CIAT

Centro Internacional de Agricultura Tropical
International Center for Tropical Agriculture

Poverty survey Honduras

Statistical treatment

Thierry Fuhs, Grégoire Leclerc

September 1999

CGIAR Consultative Group on International Agricultural Research

Poverty survey Honduras
Statistical treatment
Preliminary report

Donor: BID

Writer: Thierry Fuhs, Consultant (LISC, Cemagref, France)
Consulting in CIAT- Cali, Colombia.

Requestor: Grégoire Leclerc (GIS Lab, CIAT)

Date: 20-28/09/1999

Contents

1 Introduction

The work conducted at Cali during the week 38 (20/09/1999 – 27/09/1999), has started the collaboration between the LISC lab. of Cemagref and the GIS Lab. of CIAT. Grégoire Leclerc from CIAT had asked Thierry Fuhs from Cemagref, for a statistical help to deal with the results of the poverty survey in Honduras, funded by BID.

The planned global contribution involved the following points.

1. Coding interviews into SPLUS from Excel templates.
2. Provide technical assistance for the interpretation of the results.
3. Write SPLUS scripts for the analysis, in coordination with G. Leclerc and A. Braun. Run the scripts on final data.
4. Write a short report, and collaboration for a peer-reviewed paper.
5. Provide additional technical assistance for the development of extrapolation methods.

Points 1 to 4 above have been partially fulfilled during Th. Fuhs's visit to Cali. This document is the preliminary report of the work done insofar. It is organized as follows. Section 2 presents the issues raised by the cleaning of data. Section 3 summarizes the construction of Ravnborg's well-being indexes. Section 4 shows preliminary attempts to use decision trees in the prediction of dependent variables. Eventually, section 5 provides the planned future work.

As stated above, S-Plus is the statistical software to be used for the whole study. Most part of the statistical work consists of *scripts*, saved in files with extension `.ssc`

2 Cleansing of data

Data collected on the field appeared to require some cleansing before being used for building relevant statistical models. This cleansing may be undertaken within a twofold process:

1. Precleansing and checking of keyed values. The survey is hand keyed with no help other than Excel automatic filling of fields. There are several mistakes in many variables (e.g. case errors like « x » and « X » meaning the same). This precleansing should ideally be done by the person who keyed the survey, with an optional double-check by another person. In the meanwhile, some introductory cleaning is available in the `PobCleanData.ssc` script.

2. After-loading cleansing. Even if perfectly keyed, the design of the survey does not ensure the direct usefulness of every variable. Some ones have to be merged (e.g. those reporting building materials of house), some have to be transposed (e.g. jornalea). In addition, variable names should be carefully checked. This second-level cleansing is (for part) done thanks to the `PobCleanData.ssc` script.

3 Generation of well being indexes

These indexes have been stated by Helle Ravnborg and adapted by G. Leclerc to the available variables. `PobFunctions.ssc` scripts details the way to build each of them.

3.1 Adapting Ravnborg's indexes

The current survey does not comprehend the whole bunch of variables used by Ravnborg to state her indexes. The computation of these indexes has had therefore to be slightly adapted to the available variables. The adaptations are summarized in the below table.

Index	Modification
PTIERRA	**None**
PJORNAL	**None**
PINGRESO	**Decay of "remittances for children" (not available data)**
PGANADO	**None**
PDINERO	**None**
PSALUD	**None**
PAGRICUL	**Extension of noble grains to any culture except maize and frijol, not only coffee and cacao**
PALIMENT	**Based only on food shortages and their relative lengths. Nothing available about money loans, reduction of the number of meals or impact on day laboring.**
PCASA	**None**
PANIMAL	**None**
PUSOJORN	**None**

3.2 The global index

All indexes have been gathered to yield a global well-being index, defined for each farmer. Missing values in specializes index have been excluded to build an everywhere defined global index.

4 Exploratory data analysis with decision tree models

4.1 Short description of decision trees

Decision trees are a statistical method aimed at regression as well as classification. They are built by a stepping procedure which first selects the best variable and its best level to discriminate the sample observations. Such a couple variable-level makes a test, classifying observations in two distinct groups. The process is then pursued recursively with each of these two subsets, eventually achieving a recursive partitioning of the sample.

In a second step, an optimal pruned tree (which is a subtree tree of the one obtained by the greedy first step) is selected by removing not statistically relevant branches. This selection is performed by applying a cross-validation procedure with an error criterion penalizing the size of the initial tree (see Breiman et al. (1984) for further details).

Moreover, decision trees provide the end-user with decisive advantages:

- They allow a direct interpretation of the tests since these ones involved one sole variable at a time.
- They manage both quantitative and qualitative variables, without need of recoding the latter.
- They can handle directly missing values.
- They include a direct variable selection feature, since a variable not present in the final tree is statistically non relevant according to the studied sample.

4.2 Use of decision tree methodology on survey data

In this first attempt, it is not so clear what variables needs to be predicted and from what variables. Anyway, several simple questions raised directly. For instance, we could relate the kind of cultures to the farm structure and to the farmer's description.

```
encues.tr0 <- tree(formula = QUE.SEMB.FRIJOL ~ SEXO.ENT + ENC.FAM + MADRE.SO.. +
   EDAD + ANO.EST +
   TENENCIA, data = encuesta, na.action = na.exclude, mincut = 4, minsize = 8,
   mindev = 0.01)
```

With age classes and merged levels for predicting if they plant beans or not, the quality we reached is as follows.

Before pruning

```
       h,m,x
h,m,x   161 117
         91 310
```

```
After pruning (the only meaningful quality)
       h,m,x
h,m,x   173 105
        110 291
```

i.e. a 33% global error (38% for 'yes' and 28% for 'no')

Test with a CART-like decision tree algorithm named `rpart` (i.e. using Gini index, not deviance as in the `tree` function) seems to show the irrelevance of the predicting variables chosen, even if the accuracy is not so bad (36% global). Further investigation is required to undarken this situation.

5 Planning of future work

The table below shows the planned calendar of future tasks with the person(s) in charge.

