pdp: An R Package for Constructing Partial Dependence Plots

by Author One

Abstract Complex nonparametric models—like neural networks, random forests, and support vector machines—are more common than ever in predictive analytics, especially when dealing with large observational databases that don't adhere to the strict assumptions imposed by traditional statistical techniques (e.g., multiple linear regression which assumes linearity, homoscedasticity, and normality). Unfortunately, it can be challenging to understand the results of such models and explain them to management. Partial dependence plots offer a simple solution. Partial dependence plots are low-dimensional graphical renderings of the prediction function $\widehat{f}(\boldsymbol{x})$ so that the relationship between the outcome and predictors of interest can be more easily understood. These plots are especially useful in explaining the output from black box models. In this paper, we introduce **pdp**, a general R package for constructing partial dependence plots.

Introduction

Note: This is an updated version of the original R Journal article.

Harrison and Rubinfeld (1978) were among the first to analyze the well-known Boston housing data. One of their goals was to find a housing value equation using data on median home values from $n = 506$ census tracts in the suburbs of Boston from the 1970 census; see Harrison and Rubinfeld (1978, Table IV) for a description of each variable. The data violate many classical assumptions like linearity, normality, and constant variance. Nonetheless, Harrison and Rubinfeld—using a combination of transformations, significance testing, and grid searches—were able to find a reasonable fitting model ($R^2 = 0.81$). Part of the payoff for there time and efforts was an interpretable prediction equation which is reproduced in Equation~(1).

$$\begin{aligned}\widehat{\log(MV)} = {} & 9.76 + 0.0063RM^2 + 8.98 \times 10^{-5}AGE - 0.19\log(DIS) + 0.096\log(RAD) \\ & - 4.20 \times 10^{-4}TAX - 0.031PTRATIO + 0.36(B - 0.63)^2 - 0.37\log(LSTAT) \\ & - 0.012CRIM + 8.03 \times 10^{-5}ZN + 2.41 \times 10^{-4}INDUS + 0.088CHAS \\ & - 0.0064NOX^2\end{aligned} \tag{1}$$

Nowadays, many supervised learning algorithms can fit the data automatically in seconds—typically with higher accuracy. (We will revisit the Boston housing data in Section~**??**.) The downfall, however, is some loss of interpretation since these algorithms typically do not produce simple prediction formulas like Equation~(1). These models can still provide insight into the data, but it is not in the form of simple equations. For example, quantifying predictor importance has become an essential task in the analysis of "big data", and many supervised learning algorithms, like tree-based methods, can naturally assign variable importance scores to all of the predictors in the training data.

While determining predictor importance is a crucial task in any supervised learning problem, ranking variables is only part of the story and once a subset of "important" features is identified it is often necessary to assess the relationship between them (or subset thereof) and the response. This can be done in many ways, but in machine learning it is often accomplished by constructing *partial dependence plots* (PDPs); see Friedman (2001) for details. PDPs help visualize the relationship between a subset of the features (typically 1-3) and the response while accounting for the average effect of the other predictors in the model. They are particularly effective with black box models like random forests and support vector machines.

Let $\boldsymbol{x} = \{x_1, x_2, \ldots, x_p\}$ represent the predictors in a model whose prediction function is $\widehat{f}(\boldsymbol{x})$. If we partition $\boldsymbol{x}$ into an interest set, $\boldsymbol{z}_s$, and its compliment, $\boldsymbol{z}_c = \boldsymbol{x} \setminus \boldsymbol{z}_s$, then the "partial dependence" of the response on $\boldsymbol{z}_s$ is defined as

$$f_s(\boldsymbol{z}_s) = E_{\boldsymbol{z}_c}\left[\widehat{f}(\boldsymbol{z}_s, \boldsymbol{z}_c)\right] = \int \widehat{f}(\boldsymbol{z}_s, \boldsymbol{z}_c)\, p_c(\boldsymbol{z}_c)\, d\boldsymbol{z}_c, \tag{2}$$

where $p_c(\boldsymbol{z}_c)$ is the marginal probability density of $\boldsymbol{z}_c$: $p_c(\boldsymbol{z}_c) = \int p(\boldsymbol{x})\, d\boldsymbol{z}_s$. Equation~(2) can be estimated from a set of training data by

$$\bar{f}_s(\boldsymbol{z}_s) = \frac{1}{n}\sum_{i=1}^{n} \widehat{f}(\boldsymbol{z}_s, \boldsymbol{z}_{i,c}), \tag{3}$$

where $z_{i,c}$ $(i = 1, 2, \ldots, n)$ are the values of z_c that occur in the training sample; that is, we average out the effects of all the other predictors in the model.

