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

<https://onlinecourses.science.psu.edu/stat857/node/22>

<style>

body {

background-image: url(41335688-beautiful-tree.jpg);

background-position: center center;

background-attachment: fixed;

background-repeat: no-repeat;

background-size: 100% 100%;

padding:0;

margin:0;

}

.section .reveal .state-background {

background-image: url(41335688-beautiful-tree.jpg);

background-position: center center;

background-attachment: fixed;

background-repeat: no-repeat;

background-size: 100% 100%;

padding:0;

margin:0;

}

.reveal {

color: white;

text-shadow: 0 0 30px #000000, 0 0 40px #004d00;

}

.reveal h1 {

color: white;

text-shadow: 0 0 30px #000000, 0 0 40px #004d00;

}

.reveal h2 {

color: white;

text-shadow: 0 0 30px #000000, 0 0 40px #004d00;

}

.reveal h3 {

color: white;

text-shadow: 0 0 30px #003300, 0 0 40px #004d00;

}

.reveal h4 {

color: white;

text-shadow: 0 0 30px #000000, 0 0 40px #004d00;

}

.reveal p {

color: white;

text-shadow: 0 0 30px #000000, 0 0 40px #004d00;

}

.small-code pre code {

font-size: 1em;

text-shadow: 0 0 0px #000000, 0 0 0px #004d00;

}

.reveal section del {

color: #ff9900;

}

.reveal h1, .reveal h2, .reveal h3 {

word-wrap: normal;

-moz-hyphens: none;

}

.footer {

color: black;

background: #E8E8E8;

position: fixed;

top: 100%;

text-align: left;

width: 100%;

opacity : 0.55;

font-size: 10%;

}

.midcenter {

position: fixed;

top: 50%;

left: 50%;

}

.reveal h1, .reveal h2, .reveal h3 {

word-wrap: normal;

-moz-hyphens: none;

}

div.transbox {

background-color: rgba(0,77,0,0.4);

padding-top: 20px;

padding-right: 20px;

padding-bottom: 20px;

padding-left: 20px;

}

div.transbox p {

font-weight: bold;

color: rgba(255,255,255);

}

</style>

How to use tree based classification in R

========================================================

font-family: 'Georgia'

<div class="transbox">The goal: give some brief examples on a few approaches on growing trees and, in particular, the overview of R packages used for those purposes.</div><br>

<div class="transbox"><small>Disclaimer: all examples are based on the data from public webstorages or on the data shipped with the R packages.</small><br>

<small>Enveronnement and technologies: RStudio, Authoring R Presentations, R, Markdown, CSS, HTML5.</small><br></div>

<div class="footer"">Kate Makhnach, November 14, 2016</div>

Summary

========================================================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><ul>

<li>Introduction</li>

<li>Notation and elements</li>

<li>Settings</li>

<li>Variable combinations and missing values</li>

<li>Pruning</li>

<li>Advantages</li>

</ul></div>

\*\*\*

<div class="transbox"><ul>

<li>rpart </li>

<li>tree</li>

<li>party</li>

<li>maptree</li>

<li>partykit</li>

<li>evtree</li>

</ul></div>

Introduction

========================================================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><small><ul>

<li>can be used for both regression and classification problems</li>

<li>very different approach to classification than prototype methods</li>

<li>no centroids, no hyperplanes boundaries</li>

<li>show the probability of being in any hierarchical group</li>

<li>basic idea of decision trees and three elements</li>

<li>wide array of packages in R and packages combining</li>

</ul></small></div>

```{r, eval=F, echo=F}

# Decision trees can be used for both regression and classification problems. Here we focus on classification trees. 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.

# Classification trees are a very different approach to classification than prototype methods such as k-nearest neighbors. The basic idea of these methods is to partition the space and identify some representative centroids.

# They also differ from linear methods, e.g., linear discriminant analysis, quadratic discriminant analysis and logistic regression. These methods use hyperplanes as classification boundaries.

# Classification trees are a hierarchical way of partitioning the space. We start with the entire space and recursively divide it into smaller regions. At the end, every region is assigned with a class label.

# basic idea of decision trees;

# three elements

# impurity function

# estimate the posterior probabilities of classes in each tree node

# resubstitution error rate and the cost-complexity measure

# weakest-link pruning

# the best pruned subtrees are nested and can be obtained recursively

# method based on cross-validation for choosing the complexity parameter and the final subtree

# model averaging

# bagging procedure

# random forest procedure

# boosting approach

# 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.

