

# FFTrees: An R package to create and visualise Fast and Frugal decision Trees

Nathaniel Phillips<sup>1</sup>, Hansjörg Neth<sup>2</sup>, Wolfgang Gaissmaier<sup>2</sup> <sup>1</sup>University of Basel, <sup>2</sup>University of Konstanz

## Introduction and Goals

Complex real-world decisions under uncertainty call for rapid and robust classification strategies. *Fast and Frugal Trees* (FFTs) are a quintessential family of simple heuristics designed to make transparent, robust binary classification decisions under uncertainty [2, 3, 4]. FFTs are remarkably **effective** and **efficient**, and **easy to implement** compared to more complex algorithms such as logistic regression (LR) and CART. FFTs have been successfully implemented in a variety of applied domains, including medical, legal, and financial decision-making [1, 2, 4, 5].

The purpose of this research was to create an R package called **FFTrees** that allows anyone to create their own FFTs with the following criteria:

- Easy to use:** Build effective trees with as little as 2 lines of code.
- Customisable:** Easily customise trees to match user criteria such as the maximum number of cues or the desired balance of hits (HR) vs. false-alarms (FAR).
- Transparent:** Visualise trees and their performance using simple displays that *anyone* can understand and easily implement.
- Open Source:** Source code is freely available and well documented on GitHub with tutorials and example datasets.

## Installing and using FFTrees

You can install `FFTrees` (latest version v1.1.8) from CRAN. To open the main package vignette, run `FFTrees.guide()`

```
install.packages("FFTrees") # Install stable version (Currently v1.1.8) from CRAN
library("FFTrees")          # Load the package
FFTrees.guide()             # Open the FFTrees package guide
```

### Example: Predicting heart disease

Let's create FFTs to predict heart disease from a real world dataset (<https://archive.ics.uci.edu/ml/datasets/Heart+Disease>) called `heartdisease`. The data contains measures from 303 patients suspected of having heart disease. Each patient is characterized by a mixture of 13 numeric and factor predictors and one binary criterion variable `diagnosis` indicating whether the patient truly has heart disease or not. Here are the first few rows of the `heartdisease` data:

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	diagnosis
63	1	ta	145	233	1	hypertrophy	150	0	2.3	down	0	fd	0
67	1	a	160	286	0	hypertrophy	108	1	1.5	flat	3	normal	1
67	1	a	120	229	0	hypertrophy	129	1	2.6	flat	2	rd	1
37	1	np	130	250	0	normal	187	0	3.5	down	0	normal	0

Before we create the trees, we'll randomly split the original `heartdisease` dataset into a 50% training set `heart.train` and a 50% test set `heart.test`. The trees will be trained on `heart.train` and tested on `heart.test`:

```
# Create 50/50 split of training and test data
heartdisease <- heartdisease[sample(1:303, size = 303), ] # Randomly sort heartdisease
heart.train <- heartdisease[1:150,] # Assign rows 1:150 to heart.train
heart.test <- heartdisease[151:303,] # Assign rows 151:303 to heart.test
```

data

FFTrees

data.test

Training / Prediction separation is built into FFTrees

To create the trees, we'll use the main function `FFTrees()` with three arguments: `formula`, a formula specifying a binary criterion as a function of a set of potential predictors to be considered in the trees (the `formula = criterion ~ .` argument will *consider* all predictors for use in the trees), `data`, a dataframe of *training* data, and `data.test`, an optional dataframe of *test* data. Here, we'll create a `FFTrees` object called `heart.ffttrees` with training data `data = heart.train` and test data `data.test = heart.test`:

```
# Create an FFTrees object called heart.ffttrees
heart.ffttrees <- FFTrees(formula = diagnosis ~ ., # Predict diagnosis based on any/all cues
                          data = heart.train,      # Training data
                          data.test = heart.test)  # Testing data (prediction) data
```

The trees, including training and test statistics are now stored in the `heart.ffttrees` object. Every `FFTrees` object has multiple trees which differentially balance hit-rates and false-alarm rates [4]. Evaluating the `heart.ffttrees` object prints summary statistics about the trees, as well as statistics for the tree with the 'best' (highest value of  $v = HR - FAR$ ) training performance:

```
# Print the FFTrees object
heart.ffttrees

## [1] "An FFTrees object containing 7 trees using 4 predictors {thal,cp,ca,oldpeak}"
## [1] "FFTrees AUC: (Train = 0.88, Test = 0.89)"
## [1] "My favorite training tree is #4, here is how it performed:"
##          train  test
## n          150.00 153.00
## p(Correct)    0.80  0.82
## Hit Rate (HR)  0.82  0.88
## False Alarm Rate (FAR) 0.21 0.24
## d-prime       1.70  1.87
```

