1.1 Introduction

A decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning. It is a non-parametric supervised learning method used for both classification and regression tasks. They are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. The structure of a decision tree includes a root node, branches, and leaf nodes. Each internal node de- notes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node. A decision tree is drawn upside down with its root at the top. The feature importance and relations can be viewed clearly.

So, what is actually going on in the background? Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, you will need to trim it down for it to be not over fitted. While making decision tree, at each node of tree we ask different type of questions. Based on the question asked, we will calculate the information gain corresponding to it. The details are presented in Chapter 2.

Decision tree methodology is more commonly known as learning decision tree from data. The algorithms that come under decision tree may further be classified as *classification trees* and *regression trees*. A decision tree with numeric output is called regression tree. By a classification tree, it is understood that the output variable is categorical. Classification trees are very popular because they are used in a wide variety areas, they generate rules that are easy to interpret, and ease in their use. Classification trees are powerful because they can handle a variety of data types, scalable, can handle missing values, and, when used in ensembles of trees, they provide excellent accuracy. Classification trees have been successfully used for classification in diverse areas such as radar signal classification, character recognition, remote sensing, medical diagnosis, and expert systems and speech recognition. They require little data preparation in terms of parameter settings, and are well suited to exploratory knowledge discovery. They can handle both numerical and categorical data. It is possible to validate a model classification tree with statistical tests and for a human expert to interpret it. Consequently, the reliability of the model can be expressed and accounted for. Classification trees can handle datasets that may have errors or missing values. The model is typically robust and can handle large amounts of data in a relatively short time. The most important characteristics of decision tree classifiers are their efficacy to divide a complex decision-making process into a number of simpler decisions, and thus provide a solution which is easily interpretable.

Ross Quinlan, a machine learning researcher, developed a decision tree algorithm known as ID3 (Iterative Dichotomiser) in 1980. Later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach.

There are quite a number situations where one has huge amounts of data using which he has to make some decisions. For example, in the retail scenario, the analyst may be interested in predicting a *prospective buyer*

of some product, in the case of bank example, basing on the past customer data may have to decide whether *to offer* a bank card or *do not offer*. In the medical scenario, one may be interested in developing a medical system to diagnose a patient whether is *having* or *not having* a particular disease, based on several symptoms and medical tests conducted. A financial consulting firm would like to predict the trend of the price of a stock which may be classified into *upward*, *downward* or *no trend* based on several technical features that govern the price movement. A scientist analyzing the gene expression data would like to identify the most relevant genes and risk factors involved in breast cancer, in order to separate healthy patients from breast cancer patients and so on. All these situation demand a correct decision to be taken. Making the right decision is the key factor for successful achievement in all areas of work. A right decision may be arrived in a number of ways. However, all these solutions may have the same underlying basic idea of arriving at the correct decision. The final decision is a combination of past experiences from solving similar cases, the results of recent research findings and personal judgement. The number of solved cases and new research findings are increasing rapidly. Quite naturally, one could expect that newly made decisions will become better and more reliable. But, for the individuals and groups who have to make decisions, it is actually becoming more and more complicated, because they simply cannot process the huge amounts of data any more. And there the need for a good decision supports to make their decisions easier and more reliable. For this purpose, it is equally or even more important as suggesting the possible decisions, to provide also an explanation of how and why the suggested decision was chosen. In this manner an expert can decide whether the suggested solution is appropriate or not. In all the above situations, people are involved in the process of discovering insightful, interesting, and novel patterns, as well as developing descriptive, understandable, and predictive models from large-scale data.

Ensemble learning is an effective technique that has increasingly been adopted to com- bine multiple learning algorithms to improve overall prediction accuracy. These ensemble techniques have the advantage to alleviate the small sample size problem by averaging and incorporating over multiple classification models to reduce the potential for overfitting the training data. In this way the training data set may be used in a more efficient way, which is critical to many biological applications with small sample size. Some ensemble methods such as random forests are particularly useful for high-dimensional datasets because increased classification accuracy can be achieved by generating multiple prediction models each with a different feature subset.

1.2 Objectives

- To learn R package from programming point of view.

- To learn a machine learning technique called Decision tree that has simple interpretation.

- To have an understanding of basic philosophy of techniques that enhance the performance of decision tree models called the ensemble methods and their usage; in particular random forests and boosting.

- To perform statistical analysis on a dataset using the above techniques.

1.3 Data Source

The data set used to illustrate the techniques is downloaded from UCI machine learning repository. The location of the data set is:

https://archive.ics.uci.edu/ml/machine-learning-databases/cmc/cmc.data

1.4 Coming Up

In Chapter-1, we discuss the classification problem using a decision tree in general, and its application in the various practical situations. Algorithms relating to simple classification and regression trees are described and their algorithms are stated. We use the contraceptive method choice data set. This data set is download from the machine learning data repository of UCI. Objectives of project work are stated. A brief description of each Chapter in the project work is also given towards the end of this Chapter.

Chapter-2, Explore the various statistical features of the contraceptive method choice data set. By understanding the data we get to know about their association. Other than using the methods like box plot we here use commands like **`plotcirc`**() function and **`DescTools`** package to visualizing the data.

Chapter-3, describes the classical Decision tree. The general procedure underlying the classification and regression trees explained. Various information theory concepts such as entropy, information gain, and information ratio, which are required in performing the split rule at each node, are explained. By the **`rpart`** and **`rpart.plot`** packages of R programming can be used to perform a decision tree analysis. We also discuss various algorithms for implementing decision trees together their performance evaluation.

Chapter-4, simple decision tree construction and its evaluation on the contraceptive method choice dataset is the content of Chapter-3. Here, classification trees based on the functions implemented in the R package C50 are used. After building classification trees based on the training datasets, the performance of the models are also obtained through various evaluation methods.

Summary on the need and working of the classification and regression trees together with the results of classification algorithms on the contraceptive method choice dataset are presented in Chapter-5

2.1 Exploring Data

2.1.1 Preparing Data Frame

The file was saved as text file called cmc-data.txt, after opening it from its original source with the browser.

Read the data into R using the `read.csv()` function.

```
> cmc <- read.csv(file="cmc-data.txt", sep = ",",header=FALSE)
```

Examine the data using the `head()` fucntion.

```
> head(cmc)
  V1  V2 V3 V4  V5 V6 V7 V8  V9 V10
1 24   2  3  3   1  1  2  3   0   1
2 45   1  3 10   1  1  3  4   0   1
3 43   2  3  7   1  1  3  4   0   1
4 42   3  2  9   1  1  3  3   0   1
5 36   3  3  8   1  1  3  2   0   1
6 19   4  4  0   1  1  3  3   0   1
```

The description of the variable were presented in a separate text file at the original source in file called cmc-names.txt. Attribute Information from the file reads as follows:

```
1. Wife's age                      (numerical)
2. Wife's education                (categorical)      1=low, 2, 3, 4=high
3. Husband's education             (categorical)      1=low, 2, 3, 4=high
4. Number of children ever born    (numerical)
5. Wife's religion                 (binary)           0=Non-Islam, 1=Islam
6. Wife's now working?             (binary)           0=Yes, 1=No
7. Husband's occupation            (categorical)      1, 2, 3, 4
8. Standard-of-living index        (categorical)      1=low, 2, 3, 4=high
9. Media exposure                  (binary)           0=Good, 1=Not good
10. Contraceptive method used      (class attribute)  1=No-use
                                                      2=Long-term
                                                      3=Short-term
```

The dataset contains 1473 cases without any missing values. The structure of the data set can be visualized using the `str()` function as follows:

```
 > str(cmc)
'data.frame':    1473 obs. of  10 variables:
  $ V1 : int 24 45 43 42 36 19 38 21 27 45 ...
  $ V2 : int 2 1 2 3 3 4 2 3 2 1 ...
  $ V3 : int 3 3 3 2 3 4 3 3 3 1 ...
  $ V4 : int 3 10 7 9 8 0 6 1 3 8 ...
  $ V5 : int 1 1 1 1 1 1 1 1 1 1 ...
  $ V6 : int 1 1 1 1 1 1 1 0 1 1 ...
  $ V7 : int 2 3 3 3 3 3 3 3 3 2 ...
  $ V8 : int 3 4 4 3 2 3 2 2 4 2 ...
  $ V9 : int 0 0 0 0 0 0 0 0 0 1 ...
  $ V10: int 1 1 1 1 1 1 1 1 1 1 ...
```

From the above we note that all the attributes are read as integer type. We need to convert them to their respective types as given under attribute information. Firstly, we rename the attributes.

```
colnames(cmc)  <-  c("wife.age","wife.edu","husb.edu","num.child","islam",
"wife.working", "husb.job","SOL.index", "media.exposure", "cmu")
```

Verify whether the names of the variables were properly set:

```
 > str(cmc)
 'data.frame':   1473 obs. of  10 variables:
 $ wife.age      : int 24 45 43 42 36 19 38 21 27 45 ...
 $ wife.edu      : int 2 1 2 3 3 4 2 3 2 1 ...
 $ husb.edu      : int 3 3 3 2 3 4 3 3 3 1 ...
 $ num.child     : int 3 10 7 9 8 0 6 1 3 8 ...
 $ islam         : int 1 1 1 1 1 1 1 1 1 1 ...
 $ wife.working  : int 1 1 1 1 1 1 1 0 1 1 ...
 $ husb.job      : int 2 3 3 3 3 3 3 3 3 2 ...
 $ SOL.index     : int 3 4 4 3 2 3 2 2 4 2 ...
 $ media.exposure: int 0 0 0 0 0 0 0 0 0 1 ...
 $ cmu           : int 1 1 1 1 1 1 1 1 1 1 ...
```

Three of the categorical variables namely, Wife's Education, Husband's Education and Standard-of-living Index are each having 4 categories with designated values as `1=low, 2,3 4=high`. The middle two levels are not given any specific names. We designate these levels as `1=low, 2=mid-low,3=mid-high, 4=high`.

We now convert the education variables, for both wife and husband, as well as Standard- of-living variables into factors:

```
>cmc$wife.edu <- factor(cmc$wife.edu,labels = c("low", "mid-low",
                        "mid-high","high"),
                    ordered = TRUE)

>cmc$husb.edu <- factor(cmc$husb.edu,labels = c("low", "mid-low",
                          "mid-high","high"),
                    ordered = TRUE)

>cmc$SOL.index <- factor(cmc$SOL.index,labels = c("low", "mid-low",
                    "mid-high", "high"), ordered =
                    TRUE)
```

The response variable should also be converted into a categorical variable having three levels whose labels being ”no-use”, ”long-term”, ”short-term”.

```
cmc$cmu <- factor(cmc$cmu,labels = c("no-use","short-term","long-term" ))
```

There are three binary variables in the dataset namely, Wife’s religion, Wife’s now working? And Media exposure. Convert these variables into Boolean variables.

```
 > cmc$islam <- ifelse(cmc$islam == 1, TRUE, FALSE)

 > cmc$wife.working <- ifelse(cmc$wife.working == 1, FALSE, TRUE)

 > cmc$media.exposure <- ifelse(cmc$media.exposure == 1, FALSE, TRUE)
```

The data description, it was given that the husband’s occupation was a categorical variable. The values are given as 1, 2, 3 and 4 and no further information is provided. So, it we can only convert this variable as categorical, without specific names attached to the levels.

```
 > cmc$husb.job <- as.factor(cmc$husb.job)
```

We have done all the necessary transformations to the dataset. Let us have a final look at the structure of the data:

```
 > str(cmc)
'data.frame':    1473 obs. of  10 variables:
$ wife.age        : int  24 45 43 42 36 19 38 21 27 45 ...
$ wife.edu        : Factor w/ 4 levels "low","mid-low",..: 2 1 2 3 3 4 2 3 2
$ husb.edu        : Factor w/ 4 levels "low","mid-low",..: 3 3 3 2 3 4 3 3 3
$ num.child       : int  3 10 7 9 8 0 6 1 3 8 ...
$ islam           : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
$ wife.working    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
$ husb.job        : Factor w/ 4 levels "1","2","3","4": 2 3 3 3 3 3 3 3 3 2 .
$ SOL.index       : Factor w/ 4 levels "low","mid-low",..: 3 4 4 3 2 3 2 2 4
$ media.exposure: logi    TRUE TRUE TRUE TRUE TRUE TRUE ...
$ cmc             : Factor w/ 3 levels "no-use","long-term",..: 1 1 1 1 1 1 1
```

Save the data frame into an R data file and retrieve it to confirm that it was properly saved.

```
  > save( cmc, file = "cmc.rda" )

  > load( file = "cmc.rda" )

  > str( cmc )
'data.frame': 1473 obs. of    10 variables:
$ wife.age         : int  24 45 43 42 36 19 38 21 27 45 ...
$ wife.edu         : Ord.factor w/ 4 levels "low"<"mid-low"<..: 2 1 2 3 3 4 2
$ husb.edu         : Ord.factor w/ 4 levels "low"<"mid-low"<..: 3 3 3 2 3 4 3
$ num.child        : int  3 10 7 9 8 0 6 1 3 8 ...
$ islam            : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
$ wife.working     : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
$ husb.job         : Factor w/ 4 levels "1","2","3","4": 2 3 3 3 3 3 3 3 3 2
$ SOL.index        : Ord.factor w/ 4 levels "low"<"mid-low"<..: 3 4 4 3 2 3 2
$ media.exposure   : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
$ cmu              : Factor w/ 3 levels "no-use","long-term",..: 1 1 1 1 1 1
>
```

2.2 Understanding Data

The structure of the dataset says that there are 1473 obs. of 10 variables. Of which the variable called **`cmc`** is the classification variable with three values. As per the structure of the variable it is of type *categorical* taking three values **`"no-use", "long-term"`** and **`"short-term"`**. The value 1 corresponds to *no-use*, the value 2 corresponds to *"*`long-term`*"* and the value 3 corresponds to *"short-term"*.

The class sizes can be obtained as

```
> table(cmc$cmu)

no-use        long-term short-term
629                 333        511
```

From the above table, we observe that there are three classes and class sizes are not balanced.