```

A medical example

================================

<div class="footer">Kate Makhnach, November 14, 2016</div>

![alt text](medex.png)

```{r, eval=F, echo=F}

# One big advantage for decision trees is that the classifier generated is highly interpretable. For physicians, this is an especially desirable feature.

# In this example patients are classified into one of two classes: high risk versus low risk. It is predicted that the high risk patients would not survive at least 30 days based on the initial 24-hour data. There are 19 measurements taken from each patient during the first 24 hours. These include blood pressure, age, etc.

# First we look at the minimum systolic blood pressure within the initial 24 hours and determine whether it is above 91. If the answer is no, the patient is classified as high-risk. We don't need to look at the other measurements for this patient. If the answer is yes, then we can't make a decision yet. The classifier will then look at whether the patient's age is greater than 62.5 years old. If the answer is no, the patient is classified as low risk. However, if the patient is over 62.5 years old, we still cannot make a decision and then look at the third measurement, specifically, whether sinus tachycardia is present. If the answer is no, the patient is classified as low risk. If the answer is yes, the patient is classified as high risk.

# Only three measurements are looked at by this classifier. For some patients, only one measurement determines the final result. Classification trees operate similarly to a doctor's examination.

```

Notations

===============================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><ul>

<li>We will denote the feature space by $X$. Normally $X$ is a multidimensional Euclidean space.</li>

<li>The input vector is indicated by $X \epsilon X$ contains p features $X\_1$, $X\_2$, ... ,$X\_p$.</li>

<li>A node is denoted by $t$. We will also denote the left child node by $t\_L$ and the right one by $t\_R$ .</li>

<li>Denote the collection of all the nodes in the tree by $T$ and the collection of all the leaf nodes by $\hat T$.</li>

<li>A split will be denoted by $s$. The set of splits is denoted by $S$.</li>

</ul></div>

```{r, eval=F, echo=F}

# We will denote the feature space by X. Normally X is a multidimensional Euclidean space. However, sometimes some variables (measurements) may be categorical such as gender, (male or female). CART has the advantage of treating real variables and categorical variables in a unified manner. This is not so for many other classification methods, for instance, LDA.

# The input vector is indicated by X∈X contains p features X1,X2,⋯,Xp.

# Tree structured classifiers are constructed by repeated splits of the space X into smaller and smaller subsets, beginning with X itself.

# A node is denoted by t. We will also denote the left child node by tL and the right one by tR.

# Denote the collection of all the nodes in the tree by T and the collection of all the leaf nodes by T~.

# A split will be denoted by s. The set of splits is denoted by S.

```

Three elements

===============================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<figure>

<object width="700" height="500" data="https://onlinecourses.science.psu.edu//stat857/sites/onlinecourses.science.psu.edu.stat857/files/lesson13/defining\_nodes.swf"></object>

<figcaption>.</figcaption>

</figure>

```{r, eval=F, echo=F}

# We will also need to introduce a few additional definitions: node, terminal node (leaf node), parent node, child node.

```

How these splits can take place.

