Structure of the trees objects in R

Ambra Macis

In this document, an example of how extracting information from the three examined tree algorithms is provided.

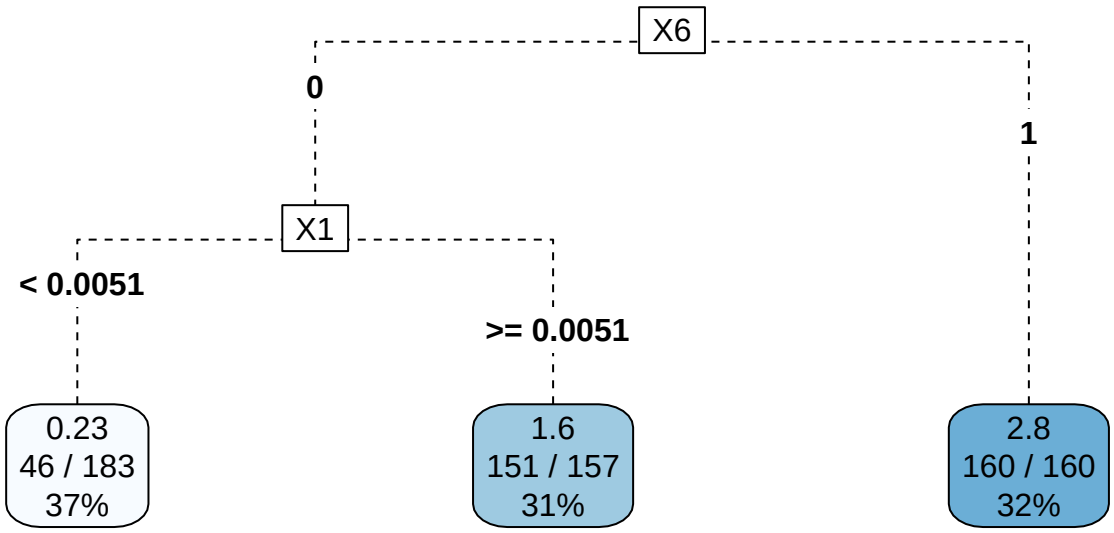
Data Simulation

```
n_rv <- 10                      # Number of covariates
prob_rvar <- seq(0.3,0.7,0.1) # Parameters for the Bernoulli r.v.
N <- 1000                       # Sample size

### Generate Standard Normal independent r.v.
library(mvtnorm)
set.seed(70522)
Norm_RV <- rmvnorm(n=N, mean=rep(0,n_rv/2), sigma=diag(n_rv/2))
### Generate Bernoulli independent r.v.
library(MultiRNG)
set.seed(70522)
Ber_RV <-draw.correlated.binary(no.row=N, d=n_rv/2, prop.vec=prob_rvar,
                                corr.mat=diag(n_rv/2))

cov <- as.data.frame(cbind(Norm_RV, Ber_RV))
colnames(cov) <- paste0("X", 1:n_rv)

### Definition of the partitions
node1 <- cov[,"X6"]==1
node2 <- cov[,"X6"]==0 & cov[,"X1"]>=0
node3 <- cov[,"X6"]==0 & cov[,"X1"]<0

maxtime <- 3                    #Length of the follow-up

simdata <- as.data.frame(matrix(data=NA, nrow=N, ncol=2))
colnames(simdata)<- c("eventtime","status")
library(simsurv)
# Node1
simdata[node1,1:2] <- simsurv(dist="exponential", lambda=2, maxt=maxtime,
                              x=as.data.frame(1:sum(node1)), seed=7052022)[,2:3]
# Node2
simdata[node2,1:2] <- simsurv(dist="exponential", lambda=0.9, maxt=maxtime,
                              x=as.data.frame(1:sum(node2)), seed=7052022)[,2:3]
# Node3
simdata[node3,1:2] <- simsurv(dist="exponential", lambda=0.1, maxt=maxtime,
                              x=as.data.frame(1:sum(node3)), seed=7052022)[,2:3]

data <- cbind(simdata,cov) #Dataset with survival data and covariates
data[, 8:12] <- lapply(data[, 8:12], as.factor)
```

```
### Training and testing sets
set.seed(70522)
# To take a similar percentage of events in the training and test sets:
library(caret)
trainset <- createDataPartition(y=data$status, p=0.5, list=F)
data_train <- data[trainset,]
data_test <- data[-trainset,]
```

