski, Yeji Park, et al. 2021. "Megastudies Improve the Impact of Applied Behavioural Science." *Nature* 600 (7889): 478–83. https://doi.org/10.1038/s41586-021-04128-4.

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.

Müller, Kirill, and Hadley Wickham. 2023. *tibble: Simple Data Frames*. https://tibble.tidyverse.org.

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.