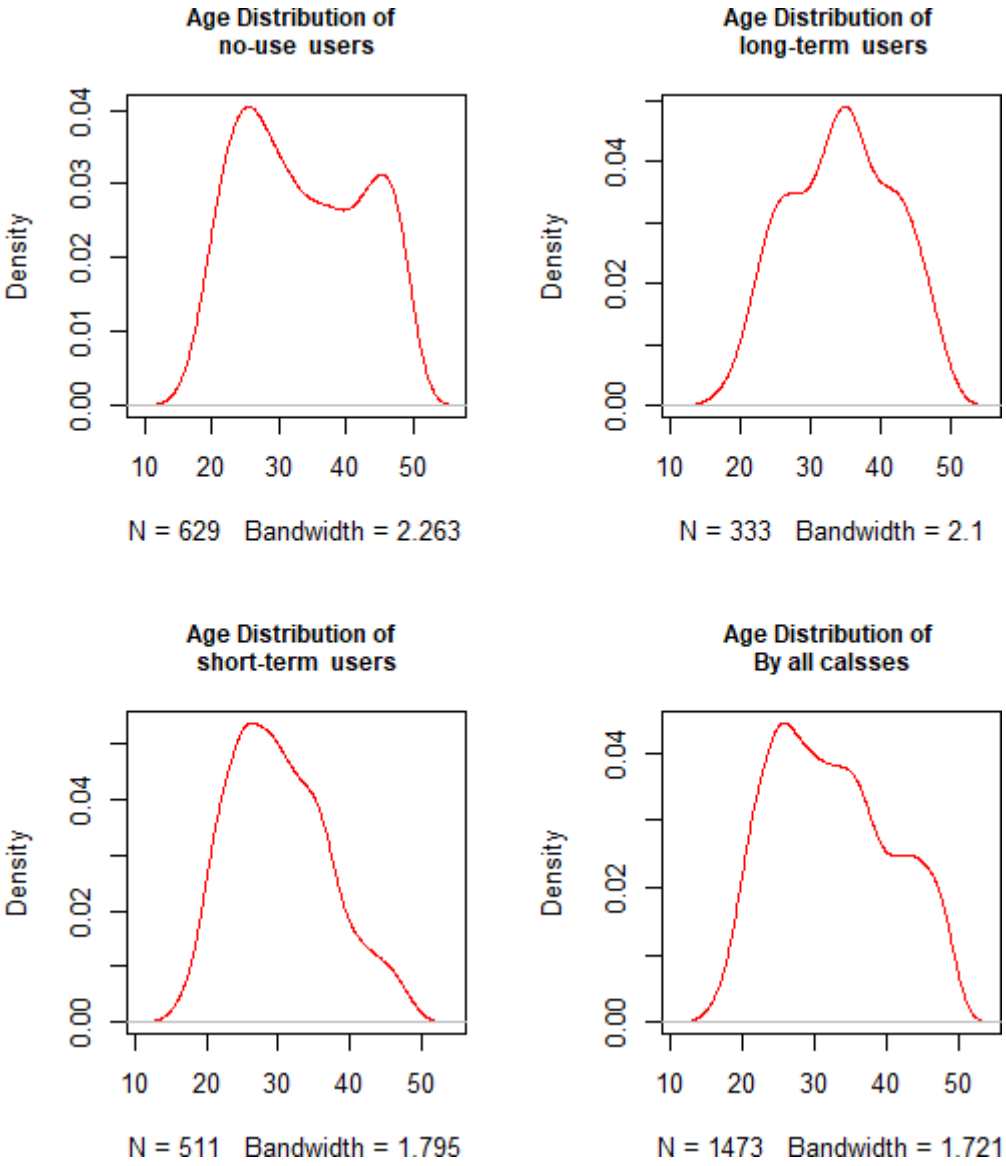
To know the summary statistics of the various variables in our total dataset, we use the **`summary()`** command. This give us an idea about minimum value, maximum value, first quartile, median, third quartile and mean of each of the numeric variables, and frequencies of various levels of categorical variables in the data set.

```
> summary(cmc)
 wife.age            wife.edu       husb.edu     num.child
 Min.   :16.00   low     :152   low     : 44   Min.   : 0.000
 1st Qu.:26.00  mid-low : 334   mid-low :178    1st Qu.: 1.000
 Median :32.00  mid-high: 410   mid-high:352    Median : 3.000
 Mean   :32.54  high       577  high      899  Mean     : 3.261
 3rd Qu.:39.00                                 3rd Qu.: 4.000
 Max.   :49.00                                 Max.     :16.000

 islam          wife.working   husb.job   SOL.index
 Mode :logical  Mode :logical  1:436      low       129
 FALSE:220      FALSE:1104     2:425      mid-low :229
 TRUE :1253     TRUE :369      3:585      mid-high:431
                               4: 27      high      684


 media.exposure          cmu
 Mode  :logical  no-use  :629
 FALSE:109       long-term :333
 TRUE :1364      short-term:511
```

2.2.1 Age Distribution

From the above summary of the variable characteristics, we observe that the age of the women ranges over the entire reproductive ages. The median age of the women included in the study is 32 years. Let us now visualize the age distribution of the women included in the study.

```
# visualizing age distribution among the classes
 > x <- levels(cmc$cmu)
 > par(mfrow=c(2,2))
 > for(k in x)
   plot(density(cmc[cmc$cmu==k,"wife.age"]),
         col = 2, cex.main = 0.9,
         main=paste("Age Distribution of \n",k," users"))
>plot(density(cmc$wife.age),col=2,cex.main=0.9,
main = 0.9,main=paste("AgeDistribution By \nall classes"))

  > par(mfrow=c(1,1))
```



The age distribution of the class"no-users" looks like a bimodal distribution. It has two peaks at about ages 25 and 45 roughly. If we use the `summary()` function, we will not get this information. Even the box-plot will not reveal this bi-modality.

```
> summary(cmc[cmc$cmu=="no-use","wife.age"])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  16.00   25.00   32.00   33.42   42.00   49.00
```

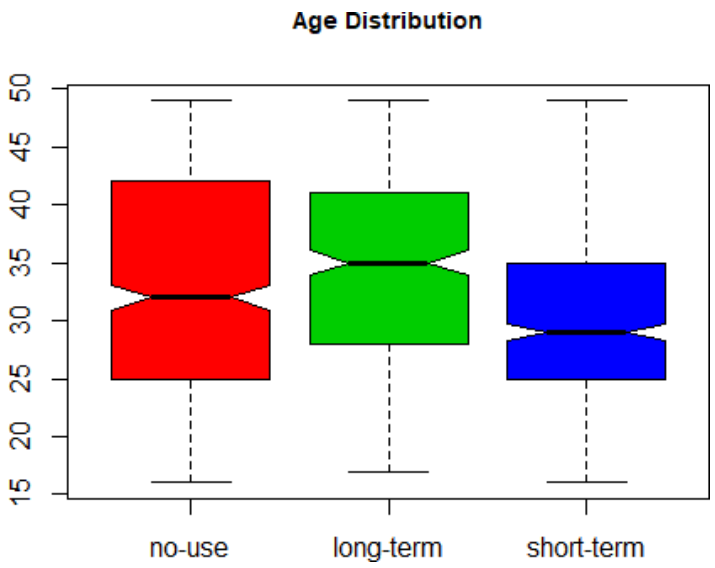
The model value of the age distribution for *short-term* users is 29 years. Which is lower than the class *no-use*. The age-distribution for this class is positively skewed. This can also observed from the box plot. For all the classes, the ages of women ranges from 16 years to 49 years.

```
> summary(cmc[cmc$cmu=="short-term","wife.age"])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  16.00   25.00   29.00   30.24   35.00   49.00
```

The age distribution of *long-term* users look like a multi-model distribution. A large proportion of long-term users are about the age 35 years. It is a little bit hard to under- stand why the minimum age for this class is 17 years. Meanings of the variables and codes used for some of the variables is hard to understand from the supporting file given along with the data file.

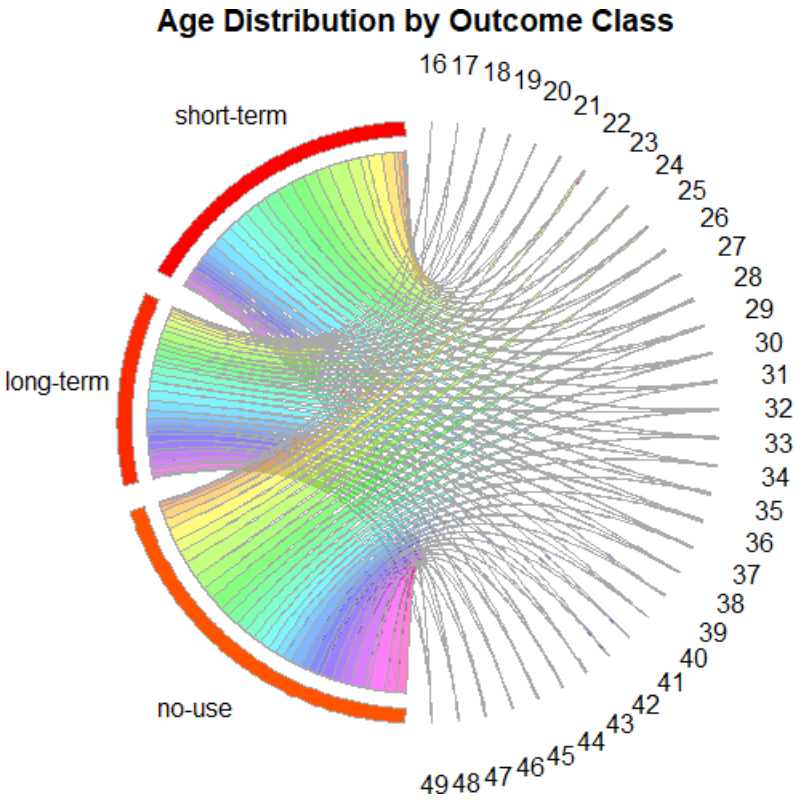
```
> summary(cmc[cmc$cmu=="long-term","wife.age"])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  17.00   28.00   35.00   34.38   41.00   49.00
```

```
> boxplot(cmc$wife.age~cmc$cmu,notch=T,col=c(2,3,4),
          main="Age distribution")
```



Yet another way of visualizing the age distribution of different classes is to use **`PlotCirc()`** function of **`DescTools`** package.

```
> library("DescTools")
> with(cmc, PlotCirc( table(cmc[,1],cmu),
        main=paste("Age Distribution by Outcome Class")))
```



When a data frame is passed to the **`Desc()`** function of `DescTools` package, it summary statistics of various columns of the data frame together with appropriate graphs. It is like the **`summary()`** function of the base R, which output is limited to only five point sum- mary for numeric variables and counts for categorical variables. The `Desc()` function also accepts a single column of a data frame. Apart from the five-point summary, it outputs several other important useful statistics, which includes skewness, kurtosis, SD, CI for mean for numeric variables.

```
> Desc(cmc$wife.age,main="Age Distribution of Women")
```

Age Distribution of Women

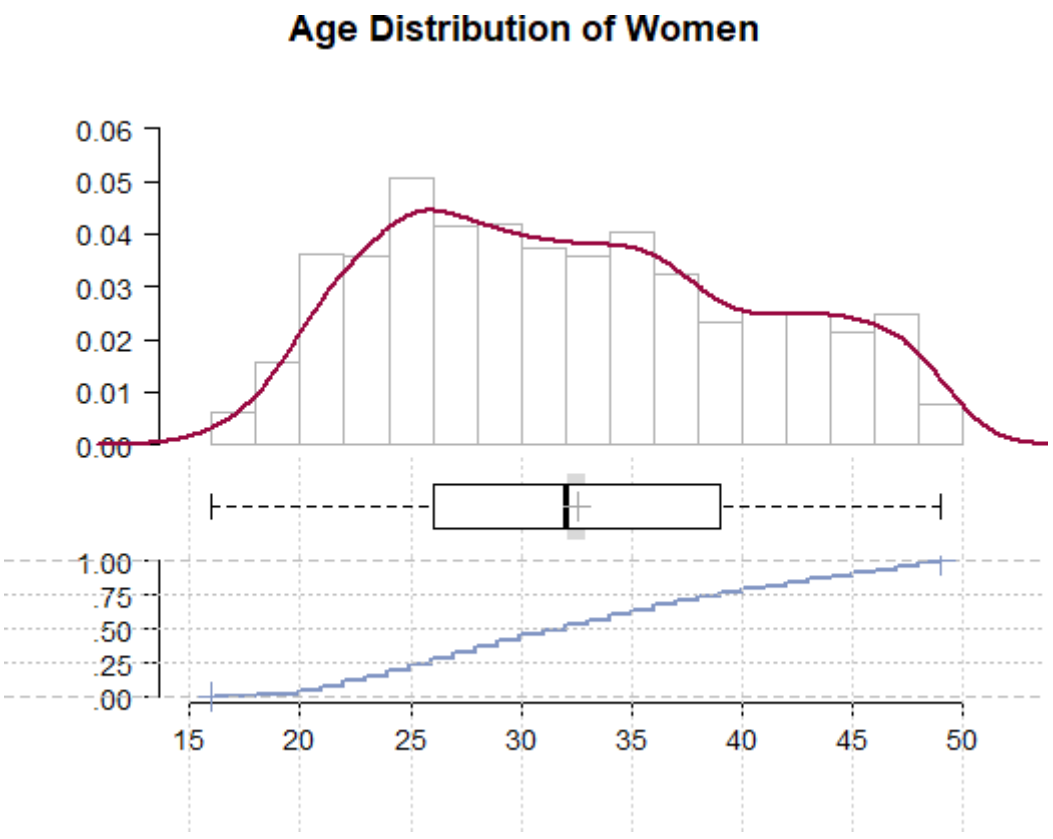
length	n	NAs	unique	0s	mean	meanCI
1’473	1’473	0	34	0	32.54	32.12
	100.0%	0.0%		0.0%		32.96

.05	.10	.25	median	.75	.90	.95
21.00	22.00	26.00	32.00	39.00	45.00	47.00

range	sd	vcoef	mad	IQR	skew	kurt
33.00	8.23	0.25	8.90	13.00	0.26	-0.95

```
lowest : 16 (3), 17 (8), 18 (7), 19 (18), 20 (28)
highest: 45 (41), 46 (22), 47 (43), 48 (30), 49 (23)
```

From the above output, we notice the lowest and the highest 5 observations together their frequencies. The women age distribution in the entire sample is slightly positively skewed.



```
> Desc(cmc[cmc$cmu=="no-use","wife.age"],
        main="Age Distribution of Women : No-Use Class")
```

Age Distribution of Women: No-Use Class

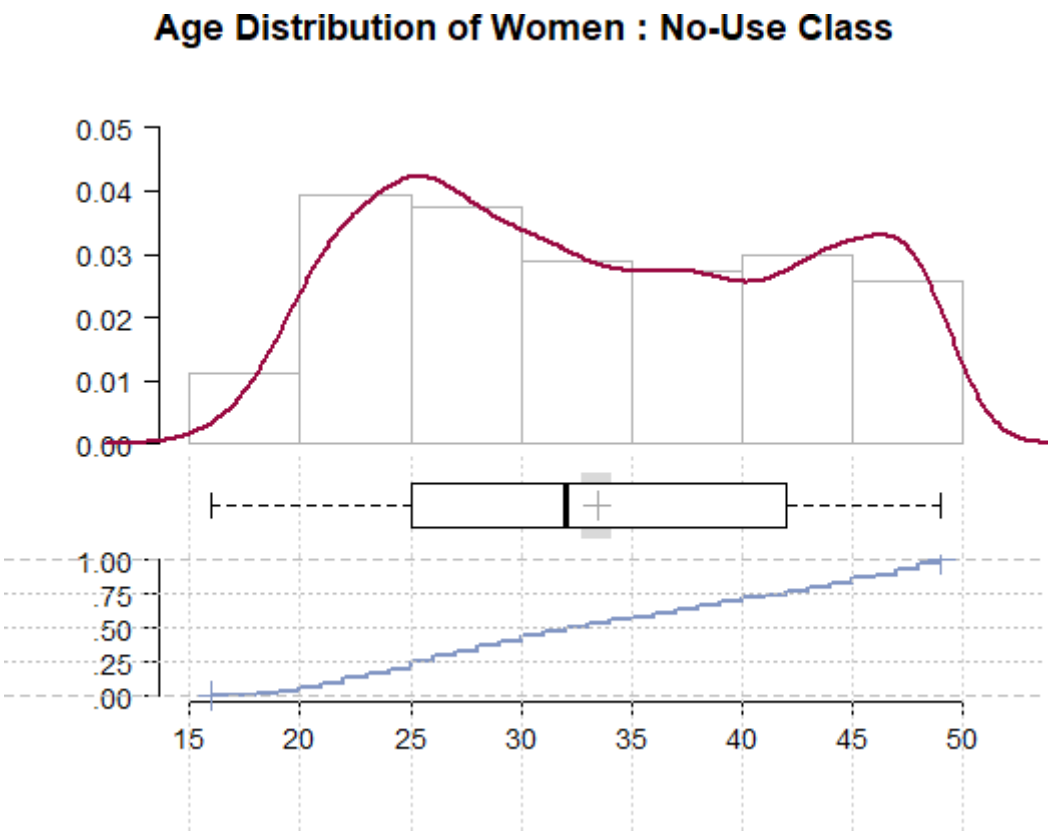
length	n	NAs	unique	0s	mean	meanCI
629	629	0	34	0	33.42	32.71
	100.0%	0.0%		0.0%		34.14

