

XV Convegno Italiano degli Utenti di Stata Bologna, 15-16 November, 2018

Calling External Routines in Stata

Giovanni Cerulli
and
Antonio Zinilli

IRCrES-CNR



1

Motivation

- ❑ Stata allows to call **external routines**, written in other software, to perform specific tasks within Stata
- ❑ This talk offers some insights on how to develop a **Stata ADO file** embedding an external software routine (R, in this case)
- ❑ We provide a user-written Stata module **stree**, written to allow users to run regression trees (a Machine Learning technique currently unavailable in Stata) by calling back the R software

2

Three “R ==> Stata” alternatives

Rcall

Integrating R with Stata by allowing inter-process communication between the two software (Haghighi, E.F., 2017)



Very flexible, but a bit time-consuming to learn

Rsource

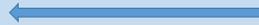
For running an R source program from an inline sequence of lines or from a file, in batch mode from within Stata



Very easy to use, but not really handy for ADO files

shell

Allowing to send commands to your operating system or to enter your operating system for interactive use



More general approach, apparently more complicated, but finally easy to use

3

The Basics of Decision Trees

Decision trees can be applied to both **regression** and **classification** problems

4

Example of a Decision Tree



- For the Hitters data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.
- At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to $\text{Years} < 4.5$, and the right-hand branch corresponds to $\text{Years} \geq 4.5$.
- The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.

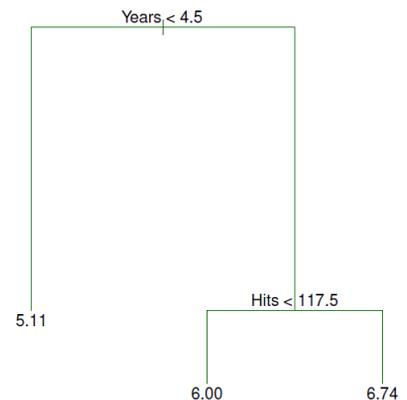
5

Interpretation of Results

Years is the most important factor in determining Salary , and players with less experience earn lower salaries than more experienced players.

Given that a player is less experienced, the number of Hits that he made in the previous year seems to play little role in his Salary .

But among players who have been in the major leagues for five or more years, the number of Hits made in the previous year does affect Salary , and players who made more Hits last year tend to have higher salaries.



6

Finding the **optimal** number of **terminal nodes**: Optimal-Tree detection

As other Machine Learning methods facing a **bias-variance trade-off**, the **optimal tree** is the one “balancing” *bias reduction* and *variance increase*, within the *largest* possible tree T_0 obtained from the *training* dataset.



The problem can be solved via a **penalization** approach, which penalizes too complex trees by at the same time allowing a not too large bias

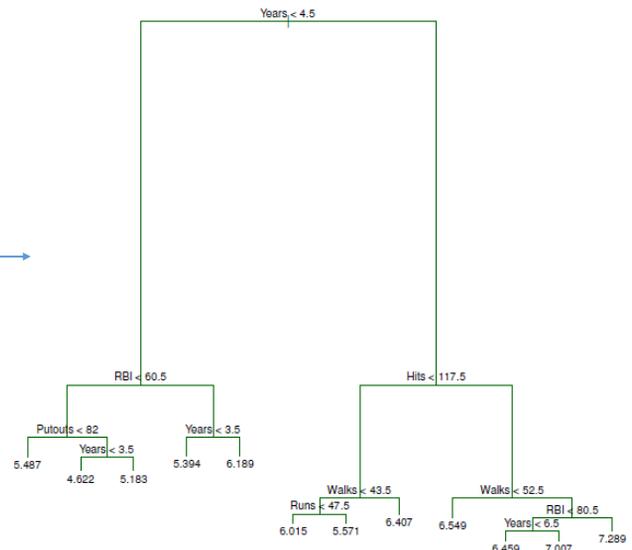


This can be done via **optimal tree-pruning**

7

Example - Regression tree for the **Hitters** data 1

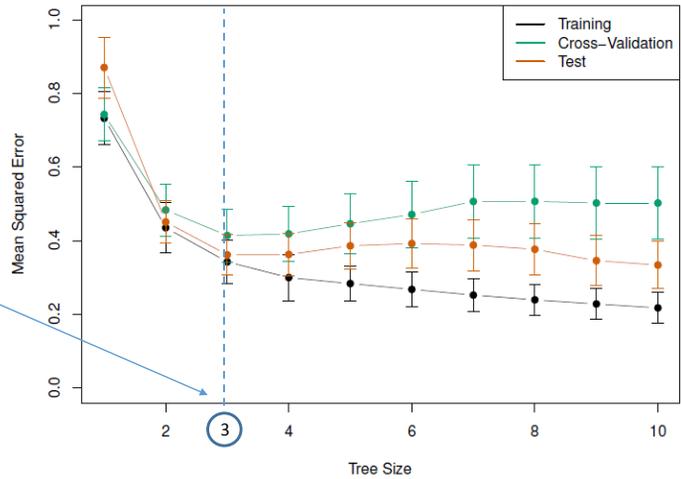
This is the **unpruned tree** that results from top-down greedy splitting on the training data.



