http://www.r-bloggers.com/a-brief-tour-of-the-trees-and-forests/

A Brief Tour of the Trees and Forests

*April 29, 2013*

By [Wesley](http://www.r-bloggers.com/author/wesley/)

[inShare](javascript:void(0);)1

(This article was first published on [**Statistical Research » R**](http://statistical-research.com/a-brief-tour-of-the-trees-and-forests/?utm_source=rss&utm_medium=rss&utm_campaign=a-brief-tour-of-the-trees-and-forests), and kindly contributed to [R-bloggers)](http://www.r-bloggers.com/)

Tree methods such as CART (classification and regression trees) can be used as alternatives to logistic regression. It is a way that can be used to show the probability of being in any hierarchical group. The following is a compilation of many of the key R packages that cover trees and forests.  The goal here is to simply give some brief examples on a few approaches on growing trees and, in particular, the visualization of the trees. These packages include classification and regression trees, graphing and visualization, ensemble learning using random forests, as well as evolutionary learning trees. There are a wide array of package in R that handle decision trees including trees for longitudinal studies.  I have found that when using several combinations of these packages simultaneously that some of the function begin to fail to work.

The concept of trees and forests can be applied in many different setting and is often seen in machine learning and data mining settings or other settings where there is a significant amount of data.  The examples below are by no means comprehensive and exhaustive. However, there are several examples given using different datasets and a variety of R packages. The first example uses some data obtain from the Harvard Dataverse Network. For reference the data can be obtain from <http://dvn.iq.harvard.edu/dvn/>. The study was recently released on April 22nd, 2013 and the raw data as well as the documentation is available on the Dataverse web site and the study ID is hdl:1902.1/21235. The other examples use data that are shipped with the R packages.

**rpart**

This package includes several example sets of data that can be used for recursive partitioning and regression trees.  Categorical or continuous variables can be used depending on whether one wants classification trees or regression trees. This package as well at the *tree* package are probably the two go-to packages for trees.  However, care should be taken as the *tree* package and the *rpart* package can produce very different results.