================================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<figure>

<object width="700" height="500" data="https://onlinecourses.science.psu.edu/stat857/sites/onlinecourses.science.psu.edu.stat857/files/lesson13/splits\_viewlet\_swf.html"></object>

<figcaption></figcaption>

</figure>

```{r, eval=F, echo=F}

# Let's take a look at how these splits can take place.

# The whole space is represented by X.

# One thing that we need to keep in mind is that the tree represents recursive splitting of the space. Therefore, every node of interest corresponds to one region in the original space. Two child nodes will occupy two different regions and if we put the two together, we get the same region as that of the parent node. In the end, every leaf node is assigned with a class and a test point is assigned with the class of the leaf node it lands in.

# https://onlinecourses.science.psu.edu/stat857/sites/onlinecourses.science.psu.edu.stat857/files/lesson13/splits\_viewlet\_swf.html

# https://onlinecourses.science.psu.edu//stat857/sites/onlinecourses.science.psu.edu.stat857/files/lesson13/defining\_nodes.swf

```

Settings

====================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><ul>

<li>a set $Q$ of binary questions $$\{Is x \epsilon A\}, A \epsilon X$$<li>

<li>goodness of split criterion $\Phi (s,t)$</li>

<li>a stop-splitting rule</li>

<li>a rule for assigning every terminal node to a class</li>

</ul></div>

```{r, eval=F, echo=F}

#The construction of a tree involves the following three general elements:

# 1. The selection of the splits, i.e., how do we decide which node (region) to split and how to split it?

# Need to define the pool of candidate splits that we might select from involves a set Q of binary questions. Basically, we ask whether our input x belongs to a certain region, A. We need to pick one A from the pool.

# The candidate split is evaluated using a goodness of split criterion F(s,t) that can be evaluated for any split s of any node t.

# 2. If we know how to make splits or 'grow' the tree, how do we decide when to declare a node terminal and stop splitting?

# A stop-splitting rule, i.e., we have to know when it is appropriate to stop splitting. One can 'grow' the tree very big. In an extreme case, one could 'grow' the tree to the extent that in every leaf node there is only a single data point. Then it makes no sense to split any farther. In practice, we often don't go that far.

# 3. We have to assign each terminal node to a class. How do we assign these class labels?

# Finally, we need a rule for assigning every terminal node to a class.

# Now, let's get into the details for each of these four decisions that we have to make...

```

Standard set of questions for suggesting possible splits

=============================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><small>If the input vector contains features of both categorical and ordered types:</small>

$$X=(X\_1, X\_2, ... ,X\_p)$$

$$X\_j - an \\_ ordered \\_ variable$$

$$c - a \\_ realvalued \\_ threshold$$

$$\{Is \\_ X\_j \leqq c?\}$$

<small>If $X\_j$ is categorical, say taking values from $\{1, 2, ... , M\}$:</small> $$\{Is \\_ X\_j \epsilon A?\}$$

where $A$ is any subset of $\{1, 2, ... , M\}$.

</div>

Determining goodness of split

===============================

<div class="footer">Kate Makhnach, November 14, 2016</div>

![alt text](split1.jpg)

\*\*\*

![alt text](split2.jpg)

The impurity function

=======================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><small>$K$ - the number of classes</br>

$p\_1, p\_2, ..., p\_K$ - the probabilities for any data point in the region belonging to class 1, 2,..., K</br>

<span style="font-weight:bold; color:orange;">An impurity function</span> is a function $\Phi$ defined on the set of all K-tuples of numbers $(p\_1, ...,p\_K)$ satisfying $p\_j \geqq 0$, $j=1, ...,K$, $\sum\_jp\_j=1$ with the properties:</br></br>

<ol>

<li>$\Phi$ achieves maximum only for the uniform distribution, that is all the $p\_j$ are equal.</li>

<li>$\Phi$ achieves minimum only at the points $(1, 0, ... , 0), (0, 1, 0, ... , 0), ..., (0, 0, ... , 0, 1)$.</li>

<li>$\Phi$ is a symmetric function of $p\_1, ...,p\_K$.</li>

</ol></small></div>

Possible impurity functions:

============================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox">