.05	.10	.25	median	.75	.90	.95
20.00	22.00	25.00	32.00	42.00	47.00	48.00

range	sd	vcoef	mad	IQR	skew	kurt
33.00	9.12	0.27	11.86	17.00	0.15	-1.24

```
lowest : 16 (2), 17 (2), 18 (5), 19 (9), 20 (17)
highest: 45 (27), 46 (10), 47 (31), 48 (21), 49 (19)
```

From the above output, we notice the lowest and the highest 5 observations together their frequencies. In no-use class. The women age distribution in the no-use sample is slightly positively skewed.



```
> Desc(cmc[cmc$cmu=="short-term","wife.age"],
          main="Age Distribution of Women : Short-Term Class")
```

Age Distribution of Women: Short-Term Class

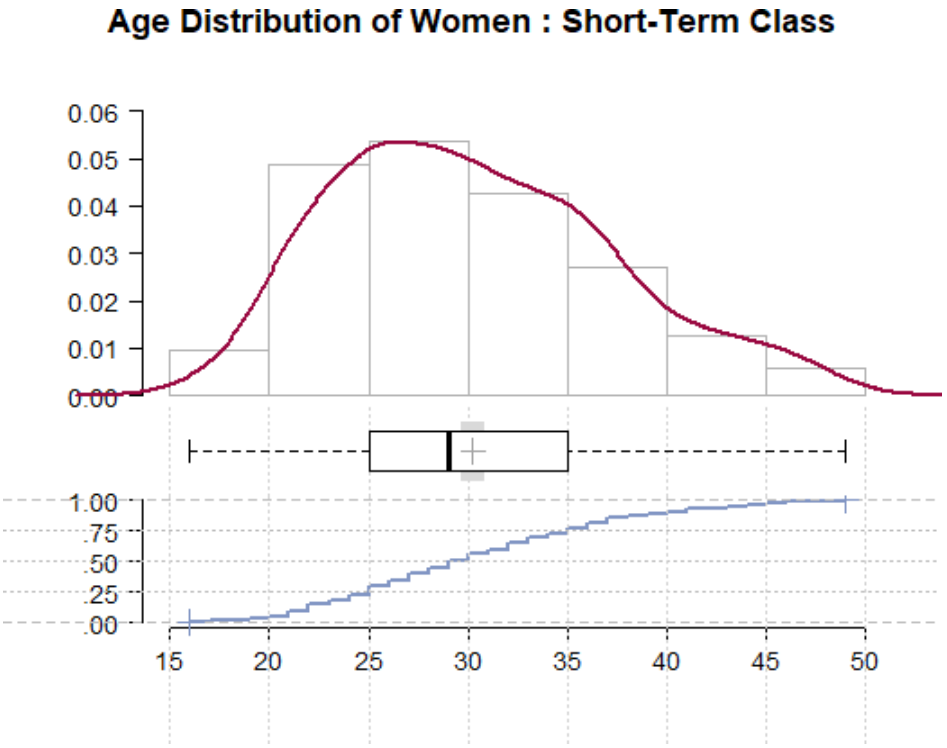
length	n	NAs	unique	0s	mean	meanCI
511	511	0	34	0	30.24	29.64
	100.0%	0.0%		0.0%		30.85

.05	.10	.25	median	.75	.90	.95
21.00	22.00	25.00	29.00	35.00	40.00	43.00

range	sd	vcoef	mad	IQR	skew	kurt
33.00	6.94	0.23	7.41	10.00	0.45	-0.39

lowest : 16, 17 (5), 18, 19 (8), 20 (9)
highest: 45 (4), 46 (7), 47 (4), 48 (3), 49

heap(?): remarkable frequency (7.0%) for the mode(s) (= 25)



From the above output, we notice the lowest and the highest 5 observations together their frequencies. In the short-term class. The women age distribution in the short-term sample is slightly positively skewed.

```
> Desc(cmc[cmc$cmu=="long-term","wife.age"],
        main="Age Distribution of Women : Long-Term Class")
```

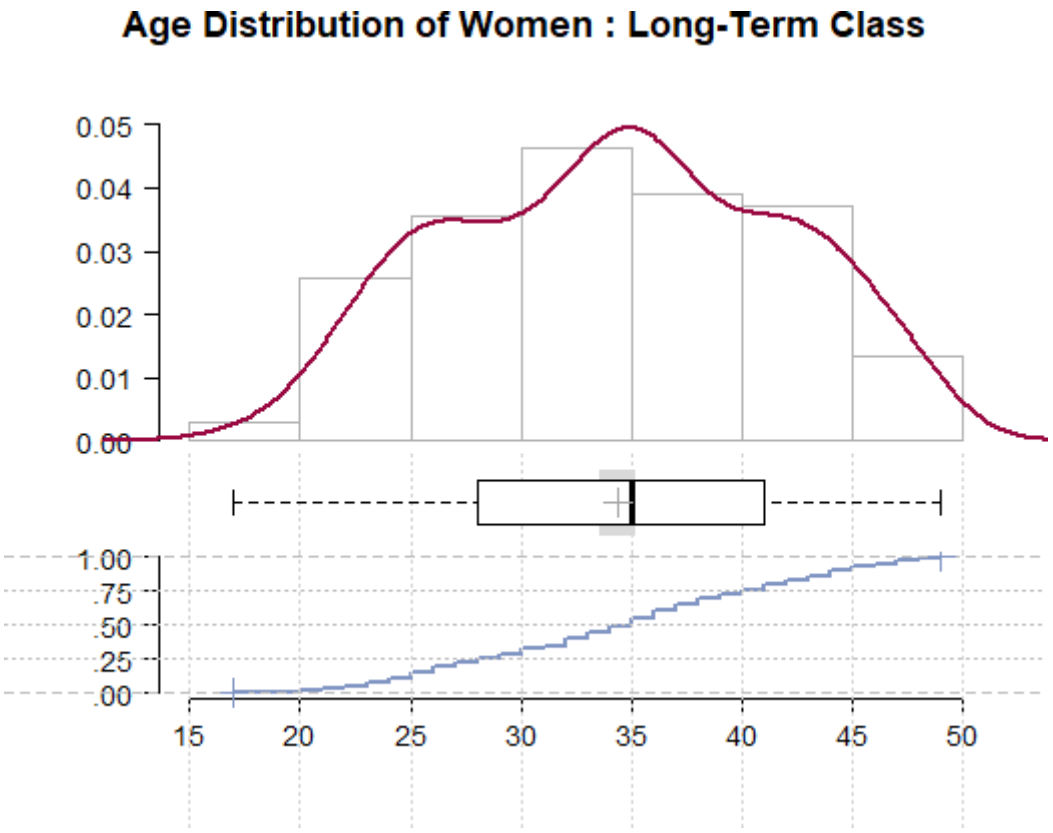
Age Distribution of Women : Long-Term Class

length	n	NAs	unique	0s	mean	meanCI
333	333	0	33	0	34.38	33.58
	100.0%	0.0%		0.0%		35.19

.05	.10	.25	median	.75	.90	.95
22.60	24.00	28.00	35.00	41.00	44.00	46.40

range	sd	vcoef	mad	IQR	skew	kurt
32.00	7.45	0.22	8.90	13.00	-0.04	-0.89

lowest : 17, 18, 19, 20 (2), 21 (6)
highest: 45 (10), 46 (5), 47 (8), 48 (6), 49 (3)

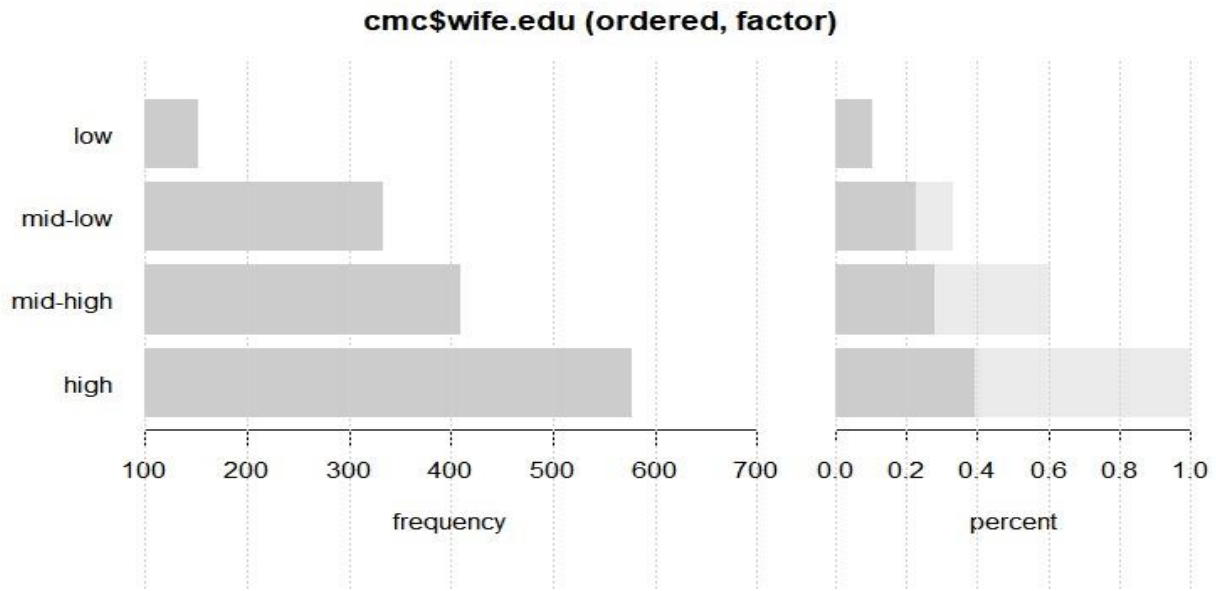


2.2.2 Women Education:

The education levels of women is a categorical variable and the levels are labelled as *low, mid-low, mid-high* and *high*. The `Desc()` function has resulted in the following output:

```
 > Desc(cmc$wife.edu)
 ----------------------------------------------------------------------------------------------------
 cmc$wife.edu (ordered, factor)

length          n    NAs unique levels  dupes
1’473       1’473      0      4      4      y
100.0%              0.0%

level     freq  perc cumfreq    cumperc 1
          low   152  10.3%       152    10.3%
2     mid-low   334  22.7%       486    33.0%
3    mid-high   410  27.8%       896    60.8%
4        high   577  39.2%     1’473   100.0%
```

From the above output we observe that the number of women increases as the level of education increases. Forty percent of women in the sample are having high level of education. Only ten percent of women are with low education level.



2.2.3 Number of Children

We will now explore distribution of number of children among different classes. The summary statistics for the number of children over the various classes are as follows:

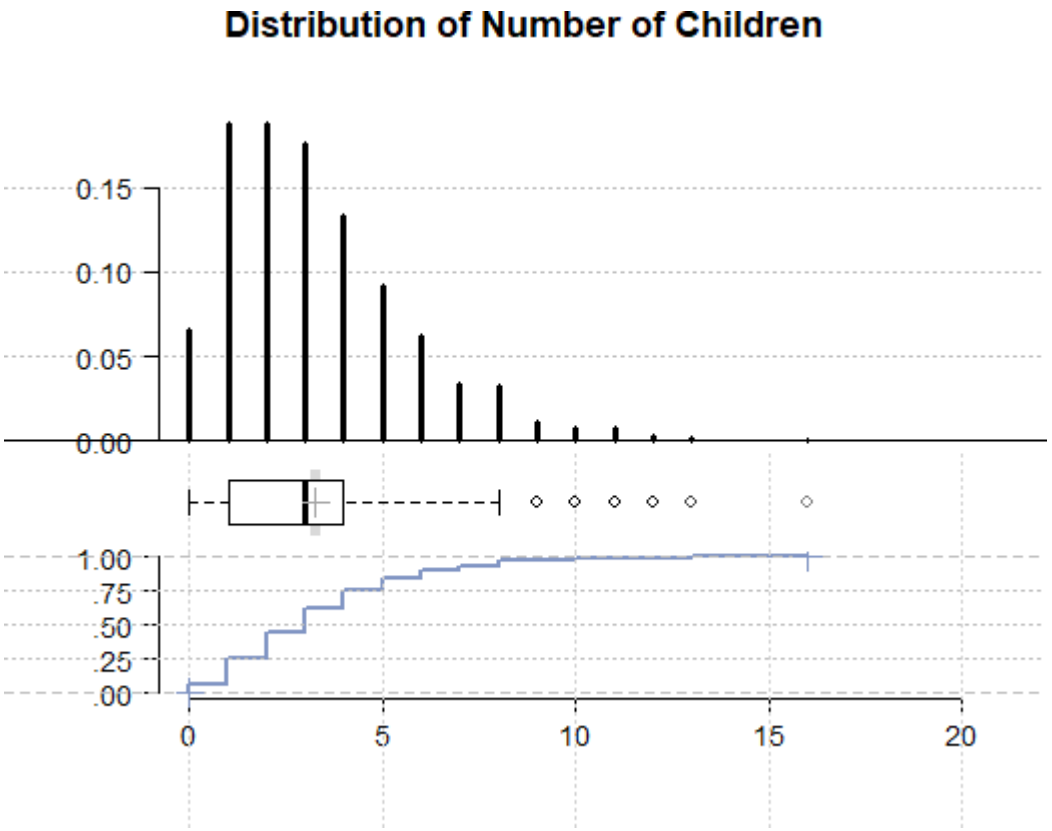
```
> summary(cmc[cmc$cmu=="no-use","num.child"])
  Min.  1st Qu.  Median   Mean    3rd Qu.  Max.
  0.000 1.000    2.000    2.935   4.000    12.000

> summary(cmc[cmc$cmu=="short-term","num.child"])
  Min. 1st Qu.    Median    Mean 3rd Qu.    Max.
   0.000    2.000    3.000    3.352    4.000   16.000

> summary(cmc[cmc$cmu=="long-term","num.child"])
  Min. 1st Qu.    Median    Mean 3rd Qu.    Max.
   1.000    2.000    3.000    3.739    5.000   13.000
```

The average number of children for the classes' **no-use, short-term** and **long-term** are 2.935, 3.352, and 3.739 respectively. These mean values are significantly difference.