Constructing a PDP (3) in practice is rather straightforward. To simplify, let $z_s = x_1$ be the predictor variable of interest with unique values $\{x_{11}, x_{12}, \ldots, x_{1k}\}$. The partial dependence of the response on x_1 can be constructed as follows:

Algorithm 1 A simple algorithm for constructing the partial dependence of the response on a single predictor x_1.

1. For $i \in \{1, 2, \ldots, k\}$:
   (a) Copy the training data and replace the original values of x_1 with the constant x_{1i}.
   (b) Compute the vector of predicted values from the modified copy of the training data.
   (c) Compute the average prediction to obtain $\bar{f}_1(x_{1i})$.
2. Plot the pairs $\{x_{1i}, \bar{f}_1(x_{1i})\}$ for $i = 1, 2, \ldots, k$.

Algorithm~1 can be quite computationally intensive since it involves k passes over the training records. Fortunately, the algorithm can be parallelized quite easily (more on this in Section~**??**). It can also be easily extended to larger subsets of two or more features as well.

Limited implementations of Friedman's PDPs are available in packages **randomForest** (Liaw and Wiener, 2002) and **gbm** (Ridgeway, 2015), among others; these are limited in the sense that they only apply to the models fit using the respective package. For example, the `partialPlot` function in **randomForest** only applies to objects of class `"randomForest"` and the `plot` function in **gbm** only applies to `"gbm"` objects. While the **randomForest** implementation will only allow for a single predictor, the **gbm** implementation can deal with any subset of the predictor space. Partial dependence functions are not restricted to tree-based models; they can be applied to any supervised learning algorithm (e.g., generalized additive models and neural networks). However, to our knowledge, there is no general package for constructing PDPs in R. For example, PDPs for a conditional random forest as implemented by the `cforest` function in the **party** and **partykit** packages; see Hothorn et al. (2015) and Hothorn and Zeileis (2016), respectively. The **pdp** (Greenwell, 2016) package tries to close this gap by offering a general framework for constructing PDPs that can be applied to several classes of fitted models.

The **plotmo** package (Milborrow, 2015) is one alternative to **pdp**. According to Milborrow, **plotmo** constructs "a poor man's partial dependence plot." In particular, it plots a model's response when varying one or two predictors while holding the other predictors in the model constant (continuous features are fixed at their median value, while factors are held at their first level). These plots allow for up to two variables at a time. They are also less accurate than PDPs, but are faster to construct. For additive models (i.e., models with no interactions), these plots are identical in shape to PDPs. As of **plotmo** version 3.3.0, there is now support for constructing PDPs, but it is not the default. The main difference is that **plotmo**, rather than applying step 1. (a)-(c) in Algorithm~1, accumulates all the data at once thereby reducing the number of internal calls to `predict`. The trade-off is a slight increase in speed at the expense of using more memory. So, why use the **pdp** package? As will be discussed in the upcoming sections, **pdp**:

- contains only a few functions with relatively few arguments;
- does not produce a plot by default;
- can be used more efficiently with `"gbm"` objects (see Section **??**);
- produces graphics based on **lattice** (Sarkar, 2008), which are more flexible than base R graphics;
- defaults to using false color level plots for multivariate displays (see Section **??**);
- contains options to mitigate the risks associated with extrapolation (see Section **??**);
- has the option to display progress bars (see Section **??**);
- has the option to construct PDPs in parallel (see Section **??**);
- is extremely flexible in the types of PDPs that can be produced (see Section **??**),

Several additional packages have been developed in recent years that provide similar functionality to **pdp**, such as the **iml** package (Molnar et al., 2018).