Relative Risk Trees

```
library(survival)
library(rpart)
RRT <- rpart(formula=Surv(eventtime, status)~., data=data_train,
             control=rpart.control(usesurrogate=2, minsplit=20, xval=10))
# Pruning RRT
minerr <- which.min(RRT$cptable[,"xerror"])
bestcp <- RRT$cptable[minerr, "CP"]
pruned_RRT <- prune(tree=RRT, cp=bestcp)
# Plot RRT
library(rpart.plot)
rpart.plot(x= pruned_RRT, type= 5, tweak=1.0, gap=0.02, branch.lty=2)
```



The main object representing the structure of the Relative Risk Tree (RRT) is the *frame* data frame. Each row represents one node. The columns include:

- *"var"*, a factor with the name of the splitting variable (or if the node is terminal it returns "<leaf>");
- *"n"*, the number of observations in the node;
- *"yval"*, the outcome at the node.

Further information are also available in the function's help (see *?rpart.object*).

```
pruned_RRT$frame
```

```
##      var   n  wt      dev      yval  complexity ncompete nsurrogate     yval2.1
## 1     X6 500 500 780.1231 1.0000000 0.218058734        4          5   1.0000000
## 2     X1 340 340 463.9415 0.6559016 0.218058734        4          5   0.6559016
## 4 <leaf> 183 183 179.6887 0.2260154 0.009346277        0          0   0.2260154
## 5 <leaf> 157 157 112.3782 1.6012796 0.004791221        0          0   1.6012796
## 3 <leaf> 160 160 147.8310 2.8183587 0.008424651        0          0   2.8183587
##       yval2.2
## 1 357.0000000
## 2 197.0000000
## 4  46.0000000
## 5 151.0000000
## 3 160.0000000
```

Starting from this dataframe, the size of the tree (number of overall nodes, terminal nodes and splits) can be obtained as follows:

```
nodes_RRT <- nrow(pruned_RRT$frame)                    # Number of overall nodes
```

```
## [1] 5
```

```
term_nodes_RRT <- sum(pruned_RRT$frame$var=="<leaf>") # Number of terminal nodes
```

```
## [1] 3
```

```
nsplit_RRT <- nodes_RRT - term_nodes_RRT               # Number of splits
```

```
## [1] 2
```

As can be seen also by the above figure, RRT has 5 nodes, of which 3 are terminal nodes, determined by 2 splits.

The splitting variables can be determined through the *frame* matrix, excluding the rows of terminal nodes.

```
split_vars_RRT <- pruned_RRT$frame[-which(pruned_RRT$frame[,1]=="<leaf>"),"var"]
```

```
## [1] "X6" "X1"
```

The splitting variables are $X6$ and $X1$. The cut-off values used for the splits are recorded in the *splits* matrix of the *rpart* object. The row label is the name of the split variable. Columns include:

- *"count"*, the number of observations sent left or right by the split;
- *"ncat"*, the number of categories/levels for the variable (+/-1 for a continuous variable; the sign determines whether the subset $x < cutpoint$ or $x > cutpoint$ is sent to the left);
- *"improve"*, the improvement in deviance given by the split;
- *"index"*, the numeric split point if the splitting variable is continuous. If the splitting variable is a factor it reports the row number of the *csplit* matrix.
- *"adj"*, the adjusted concordance for surrogate splits.

```
pruned_RRT$splits
```

```
##     count ncat     improve        index         adj
## X6    500    2 168.3721078  1.000000000 0.000000000
## X1    500   -1 143.4749318  0.002519567 0.000000000
## X2    500    1   7.5533962  1.585330148 0.000000000
## X5    500   -1   6.3595358  1.989192954 0.000000000
## X4    500    1   5.2467345  1.774409986 0.000000000
## X3      0    1   0.6920000 -2.088180969 0.037500000
## X1      0   -1   0.6880000  2.322958899 0.025000000
## X2      0    1   0.6860000 -2.647884604 0.018750000
## X5      0   -1   0.6840000  2.563222676 0.012500000
## X4      0    1   0.6820000 -2.250619612 0.006250000
## X1    340   -1 171.8893227  0.005131272 0.000000000
## X5    340   -1   5.9499559  1.767275761 0.000000000
## X2    340    1   4.8396352  1.577770805 0.000000000
## X4    340   -1   4.0196610 -1.708539523 0.000000000
## X3    340    1   3.8843340  0.660761765 0.000000000
## X5      0    1   0.5647059 -0.480078346 0.057324841
## X3      0    1   0.5588235 -0.992950301 0.044585987
## X10     0    2   0.5500000  2.000000000 0.025477707
## X2      0    1   0.5441176 -1.837232024 0.012738854
## X4      0    1   0.5411765 -0.955194476 0.006369427
```