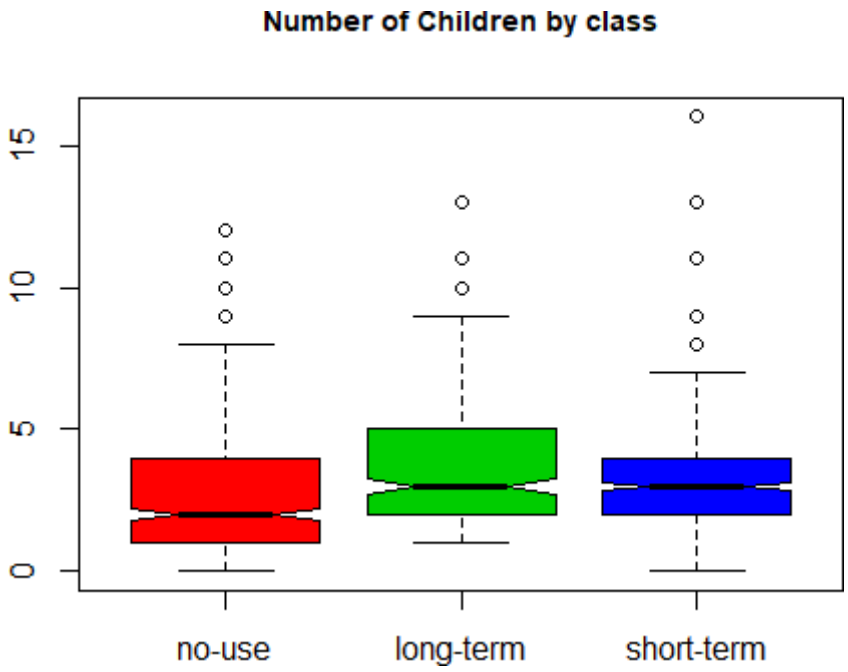
```
> summary(with(cmc,aov(num.child~cmu,)))
Df Sum Sq Mean Sq F value                    Pr(>F)
cmu              2    147   73.59   13.45 1.62e-06 ***
Residuals     1470   8041    5.47
---
Signif. codes:    0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the above output from `aov()` function, we observe that the average number of children for women following different usage durations are statistically significant.

```
> PlotFdist(cmc$num.child, args.hist=list(type="mass"),
            main="Distribution of Number of Children")
```

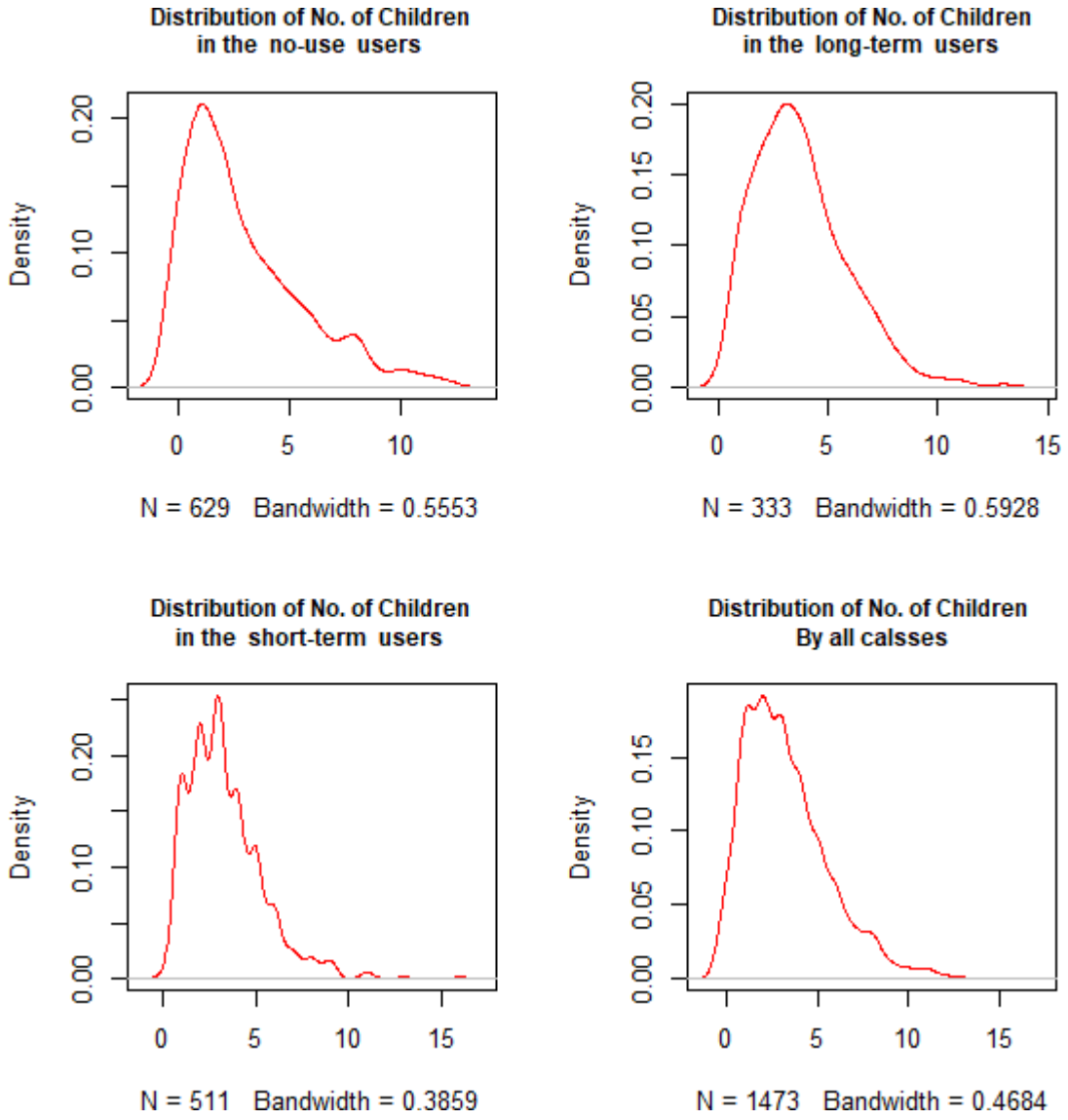



```
>boxplot(cmc$num.child~cmc$cmu,notch=T, col=c(2,3,4),
                                    Main ="Number of children in Class")
```



We observe that the number of children in women. Let us now visualize in the num.child distribution of the women included in the study.

```
>x <- levels(cmc$cmu)
>par(mfrow=c(2,2))
>for(k in x)
>plot(density(cmc[cmc$cmu==k,"num.child"]), col =2,cex.main = 0.9,
      main=paste("Distribution of No. of Children\nin the ", k," users"))
>plot(density(cmc$num.child), col = 2, cex.main = 0.9,
      main=paste("Distribution of No. of Children \nBy all classes"))
>par(mfrow=c(1,1))
```

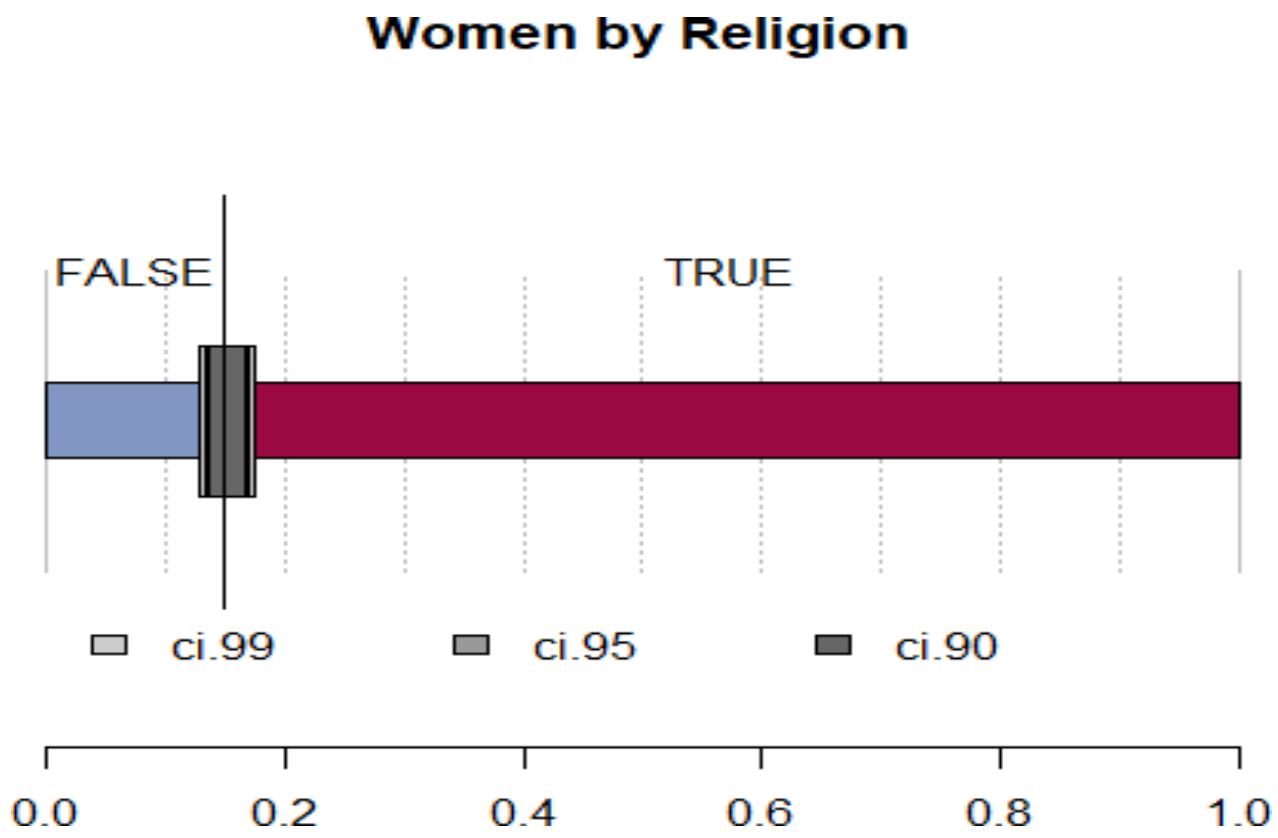


There are some binary variable in the dataset. We now have a look at them.

2.2.4 Islam

```
> Desc(cmc$islam,plotit=T,main="Women by Religion")
```

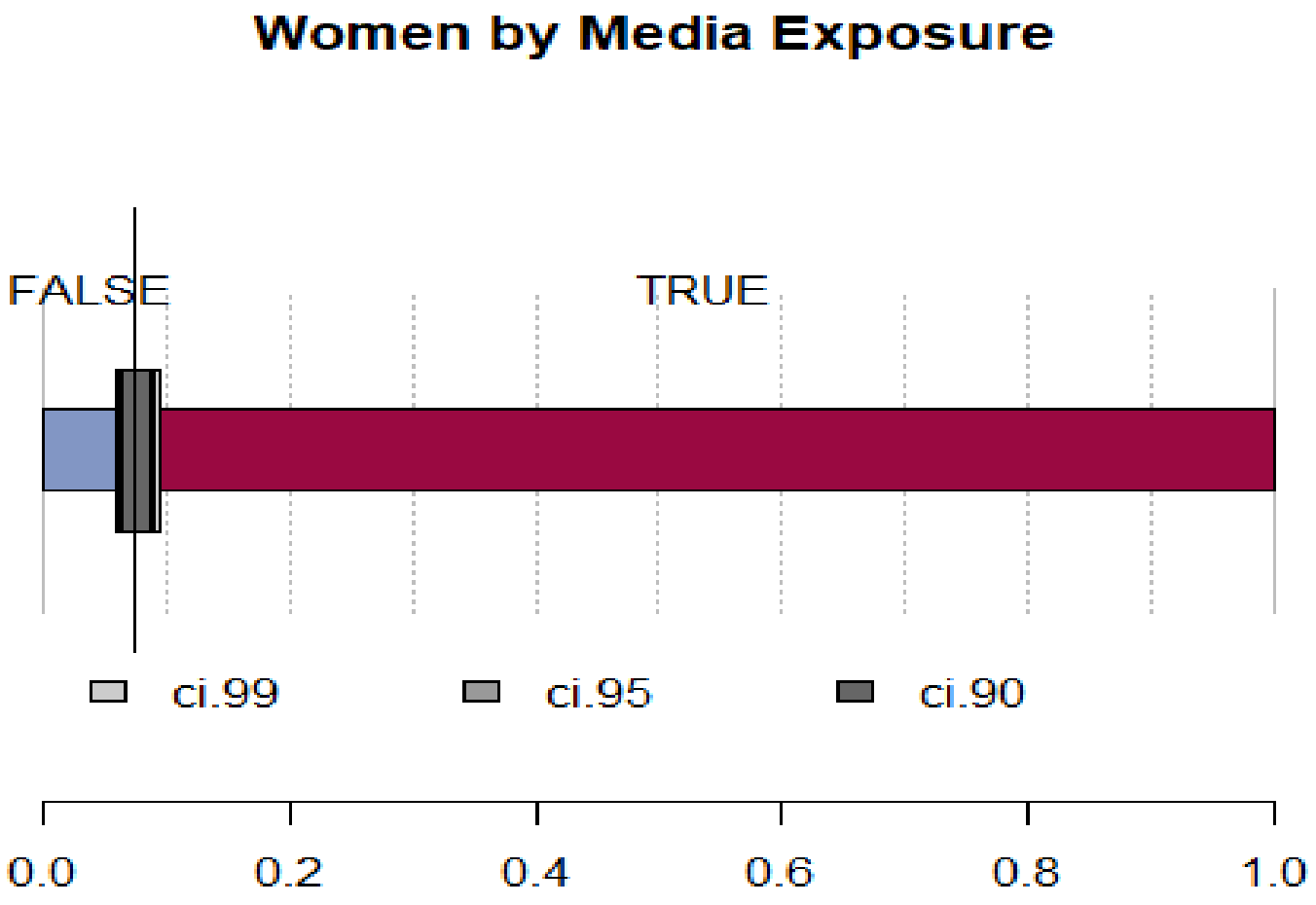
```
Women by Religion

    length      n    NAs unique
     1’473 1’473      0      2
    100.0%          0.0%

          freq    perc   lci.95  uci.95’
FALSE      220   14.9%    13.2%   16.8%
TRUE     1’253   85.1%    83.2%   86.8%

’ 95%-CI Wilson
```



2.2.5 Media-Exposure

```
> Desc(cmc$media.exposure,plotit=T,main="Women by Media Exposure")
```

```
 Women by Media Exposure


length           n     NAs unique
1’473        1’473       0      2
100.0%                0.0%

           freq    perc   lci.95 uci.95’
  FALSE     109    7.4%     6.2% 8.9%
  TRUE    1’364   92.6%    91.1% 93.8%