Entropy function: $$\sum\_{j=1}^{K} p\_j \log \frac{1}{p\_j}$$

If $p\_j = 0$, use the limit $$\lim p\_j \rightarrow \log p\_j=0$$

Misclassification rate: $$1- \max jp\_j$$

Gini index: $$\sum\_{j=1}^{K} p\_j(1-p\_j)=1- \sum\_{j=1}^{K} p\_j^2$$

</div>

The Twoing rule

============================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><small>Another splitting method is the Twoing Rule. The intuition here is that the class distributions in the two child nodes should be as different as possible and the proportion of data falling into either of the child nodes should be balanced. At node $t$, choose the split $s$ that maximizes:

$$\frac{p\_Lp\_R}{4} \left[ \sum\_{j} |p(j|t\_L)-p(j|t\_R)|\right]^2$$

When we break one node to two child nodes, we want the posterior probabilities of the classes to be as different as possible. If they differ a lot, each tends to be pure. If instead the proportions of classes in the two child nodes are roughly the same as the parent node, this indicates the splitting does not make the two child nodes much purer than the parent node and hence not a successful split.

</small></div>

Determining stopping criteria

==========================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox">$$\max\_{s \epsilon S} \Delta I(s,t) < \beta$$</div>

<figure>

<object width="700" height="500" data="https://onlinecourses.science.psu.edu/stat857/sites/onlinecourses.science.psu.edu.stat857/files/lesson13/checkboard\_viewlet\_swf.html"></object>

<figcaption></figcaption>

</figure>

Determining class assignment rules

==============================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><small>A class assignment rule assigns a class $j=1, ...,K$ to every terminal (leaf) node $t \epsilon \hat T$. The class is assigned to node $t$. $\hat T$ is denoted by κ(t), e.g., if κ(t)=2, all the points in node t would be assigned to class 2.</br></br>

For example if we use 0-1 loss, the class assignment rule is very similar to k-means (where we pick the majority class or the class with the maximum posterior probability):

$$κ(t)= \arg \max\_j p(j|t)$$

</small></div>

Variable combinations

==========================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox">How to avoid the restriction of only partitions the space by hyperplanes parallel to the coordinate planes?</br></br>

<ul>

<li>There are classification tree extensions which, instead of thresholding individual variables, perform LDA for every node.</li>

<li>We could use more complicated questions. For example, use linear combinations of variables: $$\sum a\_jx\_j \leq c?$$</li>

</ul>

</div>

Missing values

==========================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><ul>

<li>surrogate or replacement split</li>

<li>approximate the result of the best split</li>

<li>no guarantee the second best split divides data similarly</li>

<li>goodness measurements are close</li>

</ul>

</div>

Notion of Pruning

===============================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><small>remove nodes that do not provide additional information</small></div>

<figure>

<object width="700" height="500" data="https://onlinecourses.science.psu.edu/stat857/sites/onlinecourses.science.psu.edu.stat857/files/lesson13/tree\_parts\_viewlet\_swf.html"></object>

<figcaption></figcaption>

</figure>

Pruning example

===============================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><small>remove nodes that do not provide additional information</small></div>

<figure>

<object width="700" height="500" data="https://onlinecourses.science.psu.edu/stat857/sites/onlinecourses.science.psu.edu.stat857/files/lesson13/pruning\_viewlet\_swf.html"></object>

<figcaption></figcaption>

</figure>

Advantages of tree-structured classification methods.

==================================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><ul>

<li>handles both categorical and ordered variables in a simple and natural way</li>

<li>sometimes do an automatic stepwise variable selection and complexity reduction</li>

<li>provide an estimate of the misclassification rate for a test point</li>

<li>invariant under all monotone transformations of individual ordered variables</li>

<li>relatively robust to outliers and misclassified points in the training set</li>

<li>easy to interpret</li>

</div>

```{r, eval=F, echo=F}

# As we have mentioned many times, the tree structured approach handles both categorical and ordered variables in a simple and natural way.

# Classification trees sometimes do an automatic stepwise variable selection and complexity reduction.

# They provide an estimate of the misclassification rate for a test point. For every data point, we know which leaf node it lands in and we have an estimation for the posterior probabilities of classes for every leaf node. The misclassification rate can be estimated using the estimated class posterior.

# Classification trees are invariant under all monotone transformations of individual ordered variables. The reason is that classification trees split nodes by thresholding. Monotone transformations cannot change the possible ways of dividing data points by thresholding.

# Classification trees are also relatively robust to outliers and misclassified points in the training set. They do not calculate any average or anything else from the data points themselves.

# Classification trees are easy to interpret, which is appealing especially in medical applications.