For categorical variables the *csplit* matrix, together with the *splits* matrix, has to be examined. It includes one row for each categorical splitting variable, while the number of columns corresponds to the largest number of levels in the factors. The row to examine is given by the *index* column of the *splits* matrix. The columns record 1 if that level of the factor goes to the left, 3 if it goes to the right and 2 if that level is not present at that node of the tree.

```
pruned_RRT$csplit
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    3    1
```

For example, for $X6$ the *index* value in the *splits* matrix is equal to 1, so the first row of *csplit* must be considered. The first level of the factor (0) goes to the left, while the second level (1) goes to the right.

To extract the splitpoint (the row of *csplit*) for the continuous (categorical) splitting variable the following procedure can be used. The elements to take into account are the number of observations in the node and the splitting variable used. If the same splitting variable is repeated more than once in the matrix the first row is considered (the one with the highest improvement).

```
#### Cut-off values
splits_RRT <- pruned_RRT$splits
frame_RRT <- pruned_RRT$frame
cutoff_RRT <- rep(NA,length(split_vars_RRT))
ncat_RRT <- rep(NA,length(split_vars_RRT))

for(j in 1:length(split_vars_RRT))
  {
    temp <- which(splits_RRT[,1]==frame_RRT[j,"n"] &
                  rownames(splits_RRT)==split_vars_RRT[j])
    #If the same variable is repeated more times the first row is considered
    cutoff_RRT[j] <- splits_RRT[temp[1],"index"]
    ncat_RRT[j] <- splits_RRT[temp[1],"ncat"]
  }
```

```
split_vars_RRT
```

```
## [1] "X6" "X1"
```

```
cutoff_RRT
```

```
## [1] 1.000000000 0.005131272
```

```
pruned_RRT$csplit[cutoff_RRT[1],]
```

```
## [1] 1 3
```

```
ncat_RRT
```

```
## [1]  2 -1
```

The first splitting variable is $X6$ that is a factor, and we already know that the cut-off is 0/1. Looking at the first row of *csplit* we can see that the observations with $X6 = 0$ go to the left node. The cut-off for the continuous outcome $X1$ is 0.0051. Since the corresponding *ncat* is equal to -1 the observations with values of $X1 < 0.0051$ go to the left.

To extract the outcomes of the nodes, i.e. the relative risk at each node, the following R code can be used:

```
outcome_nodes_RRT <- frame_RRT[which(frame_RRT[,1]=="<leaf>"),"yval"]
```

```
## [1] 0.2260154 1.6012796 2.8183587
```

The outcomes are in order from the left leaf to the right one. They are respectively equal to 0.2260154, 1.6012796 and 2.8183587.

Finally, the paths of the tree can be identified as follows:

```
paths_RRT <- rpart.plot:::rpart.rules(pruned_RRT,roundint=FALSE)
```

```
##  Surv
##  0.23 when X6 is 0 & X1 <  0.0051
##  1.60 when X6 is 0 & X1 >= 0.0051
##  2.82 when X6 is 1
```

The resulting output is a dataframe. To extract the relative information (e.g. to understand which node is split) the following procedure can be used:

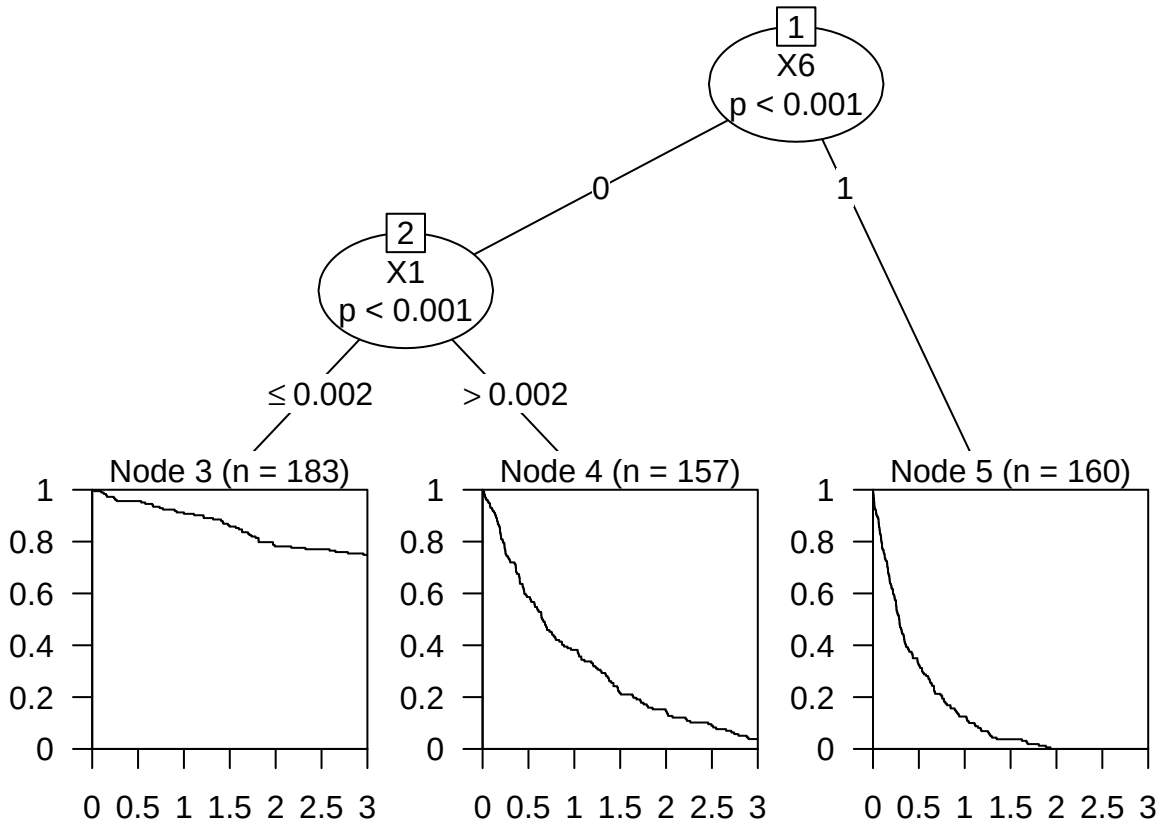
```
node_split <- matrix(NA, nrow=2, ncol=term_nodes_RRT)
for(i in 1:term_nodes_RRT)
  {
    if(paths_RRT[i,6]=="&")
      {
        node_split[1,i] <-  paths_RRT[i,c(3)]
        node_split[2,i] <- if(paths_RRT[i,6]=="&") paths_RRT[i,c(5)]
      }
  }
```

```
##      [,1] [,2] [,3]
## [1,] "X6" "X6" NA
## [2,] "0"  "0"  NA
```

The node that is split is therefore the one corresponding to $X6 = 0$.

Conditional Inference Trees with the *ctree* algorithm

```
library(partykit)
CIT <- ctree(formula= Surv(eventtime, status)~., data= data_train,
             control= ctree_control(minsplit=20))

# Plot CIT
plot(CIT)
```



The size of Conditional Inference Trees obtained with the *ctree* algorithm (CIT) can be obtained as follows:

```
nodes_CIT <- length(CIT)                    # Number of overall nodes
```

```
## [1] 5
```

```
term_nodes_CIT <- width(CIT)                # Number of terminal nodes
```

```
## [1] 3
```

```
n_splits_CIT <- nodes_CIT - term_nodes_CIT  # Number of splits
```

```
## [1] 2
```

The fitted CIT (as can be observed also through the plot) has 5 overall nodes; 3 of them are terminal nodes, determined by 2 splits.