|  |  |  |
| --- | --- | --- |
|  | library(rpart)  raw.orig <-read.csv(file="c:\\rsei212\_chemical.txt", header=T, sep="\t")    # Keep the dataset small and tidy  # The Dataverse: hdl:1902.1/21235  raw = subset(raw.orig, select=c("Metal","OTW","AirDecay","Koc"))    row.names(raw) = raw.orig$CASNumber  raw = na.omit(raw);    frmla = Metal ~ OTW + AirDecay + Koc    # Metal: Core Metal (CM); Metal (M); Non-Metal (NM); Core Non-Metal (CNM)    fit = rpart(frmla, method="class", data=raw)    printcp(fit) # display the results  **Classification tree:**  **rpart(formula = frmla, data = raw, method = "class")**  **Variables actually used in tree construction:**  **[1] AirDecay Koc OTW**  **Root node error: 172/352 = 0.48864**  **n= 352**  **CP nsplit rel error xerror xstd**  **1 0.279070 0 1.00000 1.12209 0.054285**  **2 0.069767 1 0.72093 0.79070 0.053112**  **3 0.029070 2 0.65116 0.75581 0.052644**  **4 0.017442 3 0.62209 0.76744 0.052808**  **5 0.011628 4 0.60465 0.78488 0.053039**  **6 0.010000 5 0.59302 0.77907 0.052964**  plotcp(fit) # visualize cross-validation results    summary(fit) # detailed summary of splits     |  | | --- | | **Call:**  **rpart(formula = frmla, data = raw, method = "class")**  **n= 352**  **CP nsplit rel error xerror xstd**  **1 0.27906977 0 1.0000000 1.1220930 0.05428470**  **2 0.06976744 1 0.7209302 0.7906977 0.05311248**  **3 0.02906977 2 0.6511628 0.7558140 0.05264394**  **4 0.01744186 3 0.6220930 0.7674419 0.05280786**  **5 0.01162791 4 0.6046512 0.7848837 0.05303920**  **6 0.01000000 5 0.5930233 0.7790698 0.05296401**  **Variable importance**  **Koc AirDecay OTW**  **45 34 21**  **Node number 1: 352 observations, complexity param=0.2790698**  **predicted class=CNM expected loss=0.4886364 P(node) =1**  **class counts: 5 180 3 164**  **probabilities: 0.014 0.511 0.009 0.466**  **left son=2 (177 obs) right son=3 (175 obs)**  **Primary splits:**  **Koc < 190.5 to the left, improve=17.891350, (0 missing)**  **AirDecay < 0.0266 to the left, improve= 9.972175, (0 missing)**  **OTW < 3.1 to the left, improve= 3.928582, (0 missing)**  **Surrogate splits:**  **AirDecay < 0.03495 to the left, agree=0.605, adj=0.206, (0 split)**  **OTW < 87.5 to the right, agree=0.574, adj=0.143, (0 split)**  **Node number 2: 177 observations**  **predicted class=CNM expected loss=0.3333333 P(node) =0.5028409**  **class counts: 5 118 0 54**  **probabilities: 0.028 0.667 0.000 0.305**  **Node number 3: 175 observations, complexity param=0.06976744**  **predicted class=NM expected loss=0.3714286 P(node) =0.4971591**  **class counts: 0 62 3 110**  **probabilities: 0.000 0.354 0.017 0.629**  **left son=6 (34 obs) right son=7 (141 obs)**  **Primary splits:**  **AirDecay < 0.005825 to the left, improve=8.319349, (0 missing)**  **OTW < 12.5 to the left, improve=5.497489, (0 missing)**  **Koc < 2500 to the left, improve=3.099420, (0 missing)**  **Node number 6: 34 observations, complexity param=0.01162791**  **predicted class=CNM expected loss=0.3235294 P(node) =0.09659091**  **class counts: 0 23 0 11**  **probabilities: 0.000 0.676 0.000 0.324**  **left son=12 (20 obs) right son=13 (14 obs)**  **Primary splits:**  **AirDecay < 0.000574 to the right, improve=2.925210, (0 missing)**  **OTW < 480 to the left, improve=2.165686, (0 missing)**  **Koc < 3600 to the left, improve=1.083411, (0 missing)**  **Surrogate splits:**  **Koc < 4900 to the left, agree=0.706, adj=0.286, (0 split)**  **OTW < 3.65 to the right, agree=0.647, adj=0.143, (0 split)**  **Node number 7: 141 observations, complexity param=0.02906977**  **predicted class=NM expected loss=0.2978723 P(node) =0.4005682**  **class counts: 0 39 3 99**  **probabilities: 0.000 0.277 0.021 0.702**  **left son=14 (29 obs) right son=15 (112 obs)**  **Primary splits:**  **OTW < 12.5 to the left, improve=6.551475, (0 missing)**  **Koc < 1250 to the left, improve=2.542887, (0 missing)**  **AirDecay < 0.2175 to the right, improve=1.717422, (0 missing)**  **Node number 12: 20 observations**  **predicted class=CNM expected loss=0.15 P(node) =0.05681818**  **class counts: 0 17 0 3**  **probabilities: 0.000 0.850 0.000 0.150**  **Node number 13: 14 observations**  **predicted class=NM expected loss=0.4285714 P(node) =0.03977273**  **class counts: 0 6 0 8**  **probabilities: 0.000 0.429 0.000 0.571**  **Node number 14: 29 observations, complexity param=0.01744186**  **predicted class=CNM expected loss=0.4137931 P(node) =0.08238636**  **class counts: 0 17 0 12**  **probabilities: 0.000 0.586 0.000 0.414**  **left son=28 (20 obs) right son=29 (9 obs)**  **Primary splits:**  **Koc < 1250 to the left, improve=1.6689660, (0 missing)**  **OTW < 9.4 to the right, improve=1.3546800, (0 missing)**  **AirDecay < 0.04055 to the left, improve=0.3952813, (0 missing)**  **Surrogate splits:**  **AirDecay < 0.5575 to the left, agree=0.759, adj=0.222, (0 split)**  **Node number 15: 112 observations**  **predicted class=NM expected loss=0.2232143 P(node) =0.3181818**  **class counts: 0 22 3 87**  **probabilities: 0.000 0.196 0.027 0.777**  **Node number 28: 20 observations**  **predicted class=CNM expected loss=0.3 P(node) =0.05681818**  **class counts: 0 14 0 6**  **probabilities: 0.000 0.700 0.000 0.300**  **Node number 29: 9 observations**  **predicted class=NM expected loss=0.3333333 P(node) =0.02556818**  **class counts: 0 3 0 6**  **probabilities: 0.000 0.333 0.000 0.667** |   # plot tree  plot(fit, uniform=TRUE, main="Classification Tree for Chemicals")  text(fit, use.n=TRUE, all=TRUE, cex=.8)    # tabulate some of the data  table(subset(raw, Koc>=190.5)$Metal)  CM CNM M NM  0 62 3 110 |