```

Package "rpart"

======================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r, eval=F, echo=T}

# set the working directory

# comma delimited data and no header for each variable

RawData <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data",sep = ",",header=FALSE)

responseY <- as.matrix(RawData[,dim(RawData)[2]])

predictorX <- as.matrix(RawData[,1:(dim(RawData)[2]-1)])

data.train <- as.data.frame(cbind(responseY, predictorX))

names(data.train) <- c("Y", "X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8")

library(rpart)

set.seed(19)

model.tree <- rpart(Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8, data.train, method="class")

plot(model.tree, uniform=T)

text(model.tree, use.n=T)

```

Classification tree obtained by the tree growing function

======================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r, eval=T, echo=F}

# set the working directory

setwd("C:/Users/User/Documents")

# comma delimited data and no header for each variable

RawData <- read.csv("exRPART.csv", stringsAsFactors = F)

RawData <- RawData[,-1]

responseY <- as.matrix(RawData[,dim(RawData)[2]])

predictorX <- as.matrix(RawData[,1:(dim(RawData)[2]-1)])

data.train <- as.data.frame(cbind(responseY, predictorX))

names(data.train) <- c("Y", "X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8")

library(rpart)

set.seed(19)

model.tree <- rpart(Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8, data.train, method="class")

plot(model.tree, uniform=T)

text(model.tree, use.n=T)

```

Pruning the tree

======================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r, eval=T, echo=T}

# “CP” - the complexity parameter

# the number of split - ”nsplit”

# ”xerror” - cross-validated classification error rates

# the standard deviation of the cross-validation error rates - ”xstd”

# !!!select a tree size that minimizes the cross-validated error (“xerror”)

model.tree$cptable

```

```{r, eval=F, echo=T}

opt <- model.tree$cptable[which.min(model.tree$cptable[,"xerror"]),"CP"]

model.ptree <- prune(model.tree, cp = opt)

plot(model.ptree, uniform=T)

text(model.ptree, use.n=T)

```

The optimized tree

======================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r, eval=T, echo=F}

# “CP” - the complexity parameter

# the number of split - ”nsplit”

# ”xerror” - cross-validated classification error rates

# the standard deviation of the cross-validation error rates - ”xstd”

# !!!select a tree size that minimizes the cross-validated error (“xerror”)

#model.tree$cptable

opt <- model.tree$cptable[which.min(model.tree$cptable[,"xerror"]),"CP"]

model.ptree <- prune(model.tree, cp = opt)

plot(model.ptree, uniform=T)

text(model.ptree, use.n=T)

```

Package "tree"

==============================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r, eval=T, echo=F,message = F}

library(rpart)

raw.orig = read.csv(file="http://people.hbs.edu/mtoffel/datasets/rsei212/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

#plotcp(fit) # visualize cross-validation results

#summary(fit) # detailed summary of splits

# 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)

```

```{r, eval=T, echo=T, message = F}

library(tree)

tr = tree(frmla, data=raw)

#summary(tr)

plot(tr); text(tr)

```

Package "party"

=============================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r}

library(party)

ct = ctree(frmla, data = raw)

plot(ct, main="Conditional Inference Tree")

#Table of prediction errors

table(predict(ct), raw$Metal)

# Estimated class probabilities

tr.pred = predict(ct, newdata=raw, type="prob")

```

Table of prediction errors

=============================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r}

#Table of prediction errors

table(predict(ct), raw$Metal)

# Estimated class probabilities

tr.pred = predict(ct, newdata=raw, type="prob")

```

Package "maptree"

===========================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r, eval=T}

library(maptree)

library(cluster)

draw.tree( clip.rpart (rpart ( raw), best=7),

nodeinfo=TRUE, units="species",

cases="cells", digits=0)