To extract the splitting variables the structural change test (statistic test for building the tree) can be examined. It can be obtained through the *sctest* function in the *strucchange* package.

```r
library(strucchange)
SCTest <- sctest(CIT)
```

```
## $`1`
##                      X1        X2        X3         X4        X5           X6
## statistic 1.009557e+02 3.1213762 1.2541127 0.03599432 0.3448064 1.391055e+02
## p.value   9.406198e-23 0.5525553 0.9525715 0.99999999 0.9997093 4.176692e-31
##                   X7          X8       X9       X10
## statistic 0.2245846 0.001217413 2.635009 0.5981145
## p.value   0.9999587 1.000000000 0.668485 0.9969287
##
## $`2`
##                      X1       X2       X3        X4        X5         X7
## statistic 1.154079e+02 0.316541 1.603844 0.5965181 0.1103783 0.02653715
## p.value   5.766036e-26 0.999535 0.873661 0.9945760 0.9999945 0.99999999
##                   X8        X9       X10
## statistic 0.1076238 0.4704989 0.3152374
## p.value   0.9999951 0.9977770 0.9995427
##
## $`3`
##                    X1        X2        X3        X4        X5          X7
## statistic 0.02309543 0.0303257 0.5426397 0.9627379 0.7927230 0.001994507
## p.value   0.99999999 1.0000000 0.9961821 0.9714865 0.9850832 1.000000000
##                   X8        X9        X10
## statistic 0.2969120 0.3206509 6.50066478
## p.value   0.9996414 0.9995101 0.09296824
##
## $`4`
##                   X1        X2        X3        X4        X5          X7
## statistic 0.9939474 2.2488608 0.8112503 0.9552758 1.6853453 0.007226842
## p.value   0.9684071 0.7252378 0.9838566 0.9721957 0.8567914 1.000000000
##                    X8         X9       X10
## statistic 0.04947138 0.02608566 0.1722870
## p.value   0.99999984 0.99999999 0.9999629
##
## $`5`
##                   X1        X2          X3        X4         X5        X7
## statistic 0.0310211 0.8888394 0.003814942 0.8308227 0.02630194 1.3386237
## p.value   1.0000000 0.9780496 1.000000000 0.9824946 0.99999999 0.9224257
##                    X8       X9       X10
## statistic 0.03470195 5.357168 0.1375877
## p.value   0.99999997 0.171119 0.9999858
```

```r
# Number of elements in the list SCTest, i.e. number of nodes of the tree
l_SCTest <- length(SCTest)
```

```
## [1] 5
```

```r
# When the list is composed of only one element (tree with only 1 node),
# l_SCTest is equal to the length of the matrix (2 rows and 1 column for each
# variable) If this occurs l_SCTest should be fixed equal to 1.

if(l_SCTest==2*n_rv)  l_SCTest <- 1
```

The idea is to extract, for each node, the variable that is strongly associated to the outcome (in a significant

way, i.e. $p \leq \alpha = 0.05$). Indeed, this is the way how the algorithm selects the splitting variables.

```
alpha <- 0.05

split_vars_CIT <- matrix(NA,nrow=1,ncol=l_SCTest)
for(j in 1:l_SCTest)
  {
    if(is.null(SCTest[[j]])) split_vars_CIT[1,j] <- "NULL"
    else
      {
        split_vars_CIT[1,j] <- min(SCTest[[j]]["p.value",]) #takes the minimum p
        if(as.numeric(split_vars_CIT[1,j])< alpha) # check significance
          split_vars_CIT[1,j] <- names(which.min(SCTest[[j]][2,]))
        else
          split_vars_CIT[1,j] <- "No sig"
      }
  }

split_vars_CIT
```

```
##      [,1] [,2] [,3]     [,4]     [,5]
## [1,] "X6" "X1" "No sig" "No sig" "No sig"
```

So, it can be seen, that the splitting variables used to grow the tree are $X6$ and $X1$.