Task	Description	Due date	In charge
1	**Pre-cleansing and recoding of on-field data**	**15/10/99**	**GL + C?**
2	**Building of the final Splus data frame**	**30/10/99**	**GL, TF**
3	**Exploratory statistical analysis, including factorial analysis**	**30/10/99**	**GL,TF +JG**
4	**Decision tree development (S-Plus scripts)**	**15/11/99**	**TF**
	technical choice of the method	**10/11/99**	**TF**
	development	**20/11/99**	**GL - TF**
5	**Work about Helle's well-being indexes**		
	Availability of original data	**15/10/99**	**GL**
	Methodological review	**15/10/99**	**GL**
	Temptative rebuilding	**10/11/99**	**GL - TF**
6	**Spatialization of data**	**30/10/99**	**GL**
7	**Choice of mapping system**	**15/11/99**	**GL - TF**
8	**Choice of relevant technical problems to investigate**	**30/10/99**	**GL-AB-TF**
9	**Peer-reviewed paper**	**31/12/99**	**GL–TF - ?**

Further description of tasks

Task 1. Recoding (*to be defined*) and cleansing of on-field data by the one who digitizes them

Task 2. Building of the final Splus dataframe. Requires scripts for creation of derived data. These ones may be completely known such as well-being indices, but may also involve transposition or merging of genuine data. (Scripts : `PobCleanData.ssc` with help of `PobFunctions.ssc`)

Task 3. To lead a complete descriptive statistical analysis of the cleaned data. Besides standard summary statistics, precise factorial analysis is to be made.

Task 4. Decision tree development (scripts `PobTrees.ssc` and `PobRpart.ssc`)

- technical assesment: choice of the method (TF 10/11/99)
- development (GL-TF 20/11/99)

Task 5. Specific rebuilding of Ravnborg's indexes.

Task 6. Spatialization of data, according to the `hon-samp8h.xls` database.

Task 7. Choice of a mapping system capable to manage S-Plus data and methods. Includes:

- search for Central America S-Plus map

- ArcView GIS-SPlus link investigation.

Task 8. Choice of one or two relevant questions to deal with for publications. (GL, Ann Braun, TF 30/10/99)

- A paper about applying decision trees in the preference and problem ranking domain ?
- A domain paper dealing with well-being indexes checked or produced by decision trees ?

Task 9. Technical writing of peer-reviewed paper for the end of the year.

References

Breiman, L. *et al.* (1984) "Classification And Regression Trees", Chapman & Hall, New York.

Ravnborg, H. M. *et al.* (1999) "Developing regional poverty profiles based on local perceptions", CIAT, Cali, Colombia.

Venables W. N., Ripley B. D. (1997) "Modern applied statistics with S-Plus", Springer-Verlag. 2nd edition.