**tree**

This is the primary R package for classification and regression trees.  It has functions to prune the tree as well as general plotting functions and the mis-classifications (total loss). The output from *tree* can be easier to compare to the General Linear Model (GLM) and General Additive Model (GAM) alternatives.

|  |  |
| --- | --- |
|  | ###############  # TREE package  install.packages("tree")  library(tree)    tr = tree(frmla, data=raw)  summary(tr)  **Classification tree:**  **tree(formula = frmla, data = raw)**  **Number of terminal nodes: 12**  **Residual mean deviance: 1.186 = 403.3 / 340**  **Misclassification error rate: 0.2812 = 99 / 352**  plot(tr); text(tr) |

**party**

This is another package for recursive partitioning. One of the key functions in this package is ctree. As the package documention indicates it can be used for continuous, censored, ordered, nominal and multivariate response variable in a conditional inference framework. The party package also implements recursive partitioning for survival data.

|  |  |
| --- | --- |
|  | ###############  # PARTY package  install.packages("party")  library(party)    ct = ctree(frmla, data = raw))  plot(ct, main="Conditional Inference Tree")    #Table of prediction errors  table(predict(ct), raw$Metal)    CM CNM M NM  CM 0 0 0 0  CNM 0 95 1 48  M 0 0 0 0  NM 5 85 2 116  # Estimated class probabilities  tr.pred = predict(ct, newdata=raw, type="prob") |

**maptree**

*maptree* is a very good at graphing, pruning data from hierarchical clustering, and CART models. The trees produced by this package tend to be better labeled and higher quality and the stock plots from *rpart*.

**partykit**

This contains a re-implementation of the *ctree* function and it provides some very good graphing and visualization for tree models.  It is similar to the *party* package.  The example below uses data from *airquality*dataset and the famous *species* data available in R and can be found in the documentation.

|  |  |
| --- | --- |
| 1 | <a href="<http://statistical-research.com/wp-content/uploads/2012/12/species.png>"><img alt="Species Decision Tree" src="<http://statistical-research.com/wp-content/uploads/2012/12/species.png>" width="437" height="472" /></a> <a href="<http://statistical-research.com/wp-content/uploads/2012/12/airqualityOzone.png>"><img alt="Ozone Air Quality Decision Tree" src="<http://statistical-research.com/wp-content/uploads/2012/12/airqualityOzone.png>" width="437" height="472" /></a> |

**evtree**

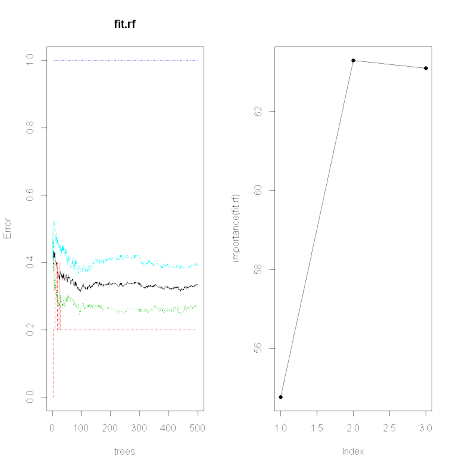
This package uses evolutionary algorithms.  The idea behind this approach is that is will reduce the *a priori* bias.  I have seen trees of this sort in the area of environmental research, bioinformatics, systematics, and marine biology.  Though there are many other areas than that of phylogentics.