For looking at the splitpoints the paths of the CIT can be considered. Then, treating it as a character we can split the string and identify the cut-off:

```
path_CIT <- partykit:::.list.rules.party((CIT))
```

```
##                                                 3
## "X6 %in% c(\"0\") & X1 <= 0.00205327735949113"
##                                                 4
##  "X6 %in% c(\"0\") & X1 > 0.00205327735949113"
##                                                 5
##                               "X6 %in% c(\"1\")"
```

```
cutoff_CIT <- rep(NA,n_splits_CIT)

if (is.factor(data_train[,split_vars_CIT[1]]))
  {
    temp <- strsplit(path_CIT[1],"")
    cutoff_CIT[1] <- temp[[1]][12]
}

if (is.numeric(data_train[,split_vars_CIT[2]]))
  {
    temp <- strsplit(path_CIT[1],"=")
    cutoff_CIT[2] <- as.numeric(sapply(temp, `[`, 2))
  }
```

```
## [1] "0"                   "0.00205327735949113"
```

For extracting the outcome of CIT, i.e. the median survival time, the predict function can be used:

```
unique(predict(CIT,type="response"))
```

```
## [1] 0.2880490 0.6534732       Inf
```

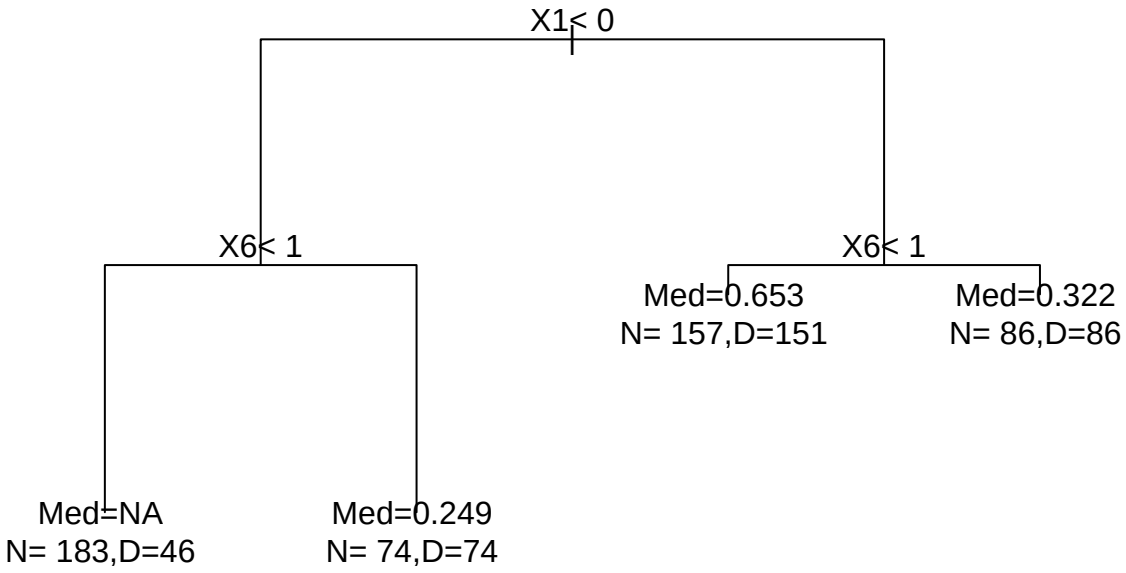
The estimated median survival times for the first two nodes are 0.2880490 and 0.6534732. The last estimated median survival time is set equal to Inf because the length of the follow-up is shorter than the median survival time (less than 50% of the entire sample experiences the event at the end of follow-up).