PDPs can be misleading in the presence of substantial interactions (Goldstein et al., 2015). To overcome this issue Goldstein, Kapelner, Bleich, and Pitkin developed the concept of *individual*

conditional expectation (ICE) plots—available in the **ICEbox** package. ICE plots display the estimated relationship between the response and a predictor of interest for each observation. Consequently, the PDP for a predictor of interest can be obtained by averaging the corresponding ICE curves across all observations. In Section~**??**, it is shown how to obtain ICE curves using the **pdp** package. It is also possible to display the PDP for a single predictor with **ICEbox**; see `?ICEbox::plot.ice` for an example. **ICEbox** only allows for one variable at a time (i.e., no multivariate displays), though color can be used effectively to display information about an additional predictor. The ability to construct centered ICE (c-ICE) plots and derivative ICE (d-ICE) plots is also available in **ICEbox**; c-ICE plots help visualize heterogeneity in the modeled relationship between observations, and d-ICE plots help to explore interaction effects.

Many other techniques exist for visualizing relationships between the predictors and the response based on a fitted model. For example, the **car** package (Fox and Weisberg, 2011) contains many functions for constructing *partial-residual* and *marginal-model* plots. *Effect displays*, available in the **effects** package (Fox, 2003), provide tabular and graphical displays for the terms in parametric models while holding all other predictors at some constant value—similar in spirit to **plotmo**'s marginal model plots. However, these methods were designed for simpler parametric models (e.g., linear and generalized linear models), whereas **plotmo**, **ICEbox**, and **pdp** are more useful for black box models (although, they can be used for simple parametric models as well).

Constructing PDPs in R

The **pdp** package is useful for constructing PDPs for many classes of fitted models in R. PDPs are especially useful for visualizing the relationships discovered by complex machine learning algorithms such as a random forest. The latest stable release is available from CRAN. The development version is located on GitHub: `https://github.com/bgreenwell/pdp`. Bug reports and suggestions are appreciated and should be submitted to `https://github.com/bgreenwell/pdp/issues`. The two most important functions exported by **pdp** are:

- `partial`
- `plotPartial`

The `partial` function evaluates the partial dependence (3) from a fitted model over a grid of predictor values; the fitted model and predictors are specified using the `object` and `pred.var` arguments, respectively—these are the only required arguments. If `plot = FALSE` (the default), `partial` returns an object of class `"partial"` which inherits from the class `"data.frame"`; put another way, by default, `partial` returns a data frame with an additional class that is recognized by the `plotPartial` function. The columns of the data frame are labeled in the same order as the features supplied to `pred.var`, and the last column is labeled `yhat`[1] and contains the values of the partial dependence function $\bar{f}_s(z_s)$. If `plot = TRUE`, then `partial` makes an internal call to `plotPartial` (with fewer plotting options) and returns the PDP in the form of a **lattice** plot (i.e., a `"trellis"` object). **Note:** it is recommended to call `partial` with `plot = FALSE` and store the results; this allows for more flexible plotting, and the user will not have to waste time calling `partial` again if the default plot is not sufficient.

The `plotPartial` function can be used for displaying more advanced PDPs; it operates on objects of class `"partial"` and has many useful plotting options. For example, `plotPartial` makes it straight forward to add a LOESS smooth, or produce a 3-D surface instead of a false color level plot (the default). Of course, since the default output produced by `partial` is still a data frame, the user can easily use any plotting package he/she desires to visualize the results—**ggplot2** (Wickham, 2009), for instance (see Section~**??** and Section~**??** for examples).

Note: as mentioned above, **pdp** relies on **lattice** for its graphics. **lattice** itself is built on top of **grid** (R Core Team, 2016). **grid** graphics behave a little differently than traditional R graphics, and two points are worth making (see `?lattice` for more details):

1. **lattice** functions return a `"trellis"` object, but do not display it; the `print` method produces the actual display. However, due to R's automatic printing rule, the result is automatically printed when using these functions in the command line. If `plotPartial` is called inside of `source` or inside a loop (e.g., `for` or `while`), an explicit `print` statement is required to display the resulting graph; hence, the same is true when using `partial` with `plot = TRUE`.
2. Setting graphical parameters via `par` typically has no effect on **lattice** plots. Instead, **lattice** provides its own `trellis.par.set` function for modifying graphical parameters.