|  |  |
| --- | --- |
|  | ###############  ## EVTREE (Evoluationary Learning)  library(evtree)    ev.raw = evtree(frmla, data=raw)  plot(ev.raw)    table(predict(ev.raw), raw$Metal)  CM CNM M NM  CM 0 0 0 0  CNM 5 142 0 49  M 0 0 0 0  NM 0 38 3 115  1-mean(predict(ev.raw) == raw$Metal) |

**randomForest**

Random forests are very good in that it is an ensemble learning method used for classification and regression.  It uses multiple models for better performance that just using a single tree model.  In addition because many sample are selected in the process a measure of variable importance can be obtain and this approach can be used for model selection and can be particularly useful when forward/backward stepwise selection is not appropriate and when working with an extremely high number of candidate variables that need to be reduced.

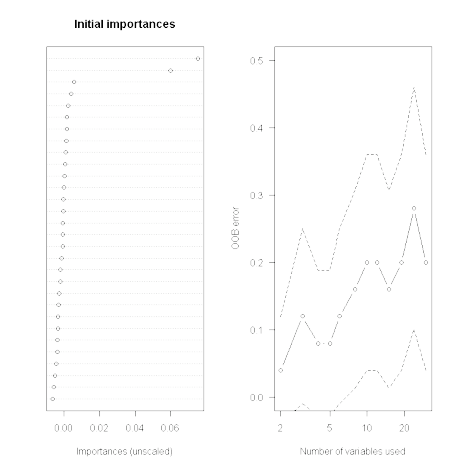
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | | ##################  ## randomForest  library(randomForest)  fit.rf = randomForest(frmla, data=raw)  print(fit.rf)  importance(fit.rf)  plot(fit.rf)  plot( importance(fit.rf), lty=2, pch=16)  lines(importance(fit.rf))  imp = importance(fit.rf)  impvar = rownames(imp)[order(imp[, 1], decreasing=TRUE)]  op = par(mfrow=c(1, 3))  for (i in seq\_along(impvar)) {  partialPlot(fit.rf, raw, impvar[i], xlab=impvar[i],  main=paste("Partial Dependence on", impvar[i]),  ylim=c(0, 1))  } | | |
|  | | >importance(rf1) | | |
|  | | %IncMSE | | IncNodePurity |
| x1 | | 30.30146 | | 8657.963 |
| x2 | | 7.739163 | | 3675.853 |
| x3 | | 0.586905 | | 240.275 |
| x4 | | -0.82209 | | 381.6304 |
| x5 | | 0.583622 | | 253.3885 |

[](http://i1.wp.com/statistical-research.com/wp-content/uploads/2012/12/ErrorAndImportance.png)

**varSelRF**

This can be used for further variable selection procedure using random forests.  It implements both backward stepwise elimination as well as selection based on the importance spectrum.  This data uses randomly generated data so the correlation matrix can set so that the first variable is strongly correlated and the other variables are less so.

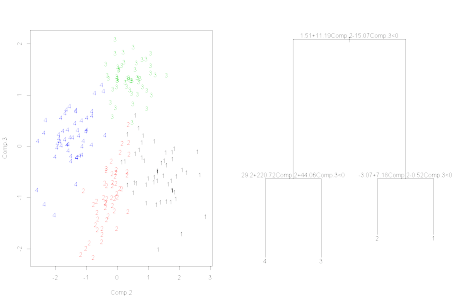
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | ##################  ## varSelRF package  library(varSelRF)  x = matrix(rnorm(25 \* 30), ncol = 30)  x[1:10, 1:2] = x[1:10, 1:2] + 2  cl = factor(c(rep("A", 10), rep("B", 15)))  rf.vs1 = varSelRF(x, cl, ntree = 200, ntreeIterat = 100,  vars.drop.frac = 0.2)    rf.vs1  plot(rf.vs1)    ## Example of importance function show that forcing x1 to be the most important  ## while create secondary variables that is related to x1.  x1=rnorm(500)  x2=rnorm(500,x1,1)  y=runif(1,1,10)\*x1+rnorm(500,0,.5)  my.df=data.frame(y,x1,x2,x3=rnorm(500),x4=rnorm(500),x5=rnorm(500))  rf1 = randomForest(y~., data=my.df, mtry=2, ntree=50, importance=TRUE)  importance(rf1)  cor(my.df) |