6 Data cleansing code

```
#################################################################
############
####
#### Name: PobCleanData.ssc
####
#### Author: Thierry Fuhs
####
#### Date: 20/09/99
####
#### Comment: script to clean the data of BID-Pobreza Honduras
Survey
####
#### Collaboration: CIAT GIS Gregoire Leclerc
####
#################################################################
###########


encuesta1 <- convert.col.type(target = encuesta1, column.spec
= list("NUMERO"), column.type = "integer")
encuesta2 <- convert.col.type(target = encuesta2, column.spec
= list("NUMERO"), column.type = "integer")
encuesta3 <- convert.col.type(target = encuesta3, column.spec
= list("NUMERO"), column.type = "integer")

dimnames(encuesta3)[[2]][1]
#eradicar las enquestas non entradas



#
#  para saber el numero de los encuestas vacias (?)
#
dim(encuesta1[encuesta1$ALDEA!="",])
dim(encuesta1[encuesta1$NOM.MUN!="",])
dim(encuesta1[encuesta1$NOM.DPTO!="",])
#el resultado dice que hay ALDEA que no tienen NOM.MUN y
NOM.DPTO
# 682 contra 679
#chequear las differencias
dim(encuesta1[encuesta1$NOM.MUN==""                              &
encuesta1$NOM.DPTO!="",])
#el resultado es 0, lo que muestra que no hay differencia
entre Municipio
# y departamento.
# Cuales son los aldeas concernadas
encuesta1[encuesta1$ALDEA!=""&encuesta1$NOM.MUN=="",][1:8]
```

```
# No problema para numeros 192 y 283, pero numero 699 esta
extrano (?)

#eliminacion de las encuestas

encuesta1 <- encuesta1[encuesta1$NOM.MUN!="",]
dim(encuesta11)
#quedan 679 observaciones
#

#ESTUDIAR A LOS PROBLEMAS EN LOS NOMBRES DE LUGAR

dim(summary(encuesta11$NOM.DPTO))-1
dim(summary(encuesta11$NOM.MUN))-1
dim(summary(encuesta11$ALDEA))

summary(encuesta11$NOM.DPTO)
summary(encuesta11$NOM.MUN)
summary(encuesta11$ALDEA)

dim(table(casefold(encuesta11$NOM.MUN)))# enlever la casse
#queda 40 Municipios pero hay algunos problemas

#####  Aseado
#DPTO    ##Cuidado a los nivels que no cambian pero no hay
problema despues.
encuesta11[encuesta11$NOM.DPTO=="Fco.  Moraza",]  ["NOM.DPTO"]
<-"Fco Morazan"
encuesta11[encuesta11$NOM.DPTO=="Olancho",]   ["NOM.DPTO"]   <-
"olancho"
#Municipio  ##Idem
encuesta11[encuesta11$NOM.MUN=="la   labor",]["NOM.MUN"] <-"la
labor"
encuesta11[encuesta11$NOM.MUN=="san  jos\202",]["NOM.MUN"]  <-
"san jose"
encuesta11[encuesta11$NOM.MUN=="Sta  rita",]["NOM.MUN"]  <-"sta
rita"
## No hay san josé de Copan en la lista de municipios
encuesta11[encuesta11$NOM.MUN=="San          jose          de
copan",]["NOM.MUN"] <-"san jose"

#Aldea
encuesta11[encuesta11$ALDEA=="san  jos\202",]["ALDEA"]  <-"san
jose"
encuesta11[encuesta11$ALDEA=="descombro",]["ALDEA"]         <-
"descombros"
encuesta11[encuesta11$ALDEA=="Cerro  Azul",]["ALDEA"] <-"cerro
azul"
encuesta11[encuesta11$ALDEA=="Zoilabe",]["ALDEA"] <-"zoilabe"
```

```
encuesta11[encuesta11$ALDEA=="el    raizal",]["ALDEA"]    <-"el
raisal"
encuesta11[encuesta11$ALDEA=="aldea  nuevo",]["ALDEA"] <-"aldea
nueva"
encuesta11[encuesta11$ALDEA=="nueva  aldea",]["ALDEA"] <-"aldea
nueva"
# rapida verificacion con la lista de las aldeas
# no hay "terra fria" "el raisal"

##
##  Modificacion de los niveles de los variables cambiadas
##   Hay que hacer eso porque S queda todos los niveles desde
el empiezo.
##
encuesta11$NOM.DPTO <- factor(encuesta11$NOM.DPTO,
     levels=names(summary(encuesta11$NOM.DPTO)[summary(encuesta
11$NOM.DPTO)!=0]))
encuesta11$NOM.MUN <- factor(encuesta11$NOM.MUN,
     levels=names(summary(encuesta11$NOM.MUN)[summary(encuesta1
1$NOM.MUN)!=0]))
encuesta11$ALDEA<- factor(encuesta11$ALDEA,
     levels=names(summary(encuesta11$ALDEA)[summary(encuesta11$
ALDEA)!=0]))

levels(encuesta11$NOM.MUN )
levels(encuesta11$ALDEA )

####
#### Aseado de los nombres de las variables
####
##encuesta1
encuesta1.col <- dimnames(encuesta11)[[2]]

encuesta1.col[12] <- "ANO.EST"


#  (3.10) QUE.PROB.BOSQ (BOSQ necesario a causa de dobles)
encuesta1.col[22:23] <- paste("QUE.PROB.BOSQ", 1:2, sep="")

#(3.13) PORQ.CONS
encuesta1.col[27:34] <- paste("PORQ.CONS", 1:8 , sep="")

# (3.16) CUAL.PROB (de tierra)
encuesta1.col[36:38] <- paste("CUAL.PROB", 1:3, sep="")

# (4.1) QUE.SEMB
encuesta1.col[39] <- "QUE.SEMB.CAFE"
encuesta1.col[40] <- "QUE.SEMB.MAIZ"
encuesta1.col[41] <- "QUE.SEMB.FRIJOL"
encuesta1.col[42] <- "QUE.SEMB.PLATANO"
encuesta1.col[43] <- "QUE.SEMB.GUINEO"
```

```
encuesta1.col[44] <- "QUE.SEMB.CACAO"