[1] There is one exception to this. When a function supplied via the `pred.fun` argument returns multiple predictions, the second to last and last columns will be labeled `yhat` and `yhat.id`, respectively (see Section **??**).

A consequence of the second point is that the `par` function cannot be used to control the layout of multiple **lattice** (and hence **pdp**) plots. Simple solutions are available in packages **latticeExtra** (Sarkar and Andrews, 2016) and **gridExtra** (Auguie, 2016). For convenience, **pdp** imports the `grid.arrange` function from **gridExtra** which makes it easy to display multiple **grid**-based graphical objects on a single plot (these include graphics produced using **lattice** (hence, **pdp**) and **ggplot2**). This is demonstrated in multiple examples throughout this paper.

Currently supported models are described in Table~1. In these cases, the user does not need to supply a prediction function (more on this in Section~**??**) or a value for the `type` argument (i.e., `"regression"` or `"classification"`). In other situations, the user may need to specify one or both of these arguments. This allows `partial` to be flexible enough to handle many of the model types not listed in Table~1; for example, neural networks from the **nnet** package (Venables and Ripley, 2002).% and projection pursuit regression (Friedman and Stuetzle, 1981) using the `ppr` function in the **stats** package.

Type of model	R package	Object class
Decision tree	**C50** (Kuhn et al., 2015)	`"C5.0"`
	party	`"BinaryTree"`
	partykit	`"party"`
	rpart (Therneau et al., 2015)	`"rpart"`
Bagged decision trees	**adabag** (Alfaro et al., 2013)	`"bagging"`
	ipred (Peters and Hothorn, 2015)	`"classbagg"`, `"regbagg"`
Boosted decision trees	**adabag** (Alfaro et al., 2013)	`"boosting"`
	gbm	`"gbm"`
	xgboost	`"xgb.Booster"`
Cubist	**Cubist** (Kuhn et al., 2014)	`"cubist"`
Discriminant analysis	**MASS** (Venables and Ripley, 2002)	`"lda"`, `"qda"`
Generalized linear model	**stats**	`"glm"`, `"lm"`
Linear model	**stats**	`"lm"`
Nonlinear least squares	**stats**	`"nls"`
Multivariate adaptive regression splines (MARS)	**earth** (Milborrow, 2016)	`"earth"`
	mda (Leisch et al., 2016)	`"mars"`
Projection pursuit regression	**stats**	`"ppr"`
Random forest	**randomForest**	`"randomForest"`
	party	`"RandomForest"`
	partykit	`"cforest"`
	ranger (Wright, 2016)	`"ranger"`
Support vector machine	**e1071** (Meyer et al., 2015)	`"svm"`
	kernlab (Karatzoglou et al., 2004)	`"ksvm"`

Table 1: Models specifically supported by the **pdp** package. **Note:** for some of these cases, the user may still need to supply additional arguments in the call to `partial`.

The `partial` function also supports objects of class `"train"` produced using the `train` function from the well-known **caret** package (Kuhn, 2016). This means that `partial` can be used with any classification or regression model that has been fit using **caret**'s `train` function; see `http://topepo.github.io/caret/available-models.html` for a current list of models supported by **caret**. An example is given in Section~**??**.

Another important argument to `partial` is `train`. If `train = NULL` (the default), `partial` tries to extract the original training data from the fitted model object. For objects that typically store a copy of the training data (e.g., objects of class `"BinaryTree"`, `"RandomForest"`, and `"train"`), this is straightforward. Otherwise, `partial` will attempt to extract the call stored in `object` (if available) and use that to evaluate the training data in the same environment from which `partial` was called. This can cause problems when, for example, the training data have been changed after fitting the model, but before calling `partial`. Hence, it is good practice to always supply the training data via the `train` argument in the call to `partial`[2]. If `train = NULL` and the training data can not be extracted from

[2]For brevity, we ignore this option in most of the examples in this paper.