8

Example - Regression tree for the Hitters data 2

The **minimum cross-validation error** occurs at a tree size of **3 nodes**

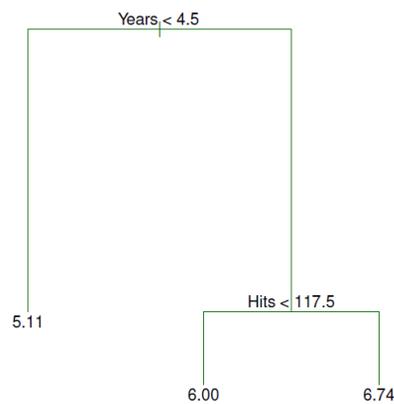


MSE for the **training**, the **cross-validation**, and the **test** as a function of the **number of terminal nodes** in the pruned tree.

9

Example - Regression tree for the Hitters data 3

3-node **optimal** tree



10

A Stata/R user-written ADO-file template

1. Write **srprog.ado**, a master Stata program calling back Stata sub-programs containing R code
2. Write **srprog1.ado**, **srprog2.ado**, ... the needed Stata sub-programs containing R code and generating an R program called **srprog.R**
3. Write the Stata program **runR.ado** executing **srprog.R** via the **shell** Stata command

11

A Stata/R user-written ADO-file template – step 1

Write a Stata program called **srprog**

Set the main directory as the present working directory (**pwd**)

Export the “.dta” dataset in the current memory into a “.csv” called “mydata.csv”

Run a program **srprog1** containing an R script conditionally on **option1**

Execute the Stata command **runR** to make Stata able to let R to do its job.

```

*****
capture program drop srprog
program define srprog
    syntax [something] [if] [in] [weights] , options
    *****
    local dir `c(pwd)'
    cd "`dir'"
    *****
    qui export delimited `var' using "mydata.csv", nolabel replace
    *****
    if "`option1'" != "" {
        srprog1 , options // Stata program containing an R script
        *****
        runR , options // Stata program running R
        *****
    }
end

```

12

A Stata/R user-written ADO-file template – step 2

1. Program called `srprog1`

2. Generate an R script called `srprog.R`

3. Write the R code instead of ...

```
*****
capture program drop srprog1
*****
program srprog1 // this does the most extended tree (baseline)
syntax ,
// Write R Code
quietly: file open rcode using srprog.R, write replace
quietly: file write rcode ///
    `"'`" _newline ///
    `"'`"
quietly: file close rcode
end
```

13

A Stata/R user-written ADO-file template – step 3

1. Stata program `runR`

2. Put the R program `srprog.R` into a local

3. Choose the operating system

4. Stata `shell` command to run `srprog.R`

```
*****
capture program drop runR
*****
program define runR
syntax , op_sys(string)
*****
// Run R from Stata
local using `"'`"srprog.R'`"'
*****
* MAC
if "`op_sys'"=="IOS"{
local rpath `"'`"/Library/Frameworks/R.framework/Resources/bin/R'`"'
local roptions `"'`"--slave'`"'
}
*****
* PC
else if "`op_sys'"=="WIN"{
local rpath `"'`"C:\Program Files\R\R-3.2.1\bin\x64\Rterm.exe'`"'
local roptions `"'`"--vanilla'`"'
}
*****
tempfile tempource tempis
copy `"'`"using'`"' `"'`"tempource'`"'
local Rcommand `"'`"rpath'`"' `"'`"roptions'`"' < `"'`"tempource'`"' > `"'`"tempis'`"'
shell `Rcommand'
*****
// Type R output in Stata
*****
disp_n as text "--> Beginning of R output from source file: " as result `"'`"using'`"'
type `"'`"tempis'`"'
disp_n as text "--> End of R output from source file: " as result `"'`"using'`"'
*****
end
```

14

The Stata user-written command **stree**

```
stree [anything] [if] [in] [weights] , model(modeltype)
op_sys(ostype) [prune(integer) cv_tree]
```