encuesta1.col[45] <- "QUE.SEMB.MAICILLO"
encuesta1.col[46] <- "QUE.SEMB.TOMATE"
encuesta1.col[47] <- "QUE.SEMB.CANA"
encuesta1.col[48] <- "QUE.SEMB.YUCA"
encuesta1.col[49] <- "QUE.SEMB.PASTOS"
encuesta1.col[50] <- "QUE.SEMB.HUERTA"
encuesta1.col[51] <- "QUE.SEMB.OTRO"

# (5.4) PQ.LOCMA (porque sembrar locales variedades de maiz)
encuesta1.col[55:64] <- paste("PQ.LOCMA", 1:10, sep="")

# (5.6) RAZ.MEJ.M (porque sembrar variedades mejoradas)
encuesta1.col[66:74] <- paste("RAZ.MEJ.M", 1:9, sep="")

# (5.7) PROB.PROD.M (pb en la produccion de maiz)
encuesta1.col[75:77] <- paste("PROB.PROD.M", 1:3, sep="")

# (5.11) PQ.LOCFR (porque sembrar locales variedades de
frijol)
encuesta1.col[81:90] <- paste("PQ.LOCFR", 1:10, sep="")

# (5.13) RAZ.MEJ.F (porque sembrar variedades mejoradas de
frijol)
encuesta1.col[92:100] <- paste("RAZ.MEJ.F", 1:9, sep="")

# (5.14) PROB.PROD.F (pb en la produccion de frijol)
encuesta1.col[101:103] <- paste("PROB.PROD.F", 1:3, sep="")

# (7.3) CUAL.PRO (suelo)
encuesta1.col[114:116] <- paste("CUAL.PRO", 1:3, sep="")

# (7.4) CAP.NEG
encuesta1.col[117:122] <- paste("CAP.NEG", 1:6, sep="")

# (7.8) QUE.LAVA
encuesta1.col[126:134] <- paste("QUE.LAVA", 1:9, sep="")

# (7.9) PQ.NOLAV
encuesta1.col[135:141] <- paste("PQ.NOLAV", 1:7, sep="")

# (7.10) PREP.SUE
encuesta1.col[142:145] <- paste("PREP.SUE", 1:4, sep="")

# (7.14) TIPO.ABON
encuesta1.col[149:155] <- paste("TIPO.ABON", 1:7, sep="")

# (7.15) UTIL.ABON
encuesta1.col[156:160] <- paste("UTIL.ABON", 1:5, sep="")

# (7.17) ABON.VERD
```

```
encuesta1.col[162:169] <- paste("ABON.VERD", 1:8, sep="")

# (8.2) QUIEN.SEM  (pastros) PB de duplicacion con una otra
variable
encuesta1.col[174] <- "QUIEN.SEM.P"

# (9.2) MES.SR
encuesta1.col[183:194] <- paste("MES.SR", 1:12, sep="")

# (9.3)  EPO.SRA
encuesta1.col[195:202] <- paste("EPO.SRA", 1:8, sep="")

# (9.4)  JORN.FAM
encuesta1.col[203:206] <- paste("JORN.FAM", 1:4, sep="")

# (9.5)  QYP.SALI
encuesta1.col[207:214] <- paste("QYP.SALI", 1:8, sep="")

##
##encuesta2
encuesta2.col <- dimnames(encuesta2)[[2]]

# (11.1)  TIEN.ANIM
encuesta2.col[2:7] <- paste("TIEN.ANIM", 1:6, sep="")

# (11.6)   VEN.DER
encuesta2.col[12:17] <- paste("VEN.DER", 1:6, sep="")

# (11.12)   COM.GAN
encuesta2.col[23:31] <- paste("COM.GAN", 1:9, sep="")

# (11.13) COMO.SUPLE
encuesta2.col[32:36] <- paste("COMO.SUPLE", 1:5, sep="")

# (11.14) PROB.CRIA
encuesta2.col[37:39] <- paste("PROB.CRIA", 1:3, sep="")

# (11.15)  CUAL.PLAG
encuesta2.col[40:44] <- paste("CUAL.PLAG", 1:5, sep="")

# (12.4) QUE.PROB.AGUA (AGUA necesario causa de dobles)
encuesta2.col[48:49] <- paste("QUE.PROB.AGUA", 1:2, sep="")

# (12.8)  AFECTA.CONT
encuesta2.col[53:54] <- paste("AFECTA.CONT", 1:2, sep="")

# (12.10)   QUE.HECHO
encuesta2.col[56:57] <- paste("QUE.HECHO", 1:2, sep="")

# (12.11)  NO.HECHO
encuesta2.col[58:59] <- paste("NO.HECHO", 1:2, sep="")
```

```
##
##encuesta3
encuesta3.col <- dimnames(encuesta3)[[2]]

# pb con acento
encuesta3.col[4] <- "CAFE.PR"

# (13.19)  QUE.PROB.COM (COM necesario a causa de dobles)
encuesta3.col[20:22] <- paste("QUE.PROB.COM", 1:3, sep="")

# (13.21)  QUE.OPORT
encuesta3.col[24:26] <- paste("QUE.OPORT", 1:3, sep="")

# (14.2)   PAREDES
encuesta3.col[28:34] <- paste("PAREDES", 1:7, sep="")

# (14.3)   TECHO
encuesta3.col[35:42] <- paste("TECHO", 1:8, sep="")

# (14.4)   PISO
encuesta3.col[43:47] <- paste("PISO", 1:5, sep="")

# (15.3)  PRESTAMO
encuesta3.col[50:55] <- paste("PRESTAMO", 1:6, sep="")

# (16.2)   QUE.EPOC
encuesta3.col[58:69] <- paste("QUE.EPOC", 1:12, sep="")

# (17.2)   PROB.SALUD
encuesta3.col[72:81] <- paste("PROB.SALUD", 1:10, sep="")

# (18.1)   PROB
encuesta3.col[82:84] <- paste("PROB", 1:3, sep="")

# (18.2) RAZON (a cambiar a causa de dobles)
encuesta3.col[85:93] <- paste(LETTERS[1:9], ".RAZ.PB", sep="")

### Verificacion de los dobles entre los nombres de variables
## la sola responsa debe ser "NUMERO" que es la llave para
fusionar
intersect(encuesta1.col, encuesta2.col)
intersect(encuesta1.col, encuesta3.col)
intersect(encuesta2.col, encuesta3.col)


## Applicacion de los nombres
dimnames(encuesta11)[[2]] <- encuesta1.col
dimnames(encuesta2)[[2]] <- encuesta2.col
dimnames(encuesta3)[[2]] <- encuesta3.col
```

```
#Creacion del dataframe completo (a hacer al final)
encuesta <- merge(merge(encuesta11, encuesta2, by="NUMERO"),
encuesta3, by="NUMERO")

dim(encuesta)
