Functions Exante IOp

Pedro Salas-Rojo and Paolo Brunori

February 2024

Documentation for the R functions used to estimate exante IOp using trees and random forests. They are defined to be used for the GEOM database (see also Brunori et al. (2023b), Salas-Rojo and Rodríguez (2022), Brunori et al. (2023a)).[1]

1 Libraries

Libraries are updated on the 1^{st} of February, 2024.

Library	Resource
partykit	https://cran.r-project.org/web/packages/partykit/index.html
glmnet	https://cran.r-project.org/web/packages/glmnet/index.html
grDevices	https://rdocumentation.org/packages/grDevices/versions/3.6.2
stringr	https://cran.r-project.org/web/packages/stringr/index.html
dineq	https://cran.r-project.org/web/packages/dineq/index.html
gtools	https://cran.r-project.org/web/packages/gtools/index.html
caret	https://cran.r-project.org/web/packages/caret/index.html

2 Code

2.1 Tune Ctree

tune_tree(data, model, cv, grid, minbu = 50)

- data: name of dataframe.
- model: formula, e.g., dependent   independents.
- cv: cross-validation information. See "trainControl" in caret package.

[1]Note: Some functions call a weights argument. If you do not want to use weights, use a vector of value 1. Debiased estimators can be obtained following Escanciano and Terschuur (2022). Please, note that these functions merely provide guidance for initial practitioners, and it is the researcher's responsibility to appropriately tune and produce the desired estimates.

- grid: grid of mincriterion values to search and tune the algorithm.
- minbu: minbucket of the tree. Default value: 50.

The function returns three objects:

- mincriterion: tuned mincriterion (1-alpha).
- RMSE: best root mean squared error obtained with tuned mincriterion.
- results: data frame containing all results from the tuning.

2.2 Get Ctree

get_tree(data, model, mincri = 0.99, minbu = 50, maxd = Inf)

- data: name of dataframe.
- model: formula, e.g., dependent   independents.
- mincri: mincriterion (1-alpha) of the tree. Ideally, use the mincriterion tuned in "tune_tree". Default value: 0.99.
- minbu: minbucket of the tree. Default value: 50.
- maxd: maximum depth. Default value: Infinite.

The function returns one tree object, of class "constparty" & "party".
Note: The function sets testtype = "Bonferroni"

2.3 Plot Ctree

plot_tree(data, tree, dep, wts, norm = FALSE, font = 6)

- tree: tree object.
- data: name of dataframe.
- dep: name of the dependent variable (with quotation marks, e.g., "income").
- wts: name of weights (with quotation marks, e.g., "weights").
- norm: if TRUE, normalize outcomes to mean = 1. Default value: FALSE.
- font: size of the font in the plot. Default value: 6.

The function returns one plot object.

2.4 Tune CForest

tune_forest(data, model, cv, grid, ntree = 50, mincri = 0, minbu = 10)

- data: name of dataframe.
- model: formula, e.g., dependent   independents.
- cv: cross-validation information. See "trainControl" in caret package.
- grid: grid of mtry values to tune the algorithm.
- ntree: number of trees to grow in each forest. Default value: 50.
- mincri: mincriterion (1-alpha) of each tree in each forest. Default value: 0.
- minbu: minbucket of each tree in each forest. Default value: 10.

The function returns three objects:

- mtry: tuned mtry.
- RMSE: root mean squared error obtained with mtry.
- results: data frame containing results from the tuning.

Note: The function sets testtype = "Bonferroni".

2.5 Get CForest

get_forest(data, model, ntree = 50, mincri = 0, minbu = 10, mtry = "default", fraction = 0.632)

- data: name of dataframe.
- model: formula, e.g., dependent   independents.
- ntree: number of trees in the forest. Default value = 50. mincri: mincriterion (1-alpha) of trees. Default value = 0.
- minbu: minbucket of each tree. Default value = 10. mtry: mtry of the random forest. Ideally, use that obtained in "tune_forest". Default value mtry = "default" = $\sqrt{k}$, where k is the number of regressors.
- fraction: fraction of the complete dataframe used to run each tree. Default: 0.632.

The function returns one random forest object, class "cforest", "constparties" & "parties".
Note: The function sets testtype = "Bonferroni".

References

Brunori, P., Ferreira, F., and Salas-Rojo, P. (2023a). Inherited inequality: a general framework and an application to south africa. *International Inequalities Institute, London School of Economics and Political Science*, WP: 107.

Brunori, P., Hufe, P., and Mahler, D. (2023b). The roots of inequality: Estimating inequality of opportunity from regression trees and forests. *The Scandinavian Journal of Economics*, 125(4):900–932.

Escanciano, J. C. and Terschuur, J. R. (2022). Debiased semiparametric u-statistics: Machine learning inference on inequality of opportunity. *arXiv preprint arXiv:2206.05235*.

Salas-Rojo, P. and Rodríguez, J. G. (2022). Inheritances and wealth inequality: a machine learning approach. *The Journal of Economic Inequality*, 20(1):27–51.