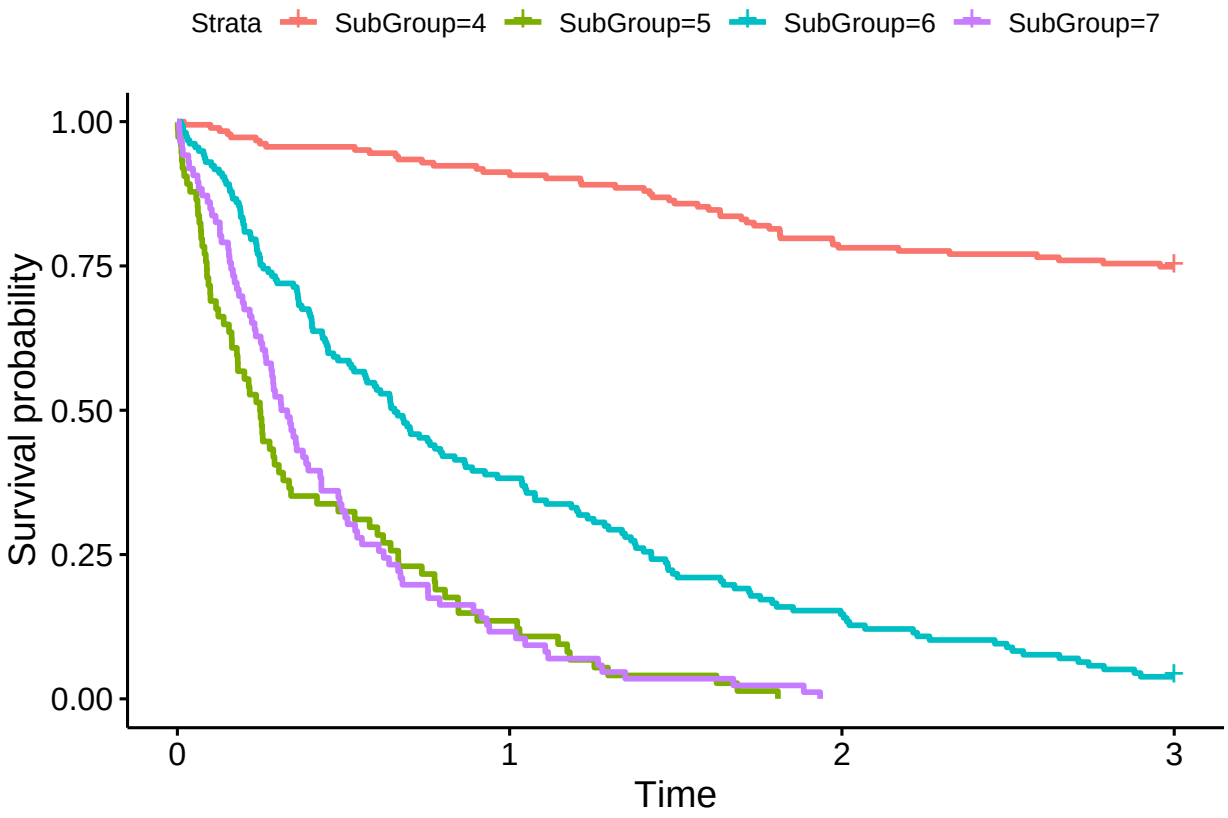
Conditional Inference Trees with the *SurvCART* algorithm

```
# Data pre-processing
data_train_SCT <- data_train
id <- 1:nrow(data_train_SCT)
data_train_SCT <- cbind(data_train_SCT, id)
data_train_SCT[, paste0("X",6:10)] <-
  lapply(data_train_SCT[, paste0("X",6:10)], as.numeric)
data_train_SCT[, paste0("X",6:10)] <- data_train_SCT[, paste0("X",6:10)] -1

# Model Fitting
library(LongCART)
SCT <- SurvCART(data=data_train_SCT, patid="id",timevar="eventtime",
                censorvar="status", gvars=paste0("X",1:10),
                tgvars=c(1,1,1,1,1,0,0,0,0,0), time.dist="exponential",
                cens.dist="NA", minsplit=20)
```

```
##   ID   n   D median.T median.C loglik    AIC var index p (Instability) improve
## 1  1 500 357    0.928        3 -592.9 1187.8  X1     0           0.000   117.9
## 2  2 257 120       NA        3 -291.8  585.7  X6     1           0.000   129.1
## 3  4 183  46       NA        3 -153.0  308.1 X10    NA           0.068      NA
## 4  5  74  74    0.249       NA   -9.7   21.5  X2    NA           0.073      NA
## 5  3 243 237    0.488        3 -183.2  368.4  X6     1           0.000    15.3
## 6  6 157 151    0.653        3 -149.8  301.6  X1    NA           0.930      NA
## 7  7  86  86    0.322       NA  -18.1   38.3  X9    NA           0.837      NA
##   Terminal
## 1    FALSE
## 2    FALSE
## 3     TRUE
## 4     TRUE
## 5    FALSE
## 6     TRUE
## 7     TRUE
```

```
# Plot SCT
par(xpd = TRUE)
plot(SCT, compress = TRUE)
text(SCT, use.n = TRUE)

# Plot Kaplan-Meier curves estimated in the terminal nodes
KMPlot.SurvCART(x= SCT, type= 1)
```