##   debe dar 679 observaciones, 378 variables

##
##   Cortado de variables
##

## Variables de produccion FRIJOL.PR etc. cortadas entre tipos
de venta y productador
## Las variables son raqueadas también siguiente 1<3<2<4<5

ciat.var.prod                                                  <-
c("FRIJOL.PR","MAIZ.PR","CAFE.PR","AGUAC.PR","PLATAN.PR",

     "TOMAT.PR","CHIL.PR","CEB.PR","REP.PR","LECH.PR",

     "YUCA.PR","MUEB.PR","ASERR.PR","AV.HUE.PR","QUES.PR","MANT
.PR")
for (nam in ciat.var.prod)
     { encuesta <- ciat.deploy.production(encuesta,nam)}

##
## Rectificacion de los tipos de variables

encuesta <- convert.col.type(target = encuesta,
               column.spec                                     =
list("SEXO.ENT","MADRE.SO..","ENC.FAM",

     "TENENCIA","PROB.TIER","AREA.MAIZ","PROB.PROD.M1",

     "AREA.FRIJ","FERTIL","PROB.ESP","PREP.TIER",

     "LAVA.SUE","PRE.LAVA","TOMA.DEC","MEJORAR","QUE.MEJO",

     "DEC.ABONO","MALEZAS","NUEVA.MALE","CAMB.PLAN",

     "POTRERO","JORNALEA","EMPLEADO","OFICIO","PROFESIO",

     "NEGOCIO","COMERCIAL","COMIDAS","DOMESTIC","TRA.PRE",

     "JORSIEMB","JORDESYE","JORCOSEC","JORFUMIG","USTEDJOR",

     "CONT.TRA","CUA.VACA","CAB.MANZ","ALIM.GANA","PRO.GAN",

     "COM.VEND.G","VEN.LECH","SUPLE.GAN","FUE.AGUA",
                          "TIE.AGUA","PROB.CONS",
"DIS.CANT","DIS.CALI",
```

```
     "PROB.CONT","DISM.CONT","COM.GRAN","PROB.COMER",
                          "MEJ.VIDA",
"TENE.CASA","AHORROS","PRESAOTRO","ALIMENTA",
                          "RANGOSAL","ENFERMO"
                          ),
               column.type = "factor")

encuesta <- convert.col.type(target = encuesta,
               column.spec                                    =
list("VAR.MAIZ","LOC.MAIZ","CUAN.MEJ.M",
                          "VAR.FRIJ","LOC.FRIJ","CUAN.MEJ.F",
                          "AREA.POT","CUAN.AREA","CUA.AREAPO"),
               column.type = "integer")

#################
####
####       MODIFICACION DE VARIABLES
####


###
### TRASPOSICION (JORN.FAM)
###
## 1. Aseado (que no se debe permanenzar)

## Ha sido hecho a la mano sobre los archivos Excel
## Los niveles han sido elegidos como 1.2.3 por ejemplo
##
##   2. Calculacion de las nuevas variables
##
encuesta <- insert.col(encuesta, "JORN.FAM1", 3,

     column.names=c("JORN.FREQ.SR","JORN.FREQ.SRA","JORN.FREQ.H
IJ"),
                               column.type= "factor")

matoo <-ciat.var.transpose(encuesta,"JORN.FAM",4,3)
encuesta$JORN.FREQ.SR <- ordered(matoo[,1],levels=c(1,2,3,4))
encuesta$JORN.FREQ.SRA <- ordered(matoo[,2],levels=c(1,2,3,4))
encuesta$JORN.FREQ.HIJ <- ordered(matoo[,3],levels=c(1,2,3,4))
remove("matoo")


##
### FUSION de las variables esclusivas
##

####
####   QUE HACER PARA LAS RESPUESTAS ABIERTAS !?!?!?
####           Cambiar "otro" por une categoria existente
```

```
####
# PAREDES
encuesta <- insert.col(encuesta, "PAREDES1", 1,
                                column.names="PAREDES",
column.type= "factor")
encuesta$PAREDES      <-      ordered(ciat.var.merge(encuesta,
"PAREDES", 6),levels=c(1,2,4,3,5,6))

#TECHO
encuesta <- insert.col(encuesta, "TECHO1", 1,
                                column.names="TECHO",
column.type= "factor")
encuesta$TECHO  <-  ordered(ciat.var.merge(encuesta,  "TECHO",
7),levels=c(1,2,7,3,4,5,6))

#PISO
encuesta <- insert.col(encuesta, "PISO1", 1,
                                column.names="PISO", column.type=
"factor")
encuesta$PISO   <-   ordered(ciat.var.merge(encuesta,   "PISO",
5),levels=c(5,4,3,2,1))


#################################################################
###
###        Diccionario de las variables en espanol e inglés
###       Used by ciat.translate.varnames function
###
ciat.vardict <- dimnames(encuesta)[[2]]
ciat.vardict <- rbind(ciat.vardict, ciat.vardict)
dimnames(ciat.vardict)[[1]] <- c("espanol","english")
dimnames(ciat.vardict)[[2]] <- dimnames(encuesta)[[2]]
ciat.vardict[1,2:14] <-
          c("Aldea","Municipio","Departamento","Fecha",
               "Entrevistador","Nombre","Género","Encargado  de
fam.",
               "Madre   sola",   "Edad",   "Estudios","Tenencia
tierra", "Area",
               )
ciat.vardict[2,2:14] <-
          c("Aldea","Municipio","Departamento","Date",
               "Interviewer","Name","Sex","In charge",
               "Alone  parent",  "Age",  "Studies","Ownership",
"Surface",
               )
```