the fitted model, the user will be prompted with an informative error message (this will occur, for example, when using `partial` with `"ksvm"` and `"xgb.Booster"` objects):

```
Error: The training data could not be extracted from object. Please supply
the raw training data using the `train` argument in the call to `partial`.
```

For illustration, we will use a corrected version of the Boston housing data analyzed in Harrison and Rubinfeld (1978); the data are available in the **pdp** package (see `?pdp::boston` for details). We begin by loading the data and fitting a random forest with default tuning parameters and 500 trees:

```
library(randomForest)  # for randomForest, partialPlot, and varImpPlot functions

data(boston, package = "pdp")  # load the (corrected) Boston housing data

# Fit a default random forest to the Boston housing data
set.seed(101)  # for reproducibility
boston.rf <- randomForest(cmedv ~ ., data = boston, importance = TRUE)
varImpPlot(boston.rf)  # Figure 1
```

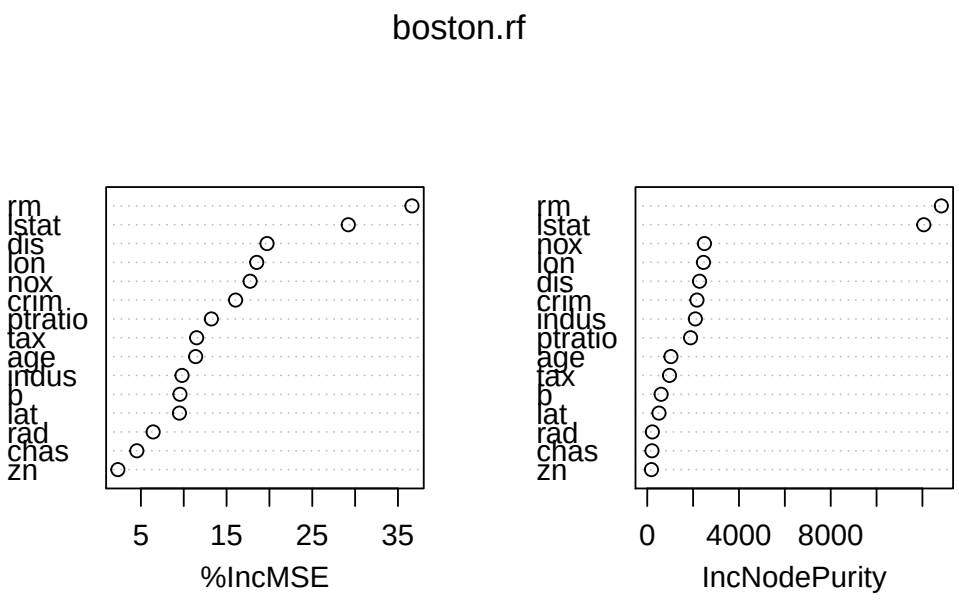

Figure 1: Dotchart of variable importance scores for the Boston housing data based on a random forest with 500 trees.

The model fit is reasonable, with an *out-of-bag* (pseudo) R^2 of 0.89. The variable importance scores are displayed in Figure~1. Both plots indicate that the percentage of lower status of the population (`lstat`) and the average number of rooms per dwelling (`rm`) are highly associated with the median value of owner-occupied homes (`cmedv`). The question then arises, "What is the nature of these associations?" To help answer this, we can look at the partial dependence of `cmedv` on `lstat` and `rm`, both individually and together.

Bibliography

E. Alfaro, M. Gámez, and N. García. adabag: An R package for classification with boosting and bagging. *Journal of Statistical Software*, 54(2):1–35, 2013. URL https://doi.org/10.18637/jss.v054.i02. [p4]

B. Auguie. *gridExtra: Miscellaneous Functions for "Grid" Graphics*, 2016. URL https://CRAN.R-project.org/package=gridExtra. R package version 2.2.1. [p4]

J. Fox. Effect displays in R for generalised linear models. *Journal of Statistical Software*, 8(15):1–27, 2003. URL https://doi.org/10.18637/jss.v008.i15. [p3]

J. Fox and S. Weisberg. *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, 2nd edition, 2011. URL http://socserv.socsci.mcmaster.ca/jfox/Books/Companion. [p3]