’ 95%-CI Wilson
```

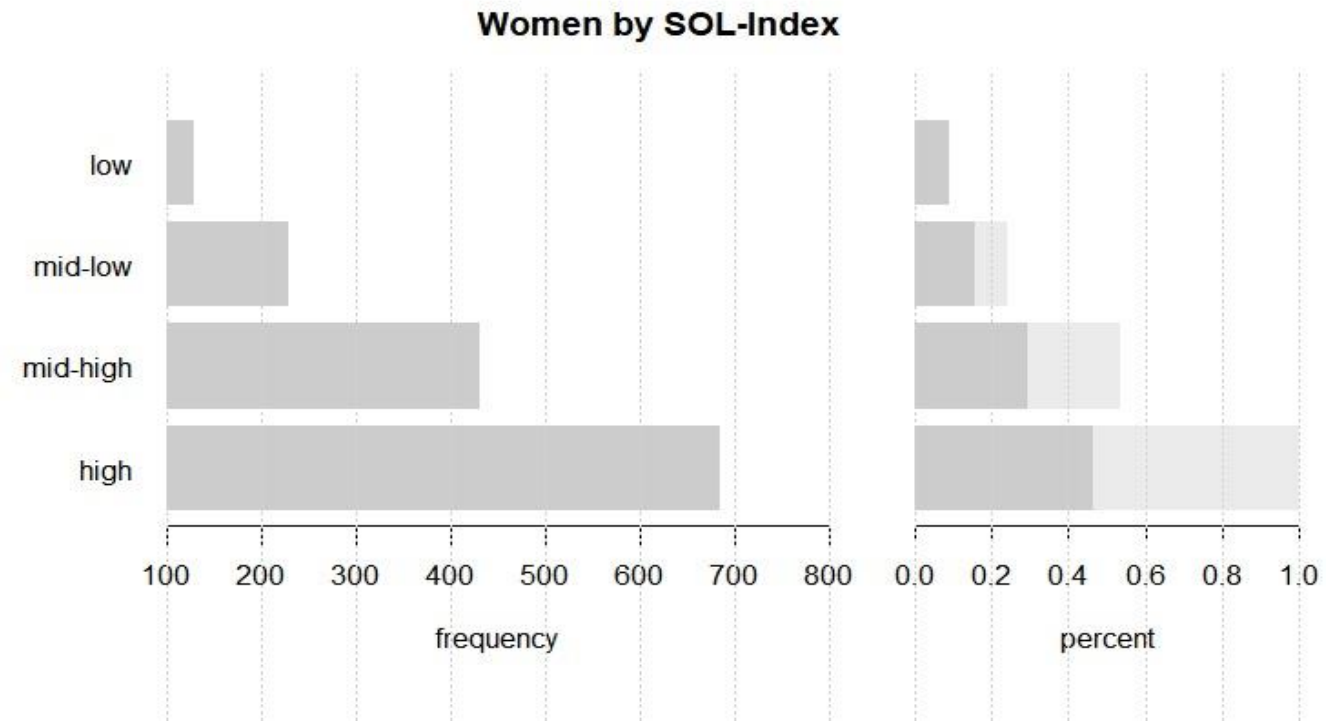


2.2.6 Status Of Living Index

```
> Desc(cmc$SOL.index,main="Women by SOL-Index")
```

```
 Women by SOL-Index

  length      n    NAs unique levels  dupes
   1’473  1’473      0      4      4      y
         100.0%   0.0%

      level   freq   perc  cumfreq  cumperc
1       low    129   8.8%      129     8.8%
2    mid-low   229  15.5%      358    24.3%
3   mid-high   431  29.3%      789    53.6%
4       high   684  46.4%    1’473   100.0%
```

```
> plot(Desc(cmc$SOL.index)[[1]]$freq[,3],
            type="h",ylab="percent",
            main="Women by SOL-Index")
```



2.2.7 Husband Education

```
> Desc(cmc$husb.edu)
----------------------------------------------------------------------------------------------------------------------------
cmc$husb.edu (ordered, factor)

   length     n    NAs unique levels  dupes
    1’473 1’473      0      4      4      y

   100.0%          0.0%

   level       freq perc     cumfreq   cumperc
1        low    44   3.0%        44     3.0%
2    mid-low   178  12.1%       222    15.1%
3   mid-high   352  23.9%       574    39.0%
4       high   899  61.0%     1’473   100.0%
```

2.2.8 Husband’s Job

```
> Desc(cmc$husb.job)
-----------------------------------------------------------------------------------------------
cmc$husb.job (factor)

length        n    NAs unique levels  dupes
   1’473  1’473      0      4      4     y
          100.0%   0.0%

    level freq    perc     cumfreq cumperc
1       3 585     39.7%        585 39.7%
2       1 436     29.6%     1’021     69.3%
3       2 425     28.9%     1’446     98.2%
4       4    27   1.8%      1’473   100.0%
```

2.3 Association

In this section we explore how the explanatory variables are related to the class variable.

Cramer's V

Cramer's *V* is a statistic used to measure the strength of association between two nominal variables, and it take values from 0 to 1. Values close to 0 indicate a weak association between the variables and values close to 1 indicate a strong association between the variables. *V* defines a perfect relationship as one which is predictive or ordered monotonic, and defines a null relationship as statistical independence.

The Cramer's *V* statistic is a symmetric measure, in the sense that it does not matter what variable is placed in the rows and what variable is placed in the columns. These measures do not lend themselves to easy interpretation. The Cramer's *V* statistic is computed using the following formula:

$$\mathrm{V}=\sqrt{\frac{\chi 2/n}{\min\{c,r\}-1}}$$

Where *r* corresponds to the number of rows, and *c* corresponds to the number of columns. *V* may be viewed as the association between two variables as a percentage of their maximum possible variation. V^2 is the mean square canonical correlation between the variables.

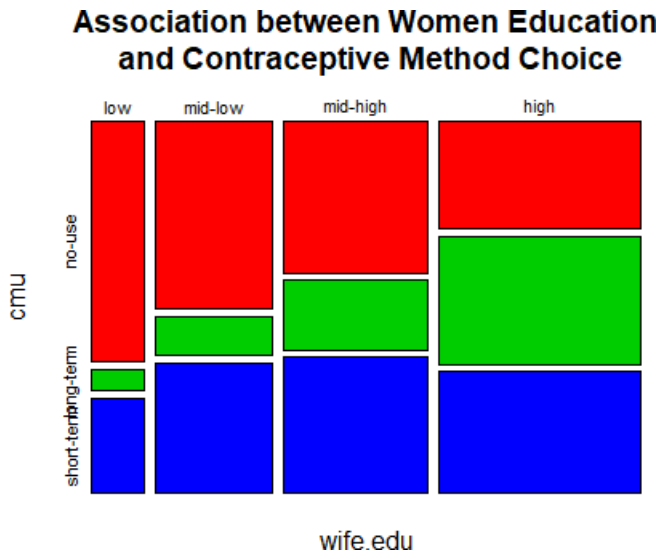
Cramer's *V* is a measure of effect size.

Let us now examine whether there is any association between education level of women and their choice of contraceptive method.

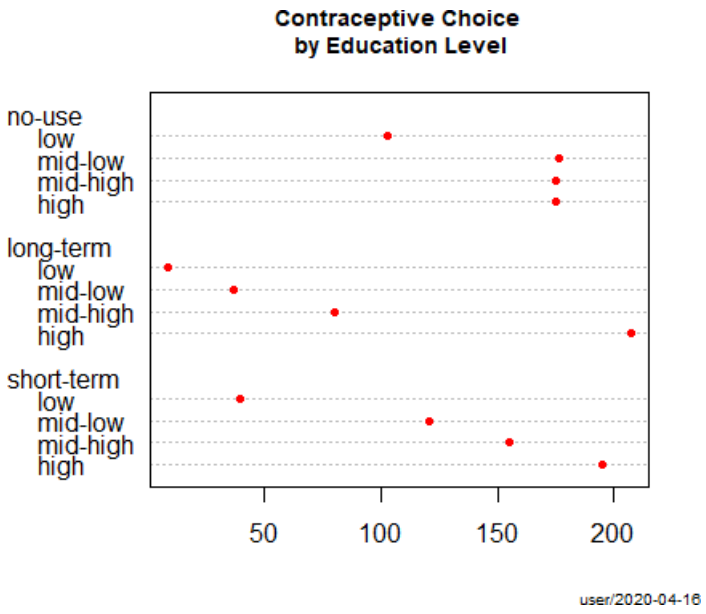
```
> with(cmc,table(wife.edu,cmu))
   cmu
    wife.edu     no-      long-      short-
                 use       term        term
    low          103          9          40
    mid-low      176         37         121
    mid-high     175         80         155
    high         175        207         195
```

As both the variables are categorical, their association may be visualized using a mosaic plot. The following command will result in the mosaic plot.

```
> plot(with (cmc, table(wife.edu, cmu ) ),col=c(2,3,4),
                          main="Association   between   Women
                Education\n andContraceptive Method Choice")
```




```
>library(DescTools)
>with(cmc,PlotDot(table(wife.edu,cmu),
main="Contraceptive Choice\n by Education Level", pch=20,col
=2,cex.main=.9 ))
```



PrintCirc() function:

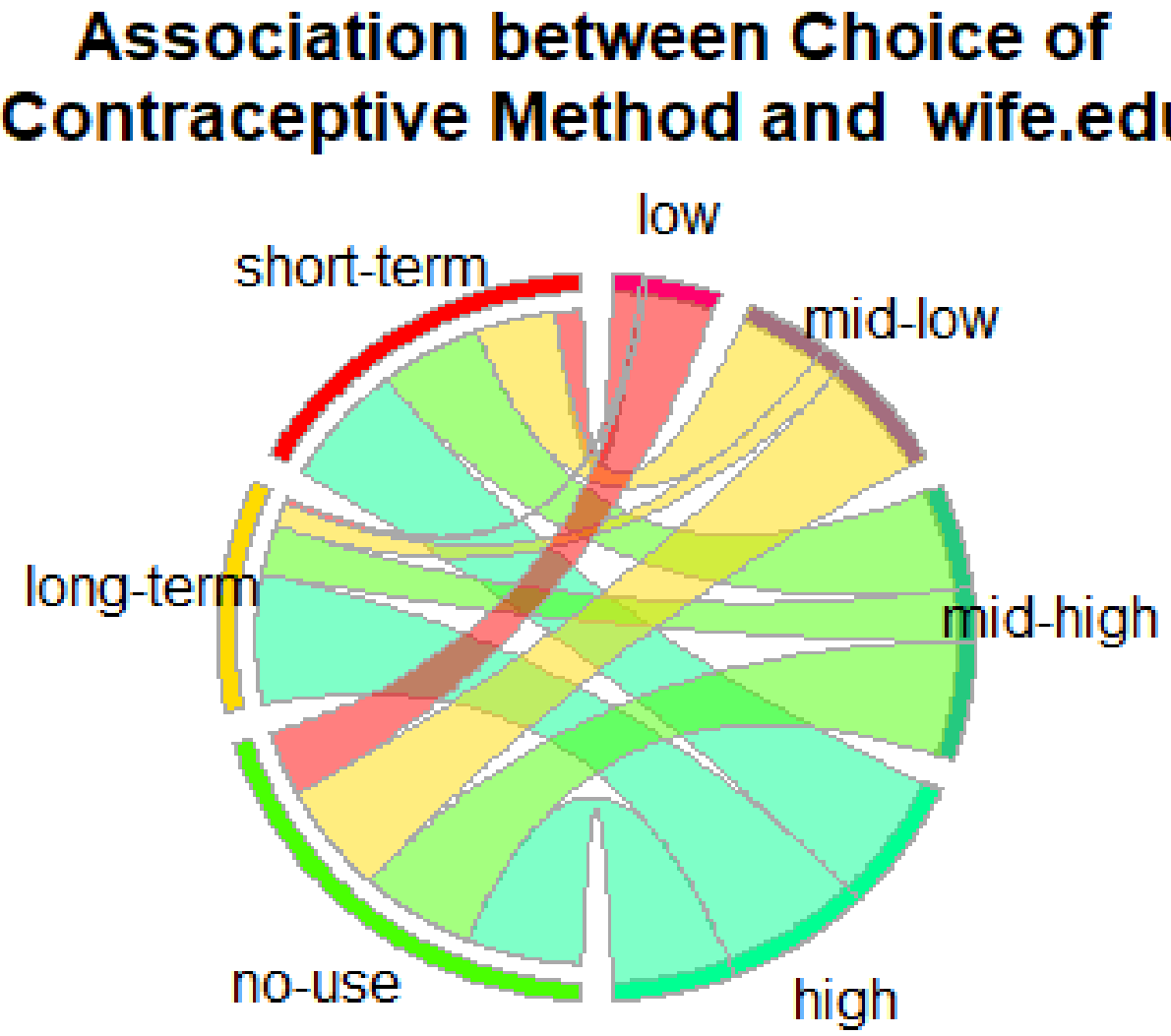
This function describes proportions in circles. It emphasizes the association structure of the data. The left side of the circle represents the columns, say the choice of the contraceptive, and the right one the rows, thus the education levels of women. The advantage of this plot is that we see both marginal densities in the plot. From the figure we notice that the proportion of women increases as we move from *low- level* of education to *high-level* of education.

Looking at the category of women having *high* level of education, first of all we no- tice, that roughly about 40% of women in the sample are having high-level of education. Within those,

percentages of women belonging to the classes ”no-use”, ”short-term” and ”long-term” are roughly 30%, 34% and 36% respectively. Similarly, we can read the other proportions from the figure.

Obviously we see more (conditional) proportions in a circular plot than on a mosaic plot. A disadvantage is that angles are nowhere near as good to compare as the lengths in the mosaic.

```
>with( cmc, PlotCirc( table(wife.edu,cmu),main=”Association between
Choice of Contraceptive Method and wife.edu “) )
```



A formal test of association may be conducted using the chisq.test() function.

```
>chisq.test(with(cmc,table(wife.edu,cmu)))

 Pearson's Chi-squared test
 data: with(cmc, table(wife.edu, cmu))
 X-squared = 140.46, df = 6, p-value < 2.2e-16
```

We reject the hypothesis of no association between women's education and her choice of contraceptive method at 5% level of significance of significance as the p-value $2.2e-16$ is smaller than 0.05. So, this variable may be of some importance in determining the class label in our classification problem.

We now consider the variable husb.edu. From the common sense point of view, we expect association between education levels of women and their husbands. This may tested formally again using the chisq.test()function.

```
with( cmc, table(wife.edu,husb.edu))
husb.edu
  wife.edu    l     mid-      mid-    hi
              o      low      high    gh
              w
  low         2       61        43    22
              6
  mid-low     1       88       130    10
              2                         4
  mid-high    5       25       151    22
                                      9
  high        1        4        28    54
                                      4
```

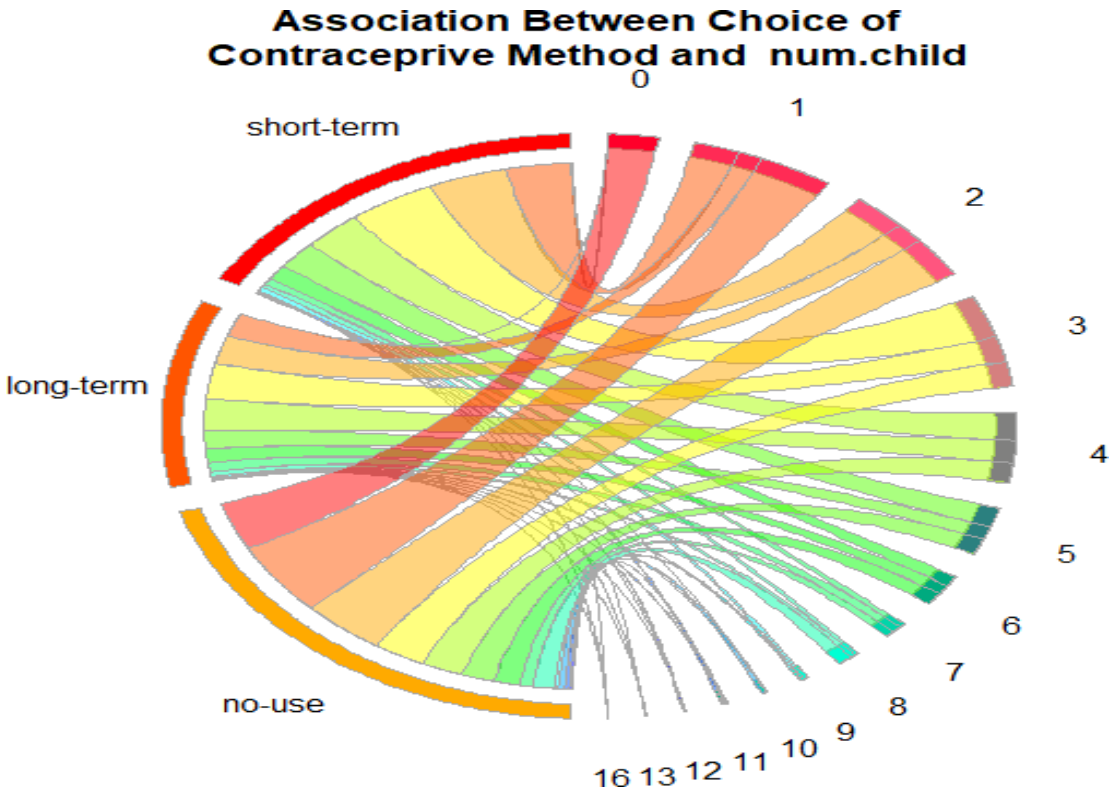
```
> chisq.test(with(cmc,table(wife.edu,husb.edu)))