X1< 0
X6< 1
X6< 1
Med=0.653
N= 157,D=151
Med=0.322
N= 86,D=86
Med=NA
N= 183,D=46
Med=0.249
N= 74,D=74



All the information about the structure of Conditional Inference Trees obtained with the *SurvCART* algorithm (SCT) are included in the *Treeout* dataframe. The *Treeout* matrix includes one row for each node. The columns include:

- *"ID"*, the identity number of the node;
- *"n"*, the number of observations in the node;
- *"D"*, the number of events in the node;
- *"median.T"*, the estimated median survival time in the node;
- *"median.C"*, the estimated median censoring time in the node;
- *"var"*, the splitting variable used for growing the tree;
- *"index"*, the cut-off that has determined the split;
- *"Terminal"*, the indicator of terminal node.

Further information about the structure of the dataframe *Treeout* can be found in the help function (?SurvCART).

```
SCT$Treeout
```

```
##   ID   n   D median.T median.C loglik    AIC var index p (Instability) improve
## 1  1 500 357    0.928        3 -592.9 1187.8  X1     0           0.000   117.9
## 2  2 257 120       NA        3 -291.8  585.7  X6     1           0.000   129.1
## 3  4 183  46       NA        3 -153.0  308.1 X10    NA           0.068      NA
## 4  5  74  74    0.249       NA   -9.7   21.5  X2    NA           0.073      NA
## 5  3 243 237    0.488        3 -183.2  368.4  X6     1           0.000    15.3
## 6  6 157 151    0.653        3 -149.8  301.6  X1    NA           0.930      NA
## 7  7  86  86    0.322       NA  -18.1   38.3  X9    NA           0.837      NA
##   Terminal
## 1    FALSE
```

```
## 2    FALSE
## 3     TRUE
## 4     TRUE
## 5    FALSE
## 6     TRUE
## 7     TRUE
```

The size of the tree can be obtained with the procedure below:

```
nodes_SCT <- nrow(SCT$Treeout)                              # Number of overall nodes
```

```
## [1] 7
```

```
term_nodes_SCT <- sum(SCT$Treeout[,"Terminal"]=="TRUE") # Number of terminal nodes
```

```
## [1] 4
```

```
nsplits_SCT <- nodes_SCT - term_nodes_SCT                # Number of splits
```

```
## [1] 3
```

The tree has 7 nodes, of which 4 are leaves. The number of splits needed to grow the tree is equal to 3.

The splitting variables and the cut-offs can be determined as follows:

```
vars_SCT <- matrix(NA,nrow=1,ncol=nsplits_SCT)
splits_SCT <- matrix(NA,nrow=1,ncol=nsplits_SCT)
for(j in 1:nsplits_SCT)
  {
   vars_SCT[1,j] <-
                 SCT$Treeout[which(SCT$Treeout[,"Terminal"]=="FALSE"),"var"][j]
   splits_SCT[1,j] <-
                SCT$Treeout[which(SCT$Treeout[,"Terminal"]=="FALSE"),"index"][j]
  }

vars_SCT
```

```
##      [,1] [,2] [,3]
## [1,] "X1" "X6" "X6"
```

```
splits_SCT
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    1
```

So, the splitting variables are $X1$, $X6$ and $X6$. Ths cut-off values for the continuous variable $X1$ is 0.

The outcome, i.e. the median survival time, at each node can be retrieved as follows:

```
outcome_SCT <- matrix(NA,nrow=1,ncol=nsplits_SCT)
for(j in 1:nsplits_SCT)
  {
    outcome_SCT[1,j] <-
              SCT$Treeout[which(SCT$Treeout[,"Terminal"]=="TRUE"),"median.T"][j]
  }

outcome_SCT
```

```
##      [,1]  [,2]  [,3]
## [1,]   NA 0.249 0.653
```

The first node has a median survival time equal to NA because the follow-up duration is shorter than the median survival time. The other estimated median survival times are 0.249 and 0.653.