Options

model(modeltype) specifies the type of model, where:

```
-----
modeltype
-----
tree           Fits a tree, either unpruned, pruned, and optimal via CV
tree_rf        Fits a tree using random forests
tree_bag       Fits a tree using bagging
tree_boost     Fits a tree using the boosting algorithm
-----
```

op_sys(ostype) specifies the operating system you are working with. Two options for ostype are available, "WIN" (Windows) and "IOS" (MAC)

prune(integer) specifies the optimal pruned tree at size (number of nodes) "integer"; for instance prune(5), prune(8), ...

cv_tree specifies to run "cross-validation" in order to determine the optimal tree size

15

Application to a **classification tree** (using **sctree***)

```
* SET THE DIRECTORY
*cd "/Users/giocer/Dropbox/Giovanni/R/R_Cerulli/R into Stata/inout" // MAC
cd "C:\Users\Cerulli\Dropbox\Giovanni\R\R_Cerulli\R into Stata\inout" // PC

* LOAD THE DATASET
use carseats , clear

* TRANSFORM ALL THE "FACTOR-VARIABLES" INTO "STRINGS" FOR R TO READ THEM
global y "High"
global VARS "High US Urban ShelveLoc"
foreach V of global VARS{
  decode `V' , gen(`V'2)
  drop `V'
  rename `V'2 `V'
}

* SET THE OUTCOME, THE COVARIATES, AND THE PRUNING SIZE
global y "High"
global xvars "CompPrice Income Advertising Population Price Age Education US Urban ShelveLoc"
global M=6 // pruning size

* ESTIMATE THE CLASSIFICATION TREE
*sctree $y $xvars , model(tree) op_sys(WIN) // only overall classification tree + train-MSE
sctree $y $xvars , model(tree) op_sys(WIN) prune($M) // only tree of size M + train-MSE + test-MSE
*sctree $y $xvars , model(tree) op_sys(WIN) cv_tree // cross-validation with optimal tree size
```

16

* For fitting a **regression tree** the companion command is called **srtree**

R output visible as Stata output - 1

```

--> Beginning of R output from source file: mytree.R

R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R è un software libero ed è rilasciato SENZA ALCUNA GARANZIA.
Siamo ben lieti se potrai redistribuirlo, ma sotto certe condizioni.
Scrivi 'license()' o 'licence()' per dettagli su come distribuirlo.

R è un progetto di collaborazione con molti contribuiti esterni.
Scrivi 'contributor()' per maggiori informazioni e 'citation()'
per sapere come citare R o i pacchetti di R nelle pubblicazioni.

Scrivi 'demo()' per una dimostrazione, 'help()' per la guida in linea, o
'help.start()' per l'help navigabile con browser HTML.
Scrivi 'q()' per uscire da R.

> list.of.packages <- c("foreign","tree","MASS","ISLR")
> new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])
> if(length(new.packages)) install.packages(new.packages, repos="http://cran.us.r-project.org")
> library("foreign")
> # Loading
> library(foreign)
> my_data <- read.csv("mydata.csv", sep=",")
> MD <- getwd()
> setwd(MD)
> data<-as.data.frame(my_data)
> y <- data[,1]
> x <- data[2:ncol(my_data)]
> data <-cbind(y,x)
> # Load libraries
> library("tree")
> library(MASS)
> # Fit a Regression Tree over the whole training sample
> tree.data=tree(y~.,data)
> #summary(tree.data)
> #plot(tree.data, main="")
> #text(tree.data,pretty=0)
> #title(main="Regression Tree over the whole training sample")
> # Grow the pruned at size A
> prune.data=prune.tree(tree.data,best=6)
> #summary(prune.data)