7 Well-being index code

```
#################################################################
############
####
#### Name: PobIndexComputation.ssc
####
#### Author: Thierry Fuhs
####
#### Date: 20/09/99
####
#### Comment: set of functions used for the computaton of well-
being indexes,
####      first stated by Ravnborg's work
####      (survey BID-Probreza Honduras)
####
#### Collaboration: CIAT GIS Gregoire Leclerc
####
#################################################################
############



#################################################################
#############
#################################################################
#############
####
#### Function: ciat.wb.indexes
#### Objective(s): COMPUTATION OF WELL-BEING INDEXES
####
#### Return: Dataframe of computed well-being indexes
####
#### Calls:  Computation of each index
####
#### Comments: Computes per se a mean index
####
#################################################################
#############

ciat.wb.indexes <-
function(encuesta, na = F)
{
     PTIERRA <- ciat.wb.indexes.PTIERRA(encuesta, na)
     PJORNAL <- ciat.wb.indexes.PJORNAL(encuesta, na)
     PINGRESO <- ciat.wb.indexes.PINGRESO(encuesta, na)
     PGANADO <- ciat.wb.indexes.PGANADO(encuesta, na)
     PDINERO <- ciat.wb.indexes.PDINERO(encuesta, na)
     PSALUD <- ciat.wb.indexes.PSALUD(encuesta, na)
     PAGRICUL <- ciat.wb.indexes.PAGRICUL(encuesta, na)
```

```
     PALIMENT <- ciat.wb.indexes.PALIMENT(encuesta, na)
     PCASA <- ciat.wb.indexes.PCASA(encuesta, na)
     PANIMAL <- ciat.wb.indexes.PANIMAL(encuesta, na)
     PUSOJORN <- ciat.wb.indexes.PUSOJORN(encuesta, na)
     df <- data.frame(PTIERRA,PJORNAL,PINGRESO,PGANADO,PDINERO,
PSALUD, PAGRICUL,
                         PALIMENT, PCASA, PANIMAL, PUSOJORN)
     PTOTAL <- apply(PINDEXES, 1, mean, na.rm=T)
     data.frame(PTOTAL,df)
}


################################
####
####           PTIERRA index
####
#### Comments: no problem
####
################################

ciat.wb.indexes.PTIERRA<-
function(encuesta, na=F)
{
     return(ifelse(encuesta$TENENCIA==1,
                    ifelse(ordered(encuesta$AREA)>=4          |
encuesta$POTRERO == 1, 33,67),
                    100))
}
###############   END OF PTIERRA INDEX

################################
####
####            PJORNAL index
####
####
#### Comments:
####      - extra computation of number of months of jornalea
for the
####      man
####      - makes an equivalence for the woman between "todo el
ano" and
####      every other values (every "cosecha")
####
################################
ciat.wb.indexes.PJORNAL <-
function(encuesta,na=F)
{
     MES.SR.TOT <- 0
     for (i in 1:12)
     {
          MES.SR.TOT <- MES.SR.TOT +
```

```
     pmin(1,as.integer(encuesta[[paste("MES.SR",i,sep="")]])-1)
     }

     return(ifelse(encuesta$JORNALEA==2,
                   ifelse(encuesta$DOMESTIC==2                    &
encuesta$COMIDAS==2, 33, 100),
                   ifelse(MES.SR.TOT <9,
                        ifelse(encuesta$EPO.SRA7!=""|
(encuesta$EPO.SRA1 != ""

     & encuesta$EPO.SRA2 != ""

     & encuesta$EPO.SRA3 != ""

     & encuesta$EPO.SRA4 != ""

     & encuesta$EPO.SRA5 != ""

     & encuesta$EPO.SRA6 != ""

     & encuesta$EPO.SRA8 != ""),100,67),
                        ifelse(encuesta$JORN.FREQ.SR >=2,

     ifelse(encuesta$EPO.SRA7!=""|(encuesta$EPO.SRA1 != ""

     & encuesta$EPO.SRA2 != ""

     & encuesta$EPO.SRA3 != ""

     & encuesta$EPO.SRA4 != ""

     & encuesta$EPO.SRA5 != ""

     & encuesta$EPO.SRA6 != ""

     & encuesta$EPO.SRA8 != ""), 100,67),
                              100))))
}
###############   END OF PJORNAL INDEX

#################################
####
####            PINGRESO index
####
####
#### Comments: no problem
####
#################################

ciat.wb.indexes.PINGRESO<-
```

```
function(encuesta, na=F)
{
     return(ifelse(encuesta$PROFESIO==1|encuesta$NEGOCIO==1|enc
uesta$COMERCIAL==1,
          33,
          ifelse(encuesta$OFICIO!=1, 100, 67)))
}
###############   END OF PINGRESO INDEX


################################
####
####       PGANADO  index
####
####
#### Comments: no problem
####
################################

ciat.wb.indexes.PGANADO<-
function(encuesta, na=F)
{
     return(ifelse(encuesta$TIEN.ANIM1==1,33, 67))
}
###############   END OF PGANADO INDEX

################################
####
####      PDINERO  index
####
####
#### Comments: no problem
####
################################

ciat.wb.indexes.PDINERO<-
function(encuesta, na=F)
{
     return(ifelse(encuesta$AHORROS==2&encuesta$PRESAOTRO==2,
          33,
          67))
}
###############   END OF PDINERO INDEX

################################
####
####  PSALUD  index
####
####
#### Comments: no problem
####
```

```
################################

ciat.wb.indexes.PSALUD <-
function(encuesta, na=F)
{
     return(ifelse(encuesta$ENFERMO==2,
          67,

     ifelse(encuesta$PROB.SALUD2!=""|encuesta$PROB.SALUD3!=""|e
ncuesta$PROB.SALUD9!="",
                              67,
                              100)))
}
###############   END OF PSALUD INDEX

################################
####
####      PAGRICUL  index
####
####      Comments :
####        NOBLE.PR  means  destination  of  noble  production
(exclude Frijol and maiz).
####           NOBLE.PR<=3 means noble production sold in most
part.
####      GRAN.USO means destination of all production.
####           GRAN.USO<=3 idem supra
####      care  to  be  taken  for  the  level  order  of  original
variables PROD.PR
####           (1<3<2<4<5)
####
####
################################

ciat.wb.indexes.PAGRICUL <-
function(encuesta, na=F)
{
     mat <- sapply(encuesta[,ciat.var.prod], as.integer)
     NOBLE.PR <- apply(mat[,-(1:2)], 1, min) # production of
nobles grains
     GRAN.USO <- apply(mat,1,min)                    #    production
totale
     return(ifelse(NOBLE.PR<=2, 33,
                    ifelse(encuesta$COM.GRAN==2,
                         ifelse(GRAN.USO<=2, 33, 66),
                         ifelse(GRAN.USO<=2, 66, 100))))
}
###############   END OF PAGRICUL INDEX


################################
####
```

```
####  PALIMENT index
####
####      Comments: not complete according to full definition
of index
####           but several variables are missing to make better
####
####
################################

ciat.wb.indexes.PALIMENT <-
function(encuesta, na=F)
{
     return(ifelse(encuesta$ALIMENTA==2, 67,
                    ifelse(encuesta$RANGOSAL==1, 67,100)))
}
###############   END OF PALIMENT INDEX

################################
####
####  PCASA index
####
####
#### Comments:  requires  the  merging  of  uselessly  exploded
variables
####      see ciat.var.merge function
####
################################

ciat.wb.indexes.PCASA <-
function(encuesta, na=F)
{
     return(ifelse(encuesta$TENE.CASA!=1                          &
encuesta$TENE.CASA!=5,
               100,
               ifelse(encuesta$PAREDES >=3 & encuesta$TECHO >=7
& encuesta$PISO >=1,
                    100,
                    ifelse(encuesta$PAREDES <3 & encuesta$TECHO
<7 & encuesta$PISO <1,
                         33,
                         67))))
}
###############   END OF PCASA INDEX

################################
####  PANIMAL index
####
####
####
#### Comments: no problem
####
```

```
################################


ciat.wb.indexes.PANIMAL <-
function(encuesta, na=F)
{
     return(
          ifelse(encuesta$TIEN.ANIM2!=""                              |
encuesta$TIEN.ANIM3!="" | encuesta$TIEN.ANIM5!="" ,
                33,
                ifelse(encuesta$TIEN.ANIM4 !="",
                     67,
                     100)))
}
###############   END OF PANIMAL INDEX

################################
####  PUSOJORN index
####
####
####
#### Comments: no problem
####
################################

ciat.wb.indexes.PUSOJORN <-
function(encuesta, na=F)
{
     return(
          ifelse(encuesta$JORSIEMB==1 | encuesta$JORDESYE==1 |
                     encuesta$JORCOSEC==1 | encuesta$JORFUMIG==1
,
                     33,
                     67))
}
###############   END OF PUSOJORN INDEX
```