[](http://i1.wp.com/statistical-research.com/wp-content/uploads/2012/12/ImportanceOOBError.png)

**oblique.tree**

This package grows an oblique decision tree (a general form of the axis-parallel tree).  This example uses the crab dataset (morphological measurements on Leptograpsus crabs) available in R as a stock dataset to grow the oblique tree.

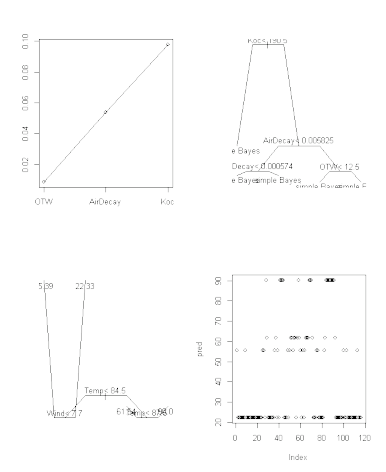
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | ###############  ## OBLIQUE.TREE  library(oblique.tree)    aug.crabs.data = data.frame( g=factor(rep(1:4,each=50)),  predict(princomp(crabs[,4:8]))[,2:3])  plot(aug.crabs.data[,-1],type="n")  text( aug.crabs.data[,-1], col=as.numeric(aug.crabs.data[,1]), labels=as.numeric(aug.crabs.data[,1]))  ob.tree = oblique.tree(formula = g~.,  data = aug.crabs.data,  oblique.splits = "only")  plot(ob.tree);text(ob.tree) |

[](http://i2.wp.com/statistical-research.com/wp-content/uploads/2012/12/ObliqueTree.png)

**CORElearn**

This is a great package that contain many different machine learning algorithms and functions.  It include trees, forests, naive Bayes, locally weighted regression, among others.

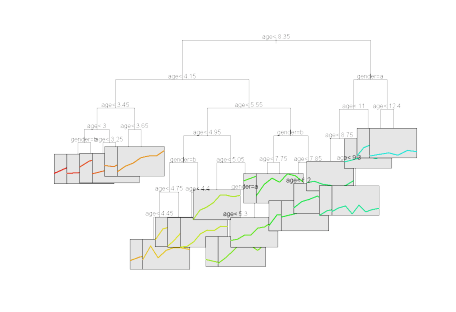
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | ##################  ## CORElearn    library(CORElearn)  ## Random Forests  fit.rand.forest = CoreModel(frmla, data=raw, model="rf", selectionEstimator="MDL", minNodeWeightRF=5, rfNoTrees=100)  plot(fit.rand.forest)    ## decision tree with naive Bayes in the leaves  fit.dt = CoreModel(frmla, raw, model="tree", modelType=4)  plot(fit.dt, raw)    airquality.sub = subset(airquality, !is.na(airquality$Ozone))  fit.rt = CoreModel(Ozone~., airquality.sub, model="regTree", modelTypeReg=1)  summary(fit.rt)  plot(fit.rt, airquality.sub, graphType="prototypes")    pred = predict(fit.rt, airquality.sub)  print(pred)  plot(pred) |

[](http://i2.wp.com/statistical-research.com/wp-content/uploads/2012/12/CORElearn.png)

**longRPart**

This provides an implementation for recursive partitioning for longitudinal data.  It uses the rules from *rpart* and the mixed effects models from *nlme* to grow regression trees. This can be a little resource intensive on some slower computers.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ##################  ##longRPart  library(longRPart)    data(pbkphData)  pbkphData$Time=as.factor(pbkphData$Time)  long.tree = longRPart(pbkph~Time,~age+gender,~1|Subject,pbkphData,R=corExp(form=~time))  lrpTreePlot(long.tree, use.n=TRE, place="bottomright") |