```

17

R output visible as Stata output - 2

```

Classification tree:
snip.tree(tree = tree.data, nodes = c(41, 201, 421, 431, 111,
31))
Variables actually used in tree construction:
[1] "ShelveLoc" "Price" "Advertising" "CompPrice"
Number of terminal nodes: 6
Residual mean deviance: 0.964 = 379.8 / 394
Misclassification error rate: 0.2325 = 93 / 400
> plot(prune.data)
> text(prune.data,pretty=0)
> title(main="Pruned tree over the training sample at size 6")
> # Estimate the train-MSE at tree size 6
> yhat=predict(prune.data,type="class")
> table(yhat,y)
y
yhat No Yes
No 158 15
Yes 78 149
> # Show the sequence of pruned trees
> attach(tree.data)
> prune.data.seq=prune.tree(tree.data)
> summary(prune.data.seq)
Length Class Mode
size 22 -none- numeric
dev 22 -none- numeric
k 22 -none- numeric
method 1 -none- character
> # Estimate the test-MSE at tree size 6
> attach(data)
> set.seed(1)
> train = sample(1:nrow(data), nrow(data)/2)
> prune.data.test=prune.tree(tree.data,best=6,newdata=data[-train,])
> yhat=predict(tree.data,best=6,newdata=data[-train,],type="class")
> data.test=data[-train,"y"]
> table(yhat,data.test)
data.test
yhat No Yes
No 107 6
Yes 9 78
>
--> End of R output from source file: mytree.R

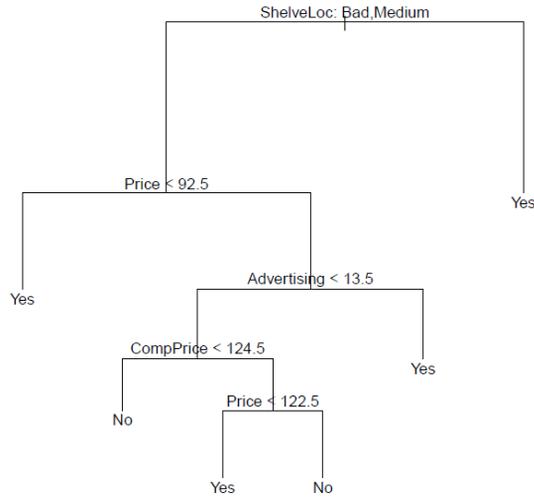
. *stree $y $xvars , model(tree) op_sys(MIN) cv_tree // cross-validation with optimal tree size

```

18

R output visible as Stata output - 3

Pruned tree over the training sample at size 6

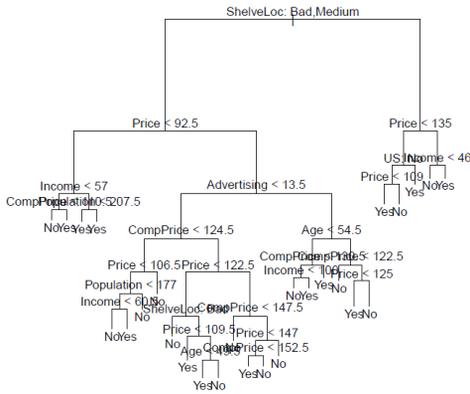


19

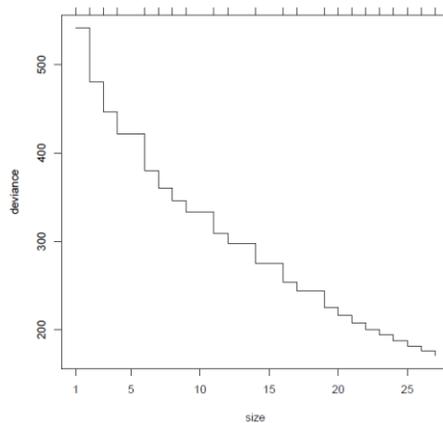
Optimal tree size via cross-validation - 1

```
sctree $y $xvars , model(tree) op_sys(WIN) cv_tree
```

Regression Tree over the whole training sample

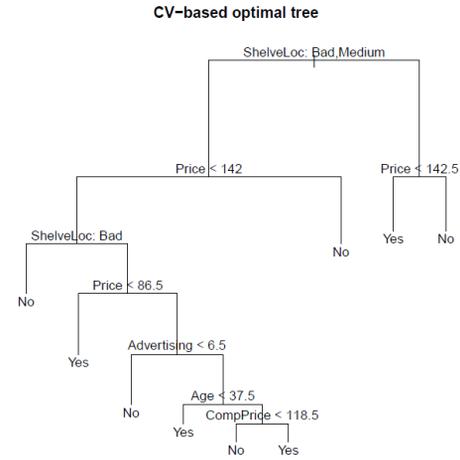
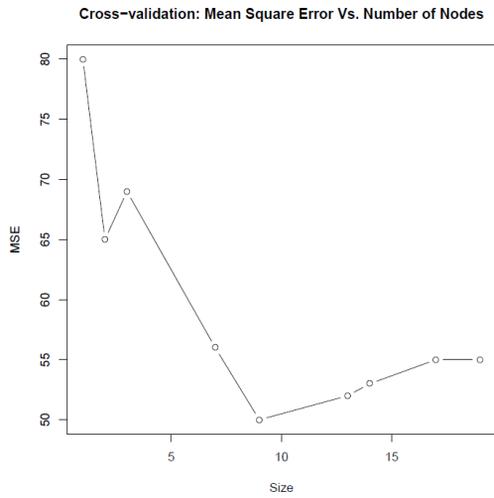


RSS sequence of pruned trees over the whole training sample



20

Optimal tree size via cross-validation - 2



21

Thanks for your attention

22