8 Code of S-Plus functions

```
#################################################################
############
####
#### Name: PobFunctions.ssc
####
#### Author: Thierry Fuhs
####
#### Date: 20/09/99
####
#### Comment: set of functions used for preparing the data
####      of the survey BID-Probreza Honduras
####
#### Collaboration: CIAT GIS Gregoire Leclerc
####
#################################################################
############



#################################################################
#############
####
#### Function: ciat.deploy.production
#### Objective(s): Explode each PROD.PR variables in
####      two variables
####           - PROD.PR which becomes the production itself
and its destination
####           - PROD.PR.Q (new) which explains who did the
plantation
####
#### Return: The dataframe with new variables inserted
#### Comments: Supposes the substring to explode of the
     shape 1h, 2m...
####
####
#################################################################
############

ciat.deploy.production <-
function(enc, nam)
{
     namq <- paste(nam,".Q",sep="")
     if(is.element(namq,dimnames(enc)[[2]]))
     {
          warning("No change")
     }
     else
     {
```

```
          print(nam)
          PR <-substring(enc[[nam]],1,1)
          PR[PR==""] <- "5"
          PR <- ordered(factor(PR, levels=c(1,3,2,4,5)))
          levels(PR) <- list(1,3,2,4,5)
          Q <- factor(substring(enc[[nam]],2,2))
          enc.out <- insert.col(enc, nam,1,column.names=namq)
          enc.out[[nam]] <- PR
          enc.out[[namq]] <- Q
          return(enc.out)
     }
}
##################    END  of  ciat.deploy.production  function
###############

#################################################################
#############
####
#### Function: ciat.var.transpose
#### Objective(s): Variable transposition
####      The  variables  JORN.FAMi  were  not  well  designed.
Instead of
####      knowing "quien jornalea de vez en cuando", we needed
####      when "el senor jornalea"
####
#### Return: A dataframe with the new variables
#### Comments: - Still in a "hack" state. Not useful if there
are more than
####      3 modalities to transpose
####      -  requires  explicit  loops  (don't  find  out  a
vectorized way to do it)
####      - uses matrices to make the work
####
####
#################################################################
############

ciat.var.transpose <-
function(enc,namevar,numvar,nummod)
{
     mat <- as.matrix(enc[,paste(namevar,1:numvar,sep="")])
     nobs <- dim(enc)[1]
     mat.out <- matrix(nrow=nobs,ncol=nummod)
     for (i in 1:numvar)
     {
          for (j in 1:nobs)
          {
               switch(mat[j,i],  ### TF  HACK HACK
                              "1"= {mat.out[j,1] <- paste(i) },
                              "2"= {mat.out[j,2] <- paste(i)},
                              "3"= {mat.out[j,3] <- paste(i)},
```

```
                                "1.2"= {mat.out[j,1] <- paste(i);
mat.out[j,2] <- paste(i)},
                                "1.3"= {mat.out[j,1] <- paste(i);
mat.out[j,3] <- paste(i)},
                                "2.3"= {mat.out[j,2] <- paste(i);
mat.out[j,3] <- paste(i)},
                                "1.2.3"=     {mat.out[j,1]     <-
paste(i); mat.out[j,2] <- paste(i); mat.out[j,3] <- paste(i)})

          }
     }
               return(data.frame(mat.out))
}

##################      END   of   ciat.var.transpose   function
################

#####################################################################
#############
####
#### Function: ciat.var.merge
#### Objective(s): Variable merge
####      Some variables have been exploded following each
modalities
####      whereas those ones were exclusive.
####
#### Return: A vector containing the new variable
#### Comments:
####      - supposes modalities belong to set {"x","1","1");
this ensures
####           regexpr will yields -1 (no match) or 1.
####      - uses matrice multiplication to make the work easy
####
####
#####################################################################
############

ciat.var.merge <-
function(enc,namevar,numvar)
{
     mat <- as.matrix(enc[,paste(namevar,1:numvar,sep="")])
     matm       <-       matrix(pmax(0,       regexpr("[1x1]",
mat)),nrow=dim(mat)[1], ncol=dim(mat)[2])
     return(matx %*% 1:numvar)
}
##################       END    of    ciat.var.merge    function
########################
```

```
#################################################################
#############
####
#### Function: ciat.post.tree
#### Objective(s): POSTSCRIPT TREE IMPROVEMENT
####      Using a dictionary of variables, trap the labels of
the tree
####      defined by tree function to translate them if
necessary.
####
#### Return: None
#### Side effect: PostScript file creation
#### Calls:    ciat.translate.varnames
####                ciat.translate.val
#### Comments:
####        trap of two inner variables of the original
post.tree function:
####           - splits (names upon the links)
####           - inside (names inside the boxes, inner nodes or
leafs)
####
####
#################################################################
############

ciat.post.tree <-
function(tree, title = deparse(substitute(tree)), file =
paste(make.names(title), ".ps",
     sep = ""), digits = .Options$digits - 3, pretty = 0,
pointsize = 12, language="notrans")
{
     frame <- tree$frame
     n <- dim(frame)[1]
     ylevels <- attr(tree, "ylevels")
     xlevels <- attr(tree, "xlevels")
     splits <- labels.tree(tree, pretty = pretty, collapse = F)
     splits <- ciat.translate.varnames(splits, language)  ## TF
     if(length(ylevels)) {
          mer <- format(signif(misclass.tree(tree, detail = T),
digits = digits))
          below <- paste(mer, "/", frame$n, sep = "")
          inside <- frame$yval
     }
     else {
          below <- format(signif(frame$dev, digits = digits))
          inside <- format(signif(frame$yval, digits = digits))
     }
     inside <- ciat.translate.val(inside, language)   ## TF
     xy <- .C("post",
          as.integer(frame$var != "<leaf>"),
          as.character(splits[, 1]),
```

```
          as.character(splits[, 2]),
          as.character(inside),
          as.character(below),
          as.integer(row.names(frame)),
          as.integer(n),
          as.character(title),
          as.character(file),
          as.double(pointsize),
          x = double(n),
          y = double(n))
     invisible(list(x = xy$x, y = xy$y))
}
#################       END    of    ciat.var.merge    function
#########################

#####################################################################
#############
####
#### Function: ciat.translate.varnames
#### Objective(s): Link labels translation
####      Switch between the existing dictionaries
####
#### Return: Matrix of translated labels
####      (keeps the format yielded by the function labels.tree
from S-Plus)
#### Side effect: None
#### Calls:    ciat.translate.varnames.mk
####
#### Called by: ciat.post.tree
####
#### Comments: simple switch between english and spanish. the
actual work
####           is done by ciat.translate.varnames.mk
####           Should surely be simplified
####
####
#####################################################################
#############

ciat.translate.varnames<-
 function(splits,language="notrans")
{
     if(!exists("ciat.vardict"))
     {
          print("No dictionary")
          return(splits)
     }
     else
     {
          return(switch(language,
```

```
                    english= ciat.translate.varnames.mk(splits,
language="english"),
                    espanol=,                              spanish=
ciat.translate.varnames.mk(splits, language="espanol"),
                    splits))
     }
}
#################    END  of  ciat.translate.varnames  function
########################

#####################################################################
#############
####
#### Function: ciat.translate.varnames.mk
#### Objective(s): Link labels translation
####      Makes the actual work
####
#### Return: Matrix of translated labels
####      (keeps the format yielded by the function labels.tree
from S-Plus)
####
#### Side effect: None
####
#### Calls:    None
####
#### Called by: ciat.translate.varnames
####
#### Global var: ciat.vardict
####
#### Comments: uses sapply to apply the translation to each
label
####           - uses regexpr and substring to fetch and
replace the labels
####           - regexpr gives the place where the match begins
and keeps
####           in attribute "match.length" the size of the
match
####           - eventually gives the result the shape required
by ciat.post.tree
####
####
#####################################################################
#############

ciat.translate.varnames.mk<-
 function(splits, language)
{
     spl <- sapply(as.vector(splits), function(x,lang){
                         rx <- regexpr("^[a-zA-Z\\.]+[:<>]", x)
                         if (rx!=-1)
                              {
```

```
                                varkey                          <-
substring(x,rx,rx+attr(rx, "match.length")-2)

     paste(ciat.vardict[lang,][[varkey]],
                                        substring(x,
rx+attr(rx, "match.length")-1), sep="")
                               }
                               else
                               x},
                               lang=language)
     return(matrix(data=spl, ncol=2))
}
#################   END of ciat.translate.varnames.mk function
#########################

#####################################################################
#############
####
#### Function: ciat.translate.val
#### Objective(s): Node labels translation
####      Switch between the languages
####
#### Return: Vector of translated values
####
#### Side effect: None
#### Calls:    ciat.translate.val.es
####                ciat.translate.val.en
####
#### Called by: ciat.post.tree
####
#### Comments: simple switch between english and spanish. the
actual work
####           is done by ciat.translate.val.en and .es
####
####
#####################################################################
#############

ciat.translate.val<-
 function(inside, language="notrans")
{
     if (length(dictionary)
     switch(language,
                    english= ciat.translate.val.en(inside),
                    espanol=,                              spanish=
ciat.translate.val.es(inside),
                    inside)

}
#################      END   of   ciat.translate.val   function
#########################
```

```
#####################################################################
#############
####
#### Function: ciat.translate.varnames.mk
#### Objective(s): Link labels translation
####      Makes the actual work
####
#### Return: Vector of translated labels
####
#### Side effect: None
####
#### Calls:    None
####
#### Called by: ciat.translate.val
####
#### Comments: suppose a binary classification and that ""
means "No"
####
#####################################################################
#############

ciat.translate.val.en<-
 function(inside)
{
     zz <- inside !=""
     zz[zz==F] <- "No"
     zz[zz==T] <- "Yes"
     return(zz)
}

ciat.translate.val.es<-
 function(inside)
{
     zz <- inside !=""
     zz[zz==F] <- "No"
     zz[zz==T] <- "Si"
     return(zz)
}
########          END     of     ciat.translate.val.en     and
ciat.translate.val.es functions ####
```