```

```{r, eval=F, echo=F}

###############

# MAPTREE

library(maptree)

library(cluster)

draw.tree( clip.rpart (rpart ( raw), best=7),

nodeinfo=TRUE, units="species",

cases="cells", digits=0)

a = agnes ( raw[2:4], method="ward" )

names(a)

a$diss

b = kgs (a, a$diss, maxclust=20)

plot(names(b), b, xlab="# clusters", ylab="penalty", type="n")

xloc = names(b)[b==min(b)]

yloc = min(b)

ngon(c(xloc,yloc+.75,10, "dark green"), angle=180, n=3)

apply(cbind(names(b), b, 3, 'blue'), 1, ngon, 4) # cbind(x,y,size,color)

```

Package "partykit"

=================================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r}

ds = read.csv("http://www.math.smith.edu/r/data/help.csv")

library(rpart); library(partykit)

ds$sub = as.factor(ds$substance)

homeless.rpart = rpart(homeless ~ female + i1 + sub + sexrisk + mcs +

pcs, method="class", data=ds)

plot(homeless.rpart)

text(homeless.rpart)

```

Package "partykit"

=================================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r}

printcp(homeless.rpart)

```

Package "evtree"

=================================================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r}

library(evtree)

ev.raw = evtree(frmla, data=raw)

plot(ev.raw)

```

Package "evtree"

==========================

class: small-code

<div class="footer">Kate Makhnach, November 14, 2016</div>

```{r}

table(predict(ev.raw), raw$Metal)

1-mean(predict(ev.raw) == raw$Metal)

```

Combining packages

======================

<div class="footer">Kate Makhnach, November 14, 2016</div>

<div class="transbox"><ul>

<li>repeat each other</li>

<li>graphics or methods</li>

<li>could not working tohehter</li>

<li>the tree package and the rpart package can produce very different results</li>

</ul></div>

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.

We discuss recursive partitioning, a technique for classification and regression using a decision tree in section 6.7.3 of the book (http://sas-and-r.blogspot.com.by/). Support for these methods is available within the rpart package. Torsten Hothorn and Achim Zeileis have extended the support for these methods with the partykit package, which provides a toolkit with infrastructure for representing, summarizing, and visualizing tree-structured regression and classification models.

In this entry, we revisit the example from the book, which worked to classify predictors of homelessness in the HELP study.

R

ds = read.csv("http://www.math.smith.edu/r/data/help.csv")

library(rpart); library(partykit)

ds$sub = as.factor(ds$substance)

homeless.rpart = rpart(homeless ~ female + i1 + sub + sexrisk + mcs +

pcs, method="class", data=ds)

plot(homeless.rpart)

text(homeless.rpart)

This reproduces Figure 6.2 (p. 236) from the book, while we can display the output from the classification tree using the printcp() command.

> printcp(homeless.rpart)

Classification tree:

rpart(formula = home ~ female + i1 + sub + sexrisk + mcs + pcs,

data = ds, method = "class")

Variables actually used in tree construction:

[1] female i1 mcs pcs sexrisk

Root node error: 209/453 = 0.5

n= 453

CP nsplit rel error xerror xstd

1 0.10 0 1.0 1.0 0.05

2 0.05 1 0.9 1.1 0.05

3 0.03 4 0.8 1.1 0.05

4 0.02 5 0.7 1.0 0.05

5 0.01 7 0.7 0.9 0.05

6 0.01 9 0.7 0.9 0.05

Using the partykit package, we can make a nice graphic describing these results. We'll use the plot.party() function on a party object (coerced from the rpart object generated above using as.party()). This provides more information about the tree (as seen in the Figure above).

plot(as.party(homeless.rpart), type="simple")

More information as well as a lovely vignette can be found here.

SAS

Recursive partitioning is available through SAS Enterprise Miner, a module not always included in SAS installations.

**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.

**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.

**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.

**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.

**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.