  Pearson's Chi-squared test

  data:  with(cmc, table(wife.edu, husb.edu))
  X-squared = 708.96, df = 9, p-value < 2.2e-16

  Warning message:
  In chisq.test(with(cmc, table(wife.edu, husb.edu))) :
  Chi-squared approximation may be incorrect


> library(vcd)
> with(cmc, assocstats( table(wife.edu,husb.edu)))
```

```
X^2 df P(> X^2) Likelihood Ratio
719.02                    9         0
Pearson          708.96   9         0
Phi-Coefficient   : NA
Contingency Coeff.: 0.57
Cramer's V        : 0.401
```

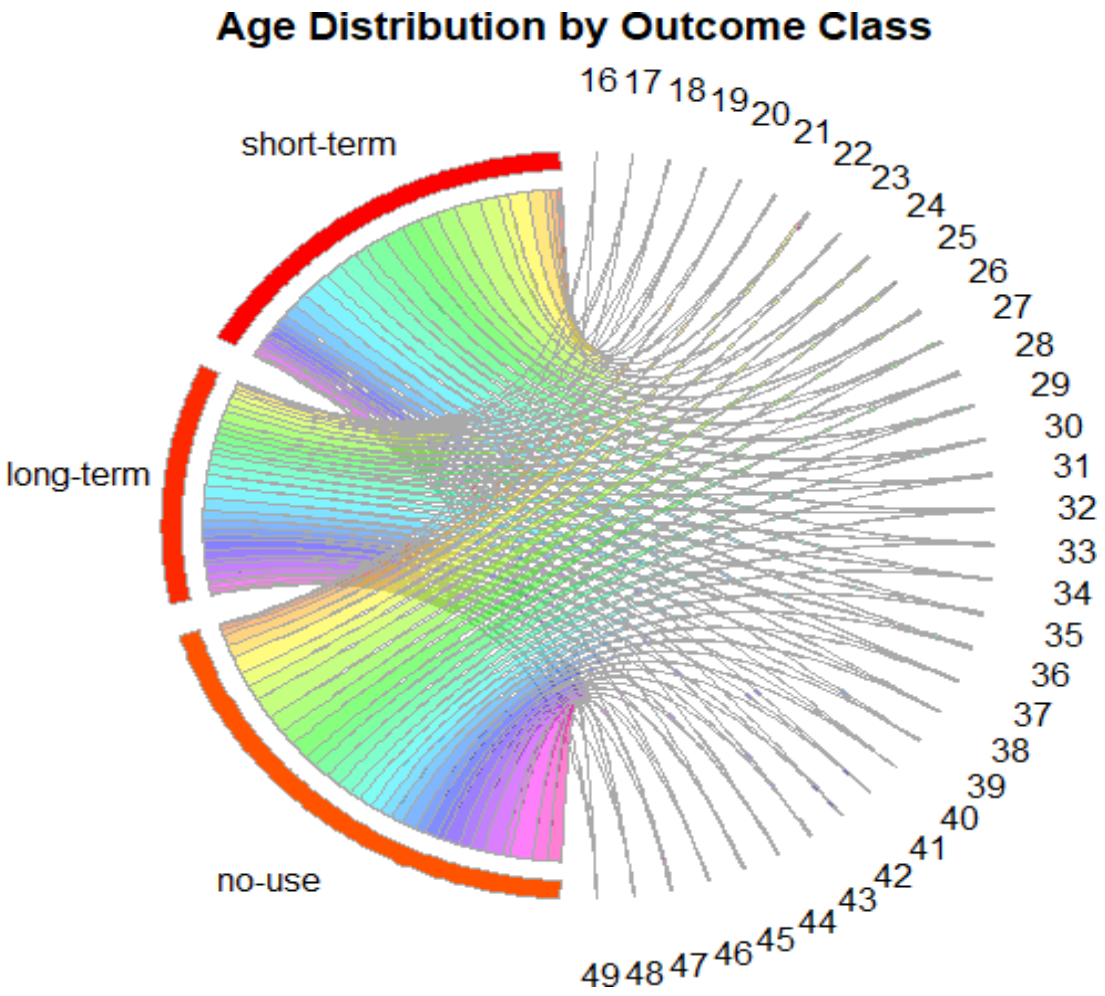
The hypothesis of independence between husband's education level and his wife's education is rejected at 5% level of significance as the p-value is less than the specified level of significance. So, as these variables are related, one of them may become redundant in the decision tree algorithm.

```
> with(cmc,assocstats(table(num.child,cmu)))
X^2 df P(> X^2) Likelihood Ratio
250.78 28                          0
Pearson          219.09 28         0

Phi-Coefficient   : NA
Contingency Coeff.: 0.36
Cramer's V        : 0.273
```

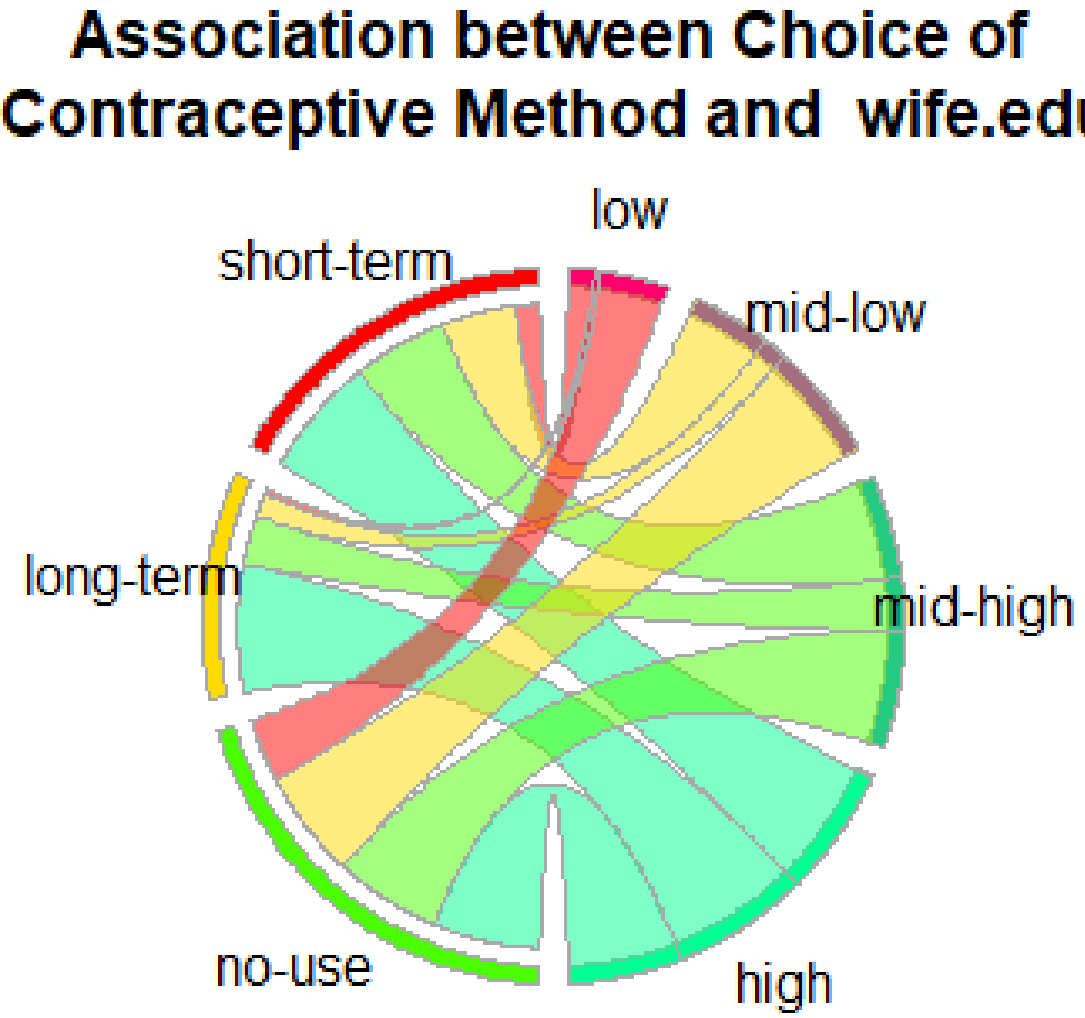
```
> with( cmc, PlotCirc( table(num.child,cmu),main="Association
Between Choice of Contraceptive Method and num.child ") )
```



```
> with(cmc,assocstats(table(wife.age,cmu)))
  X^2 df                        P(> X^2)
  Likelihood Ratio 172.26 66 1.9784e-11
  Pearson          160.86 66 6.8766e-10

  Phi-Coefficient   : NA
  Contingency Coeff.: 0.314
  Cramer's V        : 0.234

  >with( cmc, PlotCirc( table(wife.age,cmu),main="Age Distribution by
  Outcome Class ") )
```

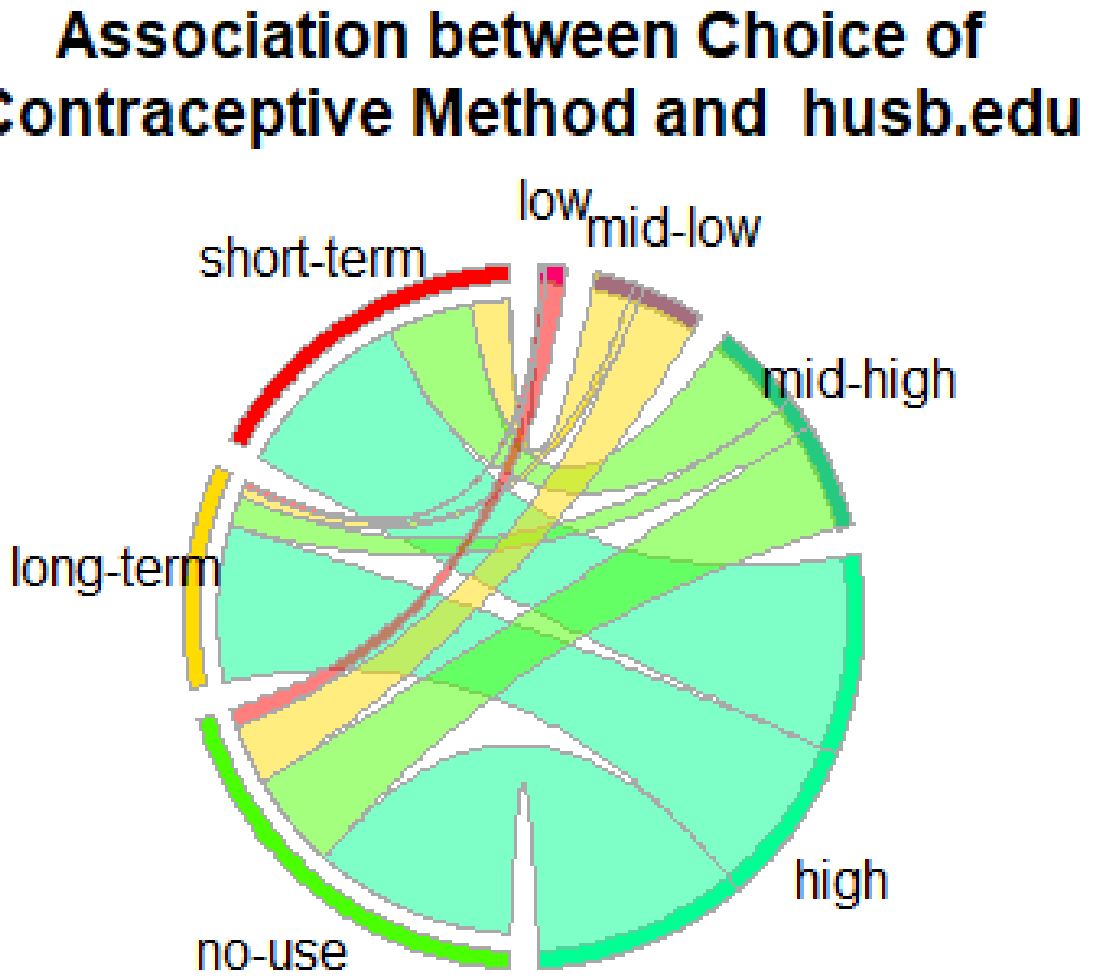


```
> with(cmc,assocstats(table(wife.edu,cmu)))
  X^2 df P(> X^2) Likelihood Ratio
  144.79                   6        0
  Pearson          140.46  6        0

  Phi-Coefficient   : NA
  Contingency Coeff.:
  0.295 Cramer's V  : 0.218

  >with( cmc, PlotCirc( table(wife.edu,cmu),main="Association between
  Choice of Contraceptive Method and wife.edu ") )
```



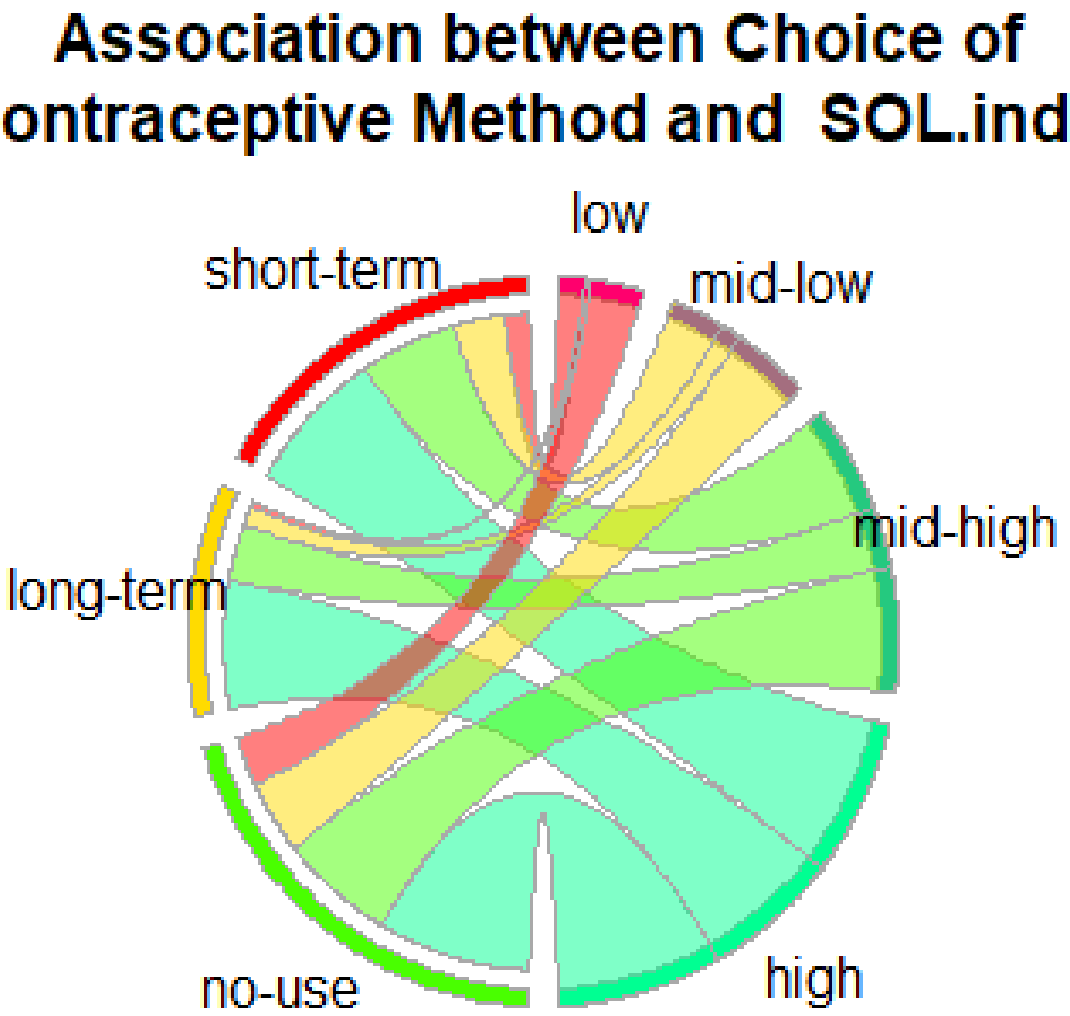
```
> with(cmc,assocstats(table(husb.edu,cmu)))
  X^2 df                    P(> X^2)
  Likelihood Ratio 81.963   6 1.4433e-15
  Pearson          73.953   6 6.3061e-14