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29:1189–1232, 2001. URL https://doi.org/10.1214/aos/1013203451. [p1]

J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823, 1981. URL https://doi.org/10.1080/01621459.1981.10477729. [p4]

A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015. URL https://doi.org/10.1080/10618600.2014.907095. [p2]

B. Greenwell. *Pdp: An R Package for Constructing Partial Dependence Functions*, 2016. URL https://CRAN.R-project.org/package=partial. R package version 0.0.1. [p2]

D. Harrison and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978. URL https://doi.org/10.1016/0095-0696(78)90006-2. [p1, 5]

T. Hothorn and A. Zeileis. *Partykit: A Laboratory for Recursive Partytioning*, 2016. URL https://CRAN.R-project.org/package=partykit. R package version 1.0-5. [p2]

T. Hothorn, K. Hornik, C. Strobl, and A. Zeileis. *Party: A Laboratory for Recursive Partytioning*, 2015. URL https://CRAN.R-project.org/package=party. R package version 1.0-25. [p2]

A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. Kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. URL https://doi.org/10.18637/jss.v011.i09. [p4]

M. Kuhn. *Caret: Classification and Regression Training*, 2016. URL https://CRAN.R-project.org/package=caret. R package version 6.0-73. [p4]

M. Kuhn, S. Weston, C. Keefer, and N. C. C. code for Cubist by Ross Quinlan. *Cubist: Rule- And Instance-Based Regression Modeling*, 2014. URL https://CRAN.R-project.org/package=Cubist. R package version 0.0.18. [p4]

M. Kuhn, S. Weston, N. Coulter, and M. C. C. code for C5. 0 by R. Quinlan. *C50: C5.0 Decision Trees and Rule-Based Models*, 2015. URL https://CRAN.R-project.org/package=C50. R package version 0.1.0-24. [p4]

F. Leisch, K. Hornik, and B. D. Ripley. *Mda: Mixture and Flexible Discriminant Analysis*, 2016. URL https://CRAN.R-project.org/package=mda. R package version 0.4-9. [p4]

A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002. URL http://CRAN.R-project.org/doc/Rnews/. [p2]

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. URL https://CRAN.R-project.org/package=e1071. R package version 1.6-7. [p4]

S. Milborrow. *Plotmo: Plot a Model's Response and Residuals*, 2015. URL https://CRAN.R-project.org/package=plotmo. R package version 3.1.4. [p2]

S. Milborrow. *Earth: Multivariate Adaptive Regression Splines*, 2016. URL https://CRAN.R-project.org/package=earth. R package version 4.4.4. [p4]

C. Molnar, B. Bischl, and G. Casalicchio. iml: An r package for interpretable machine learning. *JOSS*, 3(26):786, 2018. doi: 10.21105/joss.00786. URL https://joss.theoj.org/papers/10.21105/joss.00786. [p2]

A. Peters and T. Hothorn. *Ipred: Improved Predictors*, 2015. URL https://CRAN.R-project.org/package=ipred. R package version 0.9-5. [p4]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016. URL https://www.R-project.org/. [p3]

G. Ridgeway. *Gbm: Generalized Boosted Regression Models*, 2015. URL https://CRAN.R-project.org/package=gbm. R package version 2.1.1. [p2]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer-Verlag, New York, 2008. URL http://lmdvr.r-forge.r-project.org. ISBN 978-0-387-75968-5. [p2]

D. Sarkar and F. Andrews. *latticeExtra: Extra Graphical Utilities Based on Lattice*, 2016. URL https://CRAN.R-project.org/package=latticeExtra. R package version 0.6-28. [p4]

T. Therneau, B. Atkinson, and B. Ripley. *Rpart: Recursive Partitioning and Regression Trees*, 2015. URL https://CRAN.R-project.org/package=rpart. R package version 4.1-10. [p4]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. [p4]

H. Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org. [p3]

M. N. Wright. *Ranger: A Fast Implementation of Random Forests*, 2016. URL https://CRAN.R-project.org/package=ranger. R package version 0.6.0. [p4]

Author One
Affiliation
line 1
line 2
https://journal.r-project.org
ORCiD: 0000-0002-9079-593X
author1@work