9 Decision tree building code

```
#################################################################
############
####
#### Name: PobTree.ssc
####
#### Author: Thierry Fuhs
####
#### Date: 20/09/99
####
#### Comment: decision trees building with tree function
####           (survey BID-Probreza Honduras)
####
#### Collaboration: CIAT GIS Gregoire Leclerc
####
#################################################################
###########


######## UNTIL NOW, ONLY TRIALS ###########
###### "Real" problems still to be decided
######
###
###             1. Initial tree building
###

encues.tr0  <-  tree(formula  =  QUE.SEMB.MAIZ  ~  SEXO.ENT  +
ENC.FAM + MADRE.SO.. + EDAD + ANO.EST +
     TENENCIA, data = encuesta, na.action = na.exclude, mincut
= 4, minsize = 8,
     mindev = 0.01)

##
##   confusion matrix of the above tree
##
table(encuesta$QUE.SEMB.MAIZ,predict(encues.tr0,encuesta,type=
"class"))

encues.tr0 <- tree(formula = QUE.SEMB.MAIZ ~
                   SEXO.ENT  +  ENC.FAM  +  MADRE.SO..  +  EDAD  +
ANO.EST +
     TENENCIA, data = encuesta, na.action = na.exclude, mincut
= 4, minsize = 8,
     mindev = 0.01)

table(merge.levels(encuesta$QUE.SEMB.MAIZ,   list(c("h",   "m",
"x"))),predict(encues.tr0,encuesta,type="class"))
```

```
encues.tr0 <- tree(formula =
merge.levels(encuesta$QUE.SEMB.FRIJOL, list(c("h", "m", "x")))
    ~ SEXO.ENT + ENC.FAM + MADRE.SO.. + EDAD + ANO.EST +
    TENENCIA, data = encuesta, na.action = na.exclude, mincut
= 4, minsize = 8,
    mindev = 0.005)

table(merge.levels(encuesta$QUE.SEMB.FRIJOL, list(c("h", "m",
"x"))),predict(encues.tr0,encuesta,type="class"))

##
## With age classes
    ## Frijol
encuesta$EDADF <- cut(encuesta$EDAD, breaks=c(0,20,40,60,100),
labels=c("<20", "20-40","40-60", ">60"))
encues.tr0 <- tree(formula =
merge.levels(encuesta$QUE.SEMB.FRIJOL, list(c("h", "m", "x")))
    ~ SEXO.ENT + ENC.FAM + MADRE.SO.. + EDADF + ANO.EST +
    TENENCIA, data = encuesta, na.action = na.exclude, mincut
= 4, minsize = 8,
    mindev = 0.005)
table(merge.levels(encuesta$QUE.SEMB.FRIJOL, list(c("h", "m",
"x"))),predict(encues.tr0,encuesta,type="class"))

    ## Maiz
encuesta$EDADF <- cut(encuesta$EDAD, breaks=c(0,20,40,60,100),
labels=c("<20", "20-40","40-60", ">60"))
encues.tr0 <- tree(formula =
merge.levels(encuesta$QUE.SEMB.MAIZ, list(c("h", "m", "x")))
    ~ SEXO.ENT + ENC.FAM + MADRE.SO.. + EDADF + ANO.EST +
    TENENCIA, data = encuesta, na.action = na.exclude, mincut
= 5, minsize = 10,
    mindev = 0.005)

table(merge.levels(encuesta$QUE.SEMB.MAIZ, list(c("h", "m",
"x"))),predict(encues.tr0,encuesta,type="class"))



# with AREA instead of TENANCIA only
encues.tr0 <- tree(formula =
merge.levels(encuesta$QUE.SEMB.FRIJOL, list(c("h", "m", "x")))
~ SEXO.ENT + ENC.FAM + MADRE.SO.. + EDAD + ANO.EST +
    AREA, data = encuesta, na.action = na.exclude, mincut = 4,
minsize = 8,
    mindev = 0.005)

table(merge.levels(encuesta$QUE.SEMB.FRIJOL, list(c("h", "m",
"x"))),predict(encues.tr0,encuesta,type="class"))
```

```
###
###              2. Cross validation for optimal tree selection
###


##   initialization
tr.cv <- cv.tree(encues.tr0, , prune.misclass)

# Repeated computation and averaging because of variability of
cross-validation
#

for     (i     in     2:10)     tr.cv$dev<-     tr.cv$dev     +
cv.tree(encues.tr0,,prune.misclass)$dev

tr.cv$dev <- tr.cv$dev/10

plot(tr.cv)    # visualization of the cross-validation graph


###
###              3. Final pruning
###

# The optimal tree is selected to be the one with the smallest
size with the best
# misclassification error.


OptimalSize                                                    <-
min(tr.cv$size[match(min(tr.cv$dev),tr.cv$dev)])

tr.pr <- prune.tree(encues.tr0, best=OptimalSize)
table(merge.levels(encuesta$QUE.SEMB.FRIJOL,  list(c("h",  "m",
"x"))),
               predict(tr.pr,encuesta,type="class"))



###
###              Miscellaneaous
###

## ------ To get a screen plot of the tree -----

tabPlot.tree(tr.pr)

## ------ To build a PostScript file of the tree -----

post.tree(tr.pr, title = "PobrezaHON-Sembrado de Frijol", file
= "frijol.ps")
```

```
ciat.post.tree(tr.pr, title = "BID-Pobreza-Honduras\n Sembrado
de Maiz",
          file = "maiz.es.ps", language="spanish")
ciat.post.tree(tr.pr,        title        =        "BID-Pobreza-
Honduras\nPlantation of Maiz",
          file = "maiz.en.ps", language="english")
```