Here, we see that `heart.ffttrees` contains 7 different trees using up to 4 cues (`thal`, `cp`, `ca`, `oldpeak`). The best tree in the training data, with the highest value of  $v = HR - FAR$ , was tree #4. As you can see, tree #4 actually made *more* correct decisions and had a higher d-prime in the unseen `heart.test` test data than the `heart.train` training data!

## Plotting FFTrees

You can visualize trees by applying the generic `plot()` function to an `FFTrees` object. You can specify which tree to plot in the `tree` argument. To show the tree with the best training performance (in terms of  $v = HR - FAR$ ), include the `tree = "best.train"` argument. In the code below, I'll plot the best training tree (which we know is tree #4) in the `heart.ffttrees` object when applied to the `heart.test` test data:

```
# Plot the heartdisease FFTrees object
plot(x = heart.ffttrees, # heart.ffttrees object
     tree = "best.train", # Show best training tree
     data = "test",       # Show test (prediction) stats statistics
     main = "Heart Disease FFTs", # Data title
     decision.names = c("Healthy", "Disease")) # Classification names
```

In addition to showing performance statistics for the selected tree, the plot also shows, in the lower-right corner, an ROC curve of the performance of *all* trees in the `FFTrees` object, as well as logistic regression and CART.

To display a different tree, just specify the tree by number in the `tree` argument. We can see in the ROC curve above that tree #6 has a higher hit-rate than tree #4 but also a higher false-alarm rate. To display this tree, we'll just specify `tree = 6`:

```
# Plot tree #6 with test performance
plot(x = heart.ffttrees, # heart.ffttrees object
     tree = 6,           # Show tree #6
     data = "test",       # Show test (prediction) stats
     main = "Heart Disease FFTs", # Data title
     decision.names = c("Healthy", "Disease")) # Classification names
```

### Additional arguments to FFTrees

```
# Additional arguments to FFTrees()
FFTrees(formula = diagnosis ~ .,
        data = heartdisease,
        max.levels = 2, # Include a maximum of 2 levels in each tree
        train.p = .5,   # Automatically split data into 50/50 training and test data
        rank.method = "c", # Rank cues by conditional rather than marginal validity
        hr.weight = .8)   # Give more weight to maximizing HR than minimizing FAR
```

## Additional arguments to FFTrees

```
# Additional arguments to FFTrees()
FFTrees(formula = diagnosis ~ .,
        data = heartdisease,
        max.levels = 2, # Include a maximum of 2 levels in each tree
        train.p = .5,   # Automatically split data into 50/50 training and test data
        rank.method = "c", # Rank cues by conditional rather than marginal validity
        hr.weight = .8)   # Give more weight to maximizing HR than minimizing FAR
```

## Contact and FFTrees Source Code

[nathaniel.phillips@unibas.ch](mailto:nathaniel.phillips@unibas.ch)  
[www.github.com/ndphillips/FFTrees](https://www.github.com/ndphillips/FFTrees)



## References

[1] Aikman, D., Galesic, M., Gigerenzer, G., Kapadia, S., Katsikopoulos, K. V., Kothiyal, A., ... & Neumann, T. (2014). Taking uncertainty seriously: simplicity versus complexity in financial regulation. Bank of England Financial stability paper, (28).

[2] Gigerenzer, G., & Todd, P. M. (1999). Simple heuristics that make us smart. Oxford University Press, USA.

[3] Luan, S., Schooler, L. J., & Gigerenzer, G. (2011). A signal-detection analysis of fast-and-frugal trees. Psychological Review, 118(2), 316.

[4] Martignon, L., Katsikopoulos, K. V., & Woike, J. K. (2008). Categorization with limited resources: A family of simple heuristics. Journal of Mathematical Psychology, 52(6), 352-361.

[5] Neth, H., Meder, B., Kothiyal, A., & Gigerenzer, G. (2014). Homo heuristics in the financial world: From risk management to managing uncertainty. Journal of Risk Management in Financial Institutions, 7(2), 134-144.

QR code for this poster →