  Phi-Coefficient   : NA
  Contingency Coeff.: 0.219
  Cramer's V        : 0.158
```

```
>with( cmc, PlotCirc( table(husb.edu,cmu),main="Association between
Choice of Contraceptive Method and husb.edu ") )
```

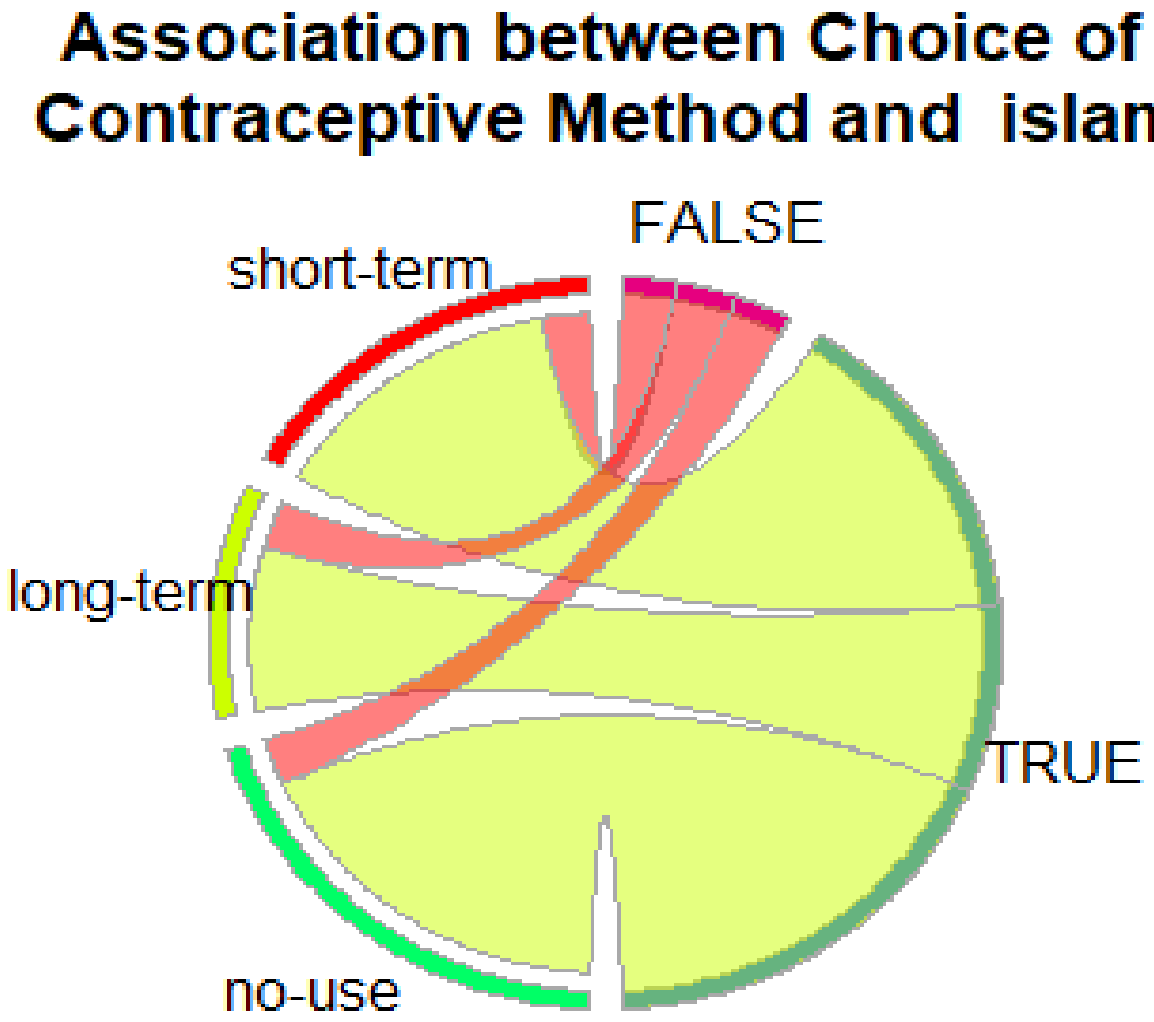


```
> with(cmc,assocstats(table(SOL.index,cmu)))
  X^2 df                    P(> X^2)
  Likelihood Ratio 66.389  6 2.2444e-12
  Pearson          62.199  6 1.6070e-11

  Phi-Coefficient   : NA
  Contingency Coeff.:
  0.201 Cramer's V  : 0.145
```

```
>with( cmc, PlotCirc( table(SOL.index,cmu),main="Association between
Choice of Contraceptive Method and SOL.index ") )
```

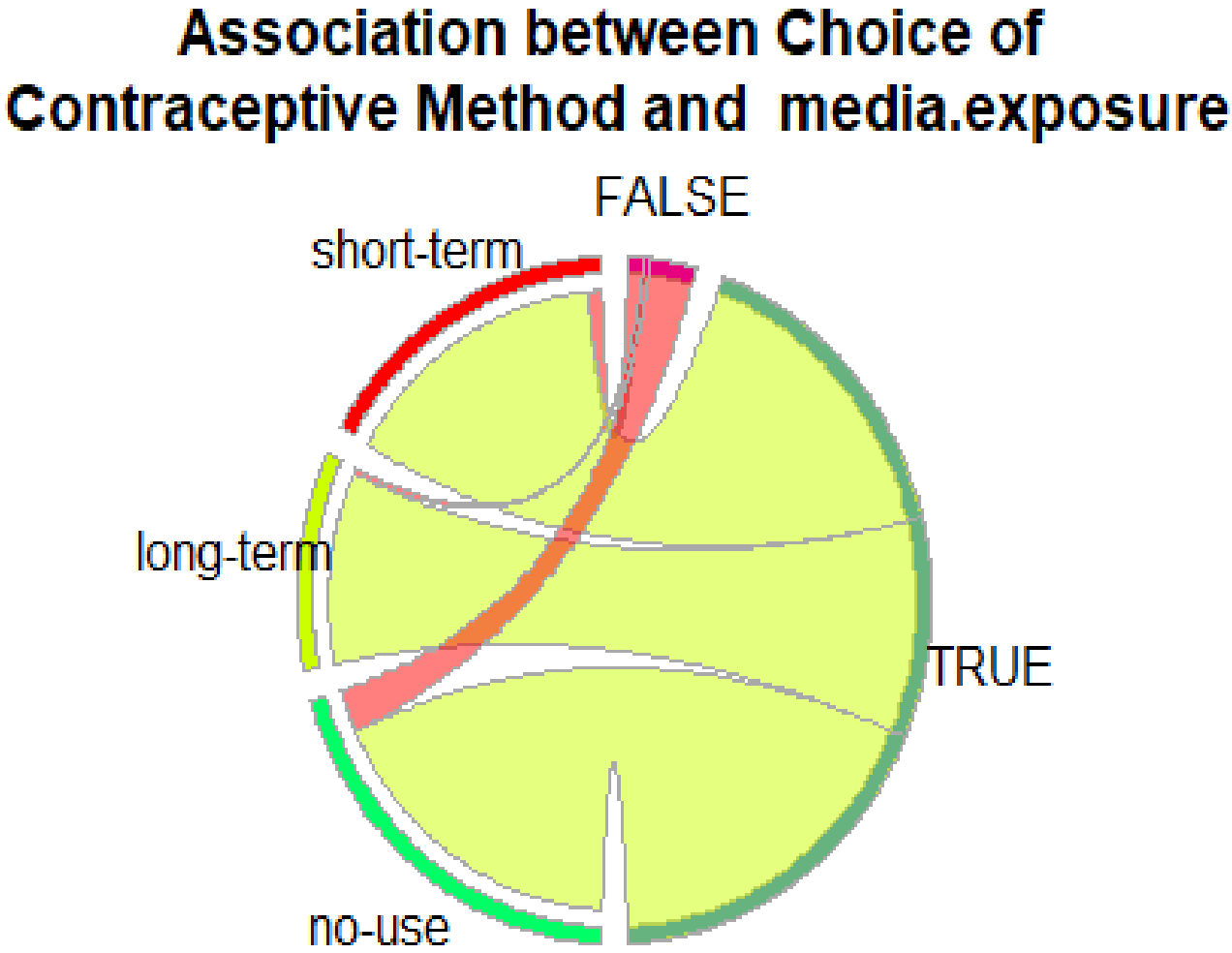


```
> with(cmc,assocstats(table(islam,cmu)))
  X^2 df                    P(> X^2)
  Likelihood Ratio 20.054   2 4.4200e-05
  Pearson          21.622   2 2.0177e-05

  Phi-Coefficient   : NA
  Contingency Coeff.: 0.12
  Cramer's V        : 0.121

  >with( cmc, PlotCirc( table(islam,cmu),main="Association between Choice
  of Contraceptive Method and islam ") )
```



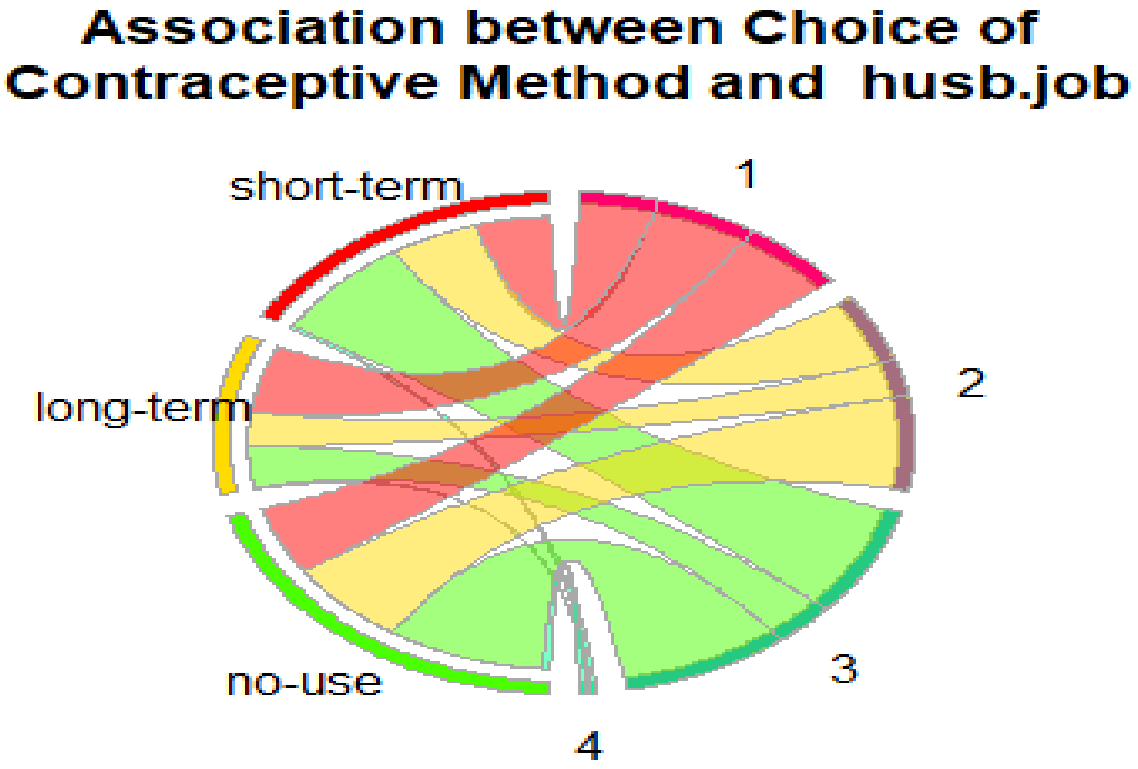
```
> with(cmc,assocstats(table(media.exposure,cmu)))
  X^2 df                  P(> X^2)
  Likelihood Ratio 32.236 2 1.0000e-07
  Pearson          31.572  2 1.3937e-07

  Phi-Coefficient   : NA
  Contingency Coeff.:
  0.145 Cramer's V  : 0.146
```

```
>with( cmc, PlotCirc( table(media.exposure,cmu),main="Association between
Choice of Contraceptive Method and media.exposure ") )
```



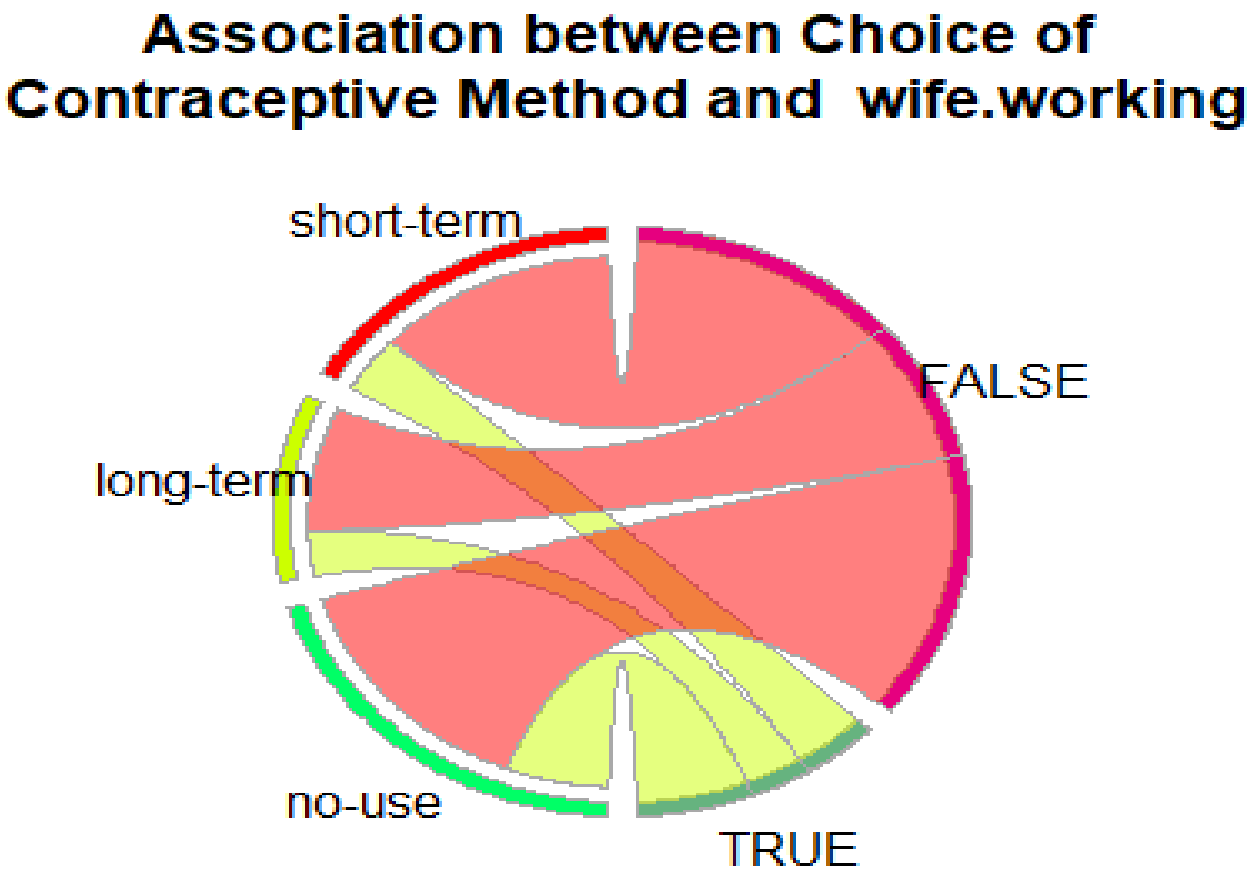
```
> with(cmc,assocstats(table(husb.job,cmu)))
                    X^2 df   P(> X^2)
Likelihood Ratio 62.229  6 1.5849e-11
Pearson          65.401  6 3.5726e-12