[](http://i2.wp.com/statistical-research.com/wp-content/uploads/2012/12/longRTreeplot.png)

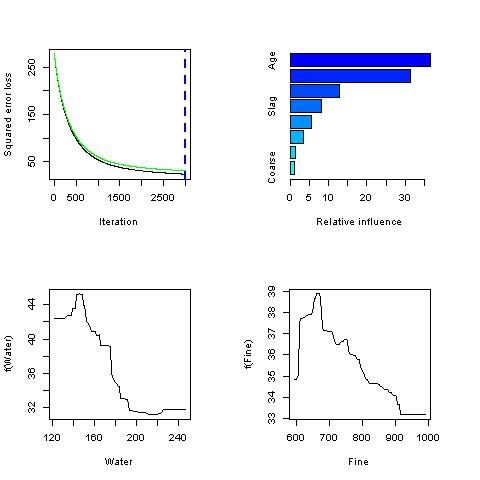
**REEMtree**

This package is useful for longitudinal studies where random effects exist.  This example uses the *pbkphData* dataset available in the *longRPart*package.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | ##################  ## REEMtree Random Effects for Longitudinal Data  library(REEMtree)  pbkphData.sub = subset(pbkphData, !is.na(pbkphData$pbkph))  reem.tree = REEMtree(pbkph~Time, data=pbkphData.sub, random=~1|Subject)  plot(reem.tree)  ranef(reem.tree) #random effects    reem.tree = REEMtree(pbkph~Time, data=pbkphData.sub, random=~1|Subject,  correlation=corAR1())  plot(reem.tree) |

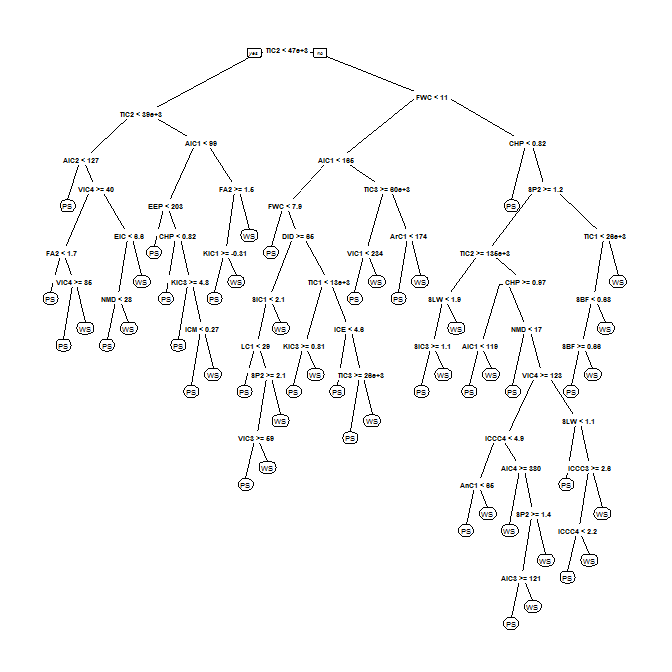
[inShare](javascript:void(0);)1

**Related**

[](http://www.r-bloggers.com/ensemble-packages-in-r/)

[Ensemble Packages in R](http://www.r-bloggers.com/ensemble-packages-in-r/)

In "R bloggers"

[](http://www.r-bloggers.com/draw-nicer-classification-and-regression-trees-with-the-rpart-plot-package/)

[Draw nicer Classification and Regression Trees with the rpart.plot package](http://www.r-bloggers.com/draw-nicer-classification-and-regression-trees-with-the-rpart-plot-package/)

In "R bloggers"

[rxDTree(): a new type of tree algorithm for big data](http://www.r-bloggers.com/rxdtree-a-new-type-of-tree-algorithm-for-big-data/)

by Joseph Rickert The rxDTree() function included in the RevoScaleR package distributed with Revolution R Enterprise is an an example of a new class of algorithms that are being developed to deal with very large data sets. Although the particulars differ, what these algorithms have in common is the use…

In "R bloggers"