Phi-Coefficient   : NA
Contingency Coeff.: 0.206
Cramer's V        : 0.149
```

```
>with( cmc, PlotCirc( table(husb.job,cmu),main="Association between
Choice of Contraceptive Method and husb.job") )
```




```
> with(cmc,assocstats(table(wife.working,cmu)))

                    X^2 df P(> X^2)
Likelihood Ratio 5.2732  2 0.071606
Pearson          5.1872  2 0.074752

Phi-Coefficient   : NA
Contingency Coeff.: 0.059
Cramer's V        : 0.059
```

```
>with( cmc, PlotCirc( table(wife.working,cmu),main="Association between
Choice of Contraceptive Method and wife.working") )
```




```
> library(rcompanion)

> with(cmc,cramerV(table(wife.edu,husb.edu),verbose   = T))

   Rows              4
   Columns           4
   N                 1473
   Chi-squared       709
   Phi               0.48
                     13
   Corrected         NA
   Phi
   V                 0.40
                     05
   Corrected V       NA


   Cramer V
  0.4005
```

```
> Desc(cmu ~ num.child, data=cmc)
```

---

cmu ~ num.child

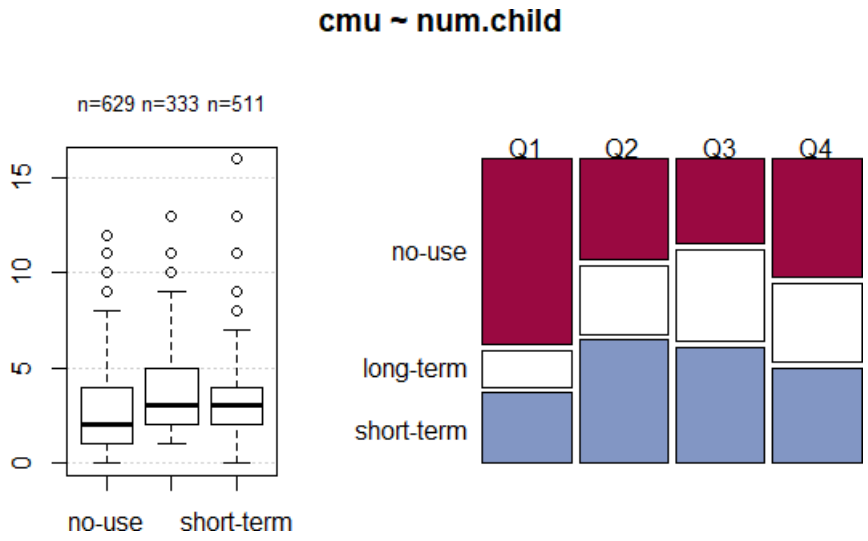
Summary:
n pairs: 1’473, valid: 1’473 (100.0%), missings: 0 (0.0%), groups: 3

	no-use	long-term	short-term
mean	2.935	3.739	3.352
median	2.000	3.000	3.000
sd	2.655	2.104	2.050
IQR	3.000	3.000	2.000
n	629	333	511
np	42.702%	22.607%	34.691%
NAs	0	0	0
0s	95	0	2

Kruskal-Wallis rank sum test:
Kruskal-Wallis chi-squared = 59.662, df = 2, p-value = 1.108e-13

Proportions of cmu in the quantiles of num.child:

	Q1	Q2	Q3	Q4
no-use	63.8 %	34.4 %	28.9 %	40.8 %
long-term	12.3 %	23.6 %	31.5 %	26.9 %
short-term	23.9 %	42.1 %	39.6 %	32.3 %




```
> Desc(cmu ~ wife.age, data=cmc)
```

---

cmu ~ wife.age

Summary:
n pairs: 1’473, valid: 1’473 (100.0%), missings: 0 (0.0%), groups: 3

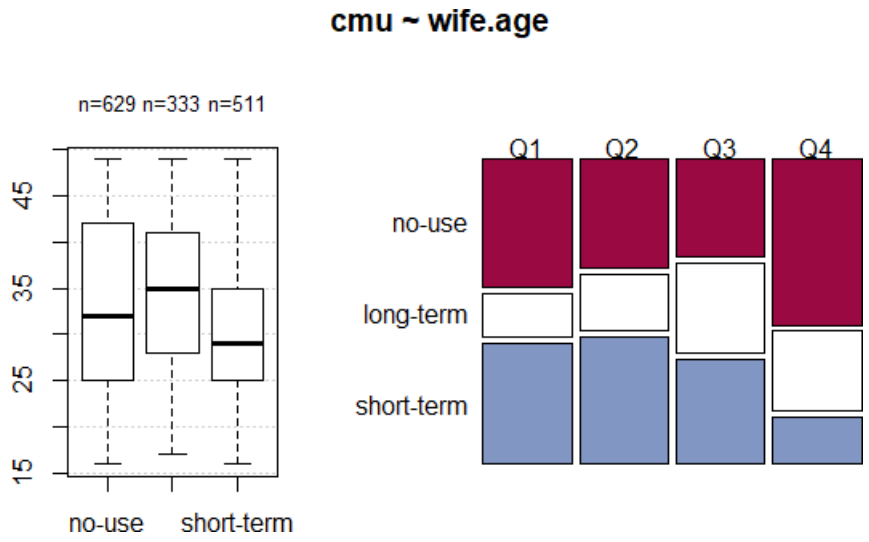
	no-use	long-term	short-term
mean	33.424	34.384	30.245
median	32.000	35.000	29.000
sd	9.124	7.455	6.944
IQR	17.000	13.000	10.000
n	629	333	511
np	42.702%	22.607%	34.691%
NAs	0	0	0
0s	0	0	0

Kruskal-Wallis rank sum test:
Kruskal-Wallis chi-squared = 59.864, df = 2, p-value = 1.002e-13

Proportions of cmu in the quantiles of wife.age:

	Q1	Q2	Q3	Q4
no-use	43.8 %	37.2 %	33.4 %	56.8 %

long-term	14.8 %	19.4 %	30.9 %	27.1 %
short-term	41.4 %	43.4 %	35.7 %	16.2 %




```
> Desc(cmu ~ wife.edu, data=cmc)
```

---

cmu ~ wife.edu

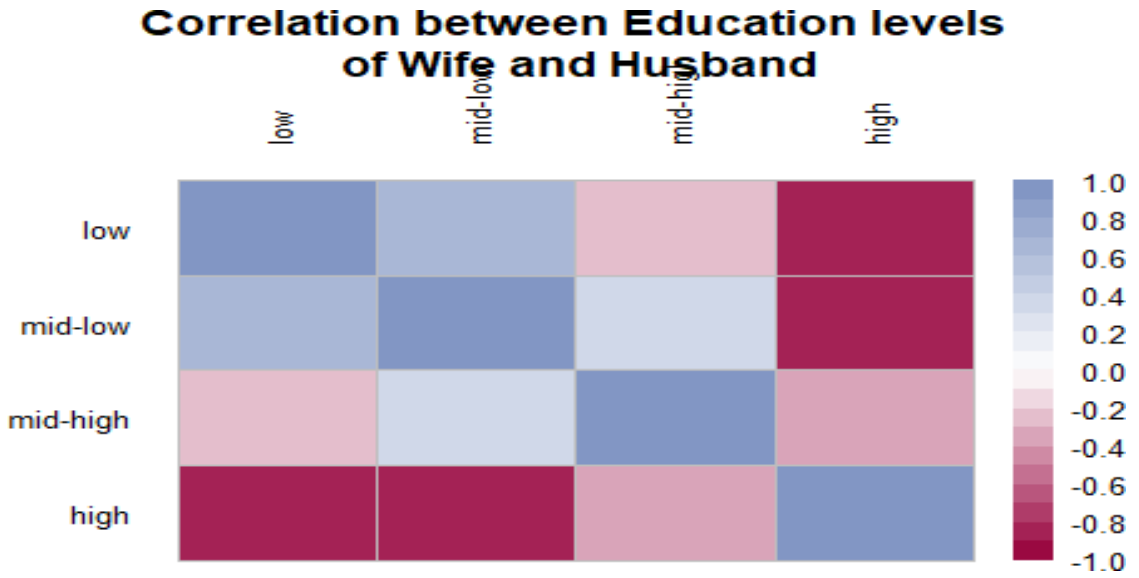
Summary:
n: 1’473, rows: 3, columns: 4

Pearson’s Chi-squared test:
X-squared = 140.46, df = 6, p-value < 2.2e-16
Likelihood Ratio:
X-squared = 144.79, df = 6, p-value < 2.2e-16
Mantel-Haenszel Chi-squared:
X-squared = 33.022, df = 1, p-value = 9.114e-09

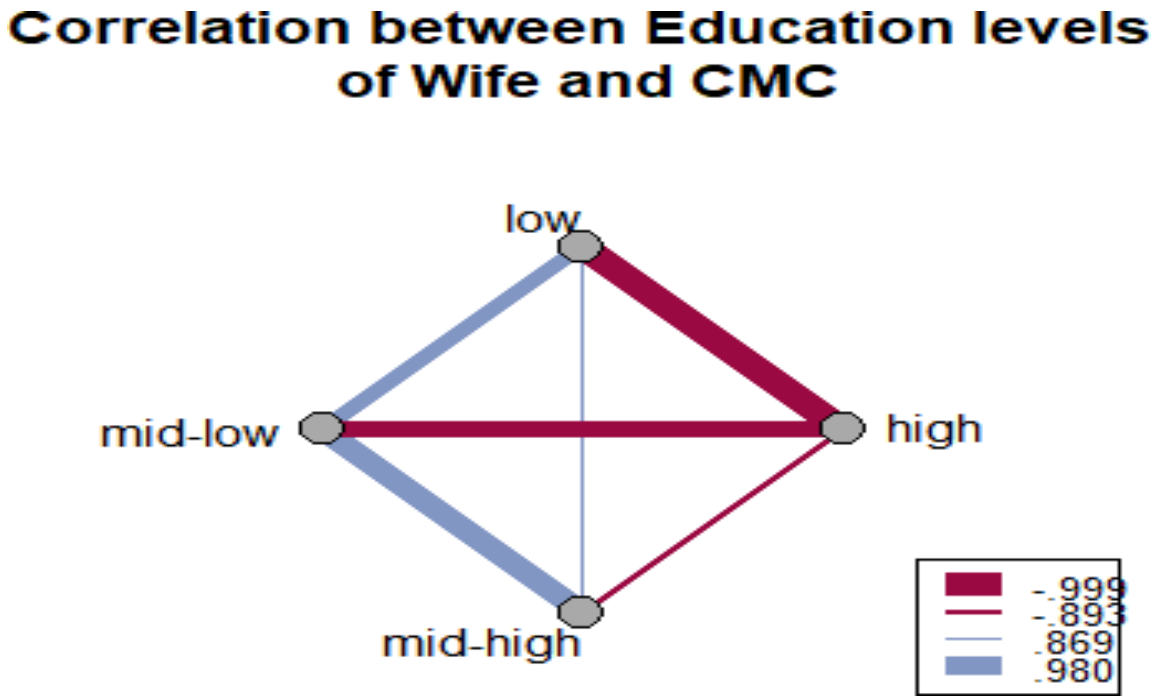
Phi-Coefficient	0.309
Contingency Coeff.	0.295

Cramer's V		0.218				
	wife.edu	low	mid-low	mid-high	high	Sum
cmu						
no-use	freq	103	176	175	175	629
	perc	7.0%	11.9%	11.9%	11.9%	42.7%
	p.row	16.4%	28.0%	27.8%	27.8%	.
	p.col	67.8%	52.7%	42.7%	30.3%	.
long-term	freq	9	37	80	207	333
	perc	0.6%	2.5%	5.4%	14.1%	22.6%
	p.row	2.7%	11.1%	24.0%	62.2%	.
	p.col	5.9%	11.1%	19.5%	35.9%	.
short-term	freq	40	121	155	195	511
	perc	2.7%	8.2%	10.5%	13.2%	34.7%
	p.row	7.8%	23.7%	30.3%	38.2%	.
	p.col	26.3%	36.2%	37.8%	33.8%	.
Sum	freq	152	334	410	577	1'473
	perc	10.3%	22.7%	27.8%	39.2%	100.0%
	p.row	.	.	.	.	.
	p.col	.	.	.	.	.

```
>with( cmc, PlotCorr( cor( table(wife.edu, husb.edu)),
        main="Correlation between Education levels\n of
        Wife and Husband"))
```




```
> with(cmc, PlotWeb( cor( table( cmu, wife.edu ) ),
       main="Correlation between Education levels\n of Wife and CMC"))
```



3. Simple Decision Tree

3.1Decision Trees - An Overview

A classification tree divides the data into smaller and smaller portions to identify patterns that can be used for prediction. The knowledge so gained is presented in the form of a logical structure that can be easily understood without any statistical knowledge. This aspect of the model assists managerial personnel for developing business strategies and process improvement.

In a classification tree the output variable is a categorical variable. A decision tree model allows us to make prediction on an output variable, based on a series of rules arranged in a tree-like structure. Each rule in a decision tree corresponds to a series of split points. These split points are referred to as **nodes**. A node may or may not have a sub-node and this is based on the class composition of the data items currently available at that node. This class composition is usually expressed in terms of purity of the node. If every data item at a node belongs to a single class, then that node is called a **pure node**. A pure node becomes **leaf node**, meaning that no further split of the data points is possible. Every node is labelled with class name and this is decided based on the highest proportion of observations of a particular class available at that node. Depending on the **error rate** that we may commit or the **bucket size** prescribed, a decision is made whether to split or not to split the node.

Based on a **split criterion**, all the items available at a node will be divided into two groups: data items meeting the criterion and data items not meeting the criterion. These groups form the two **child nodes**. Each of the child nodes in turn divides the items based on a new split criterion and this process continuous until a **stopping condition** is met. The stopping condition for a node may be:

- All(or nearly all) of the data items in the node have the same class label, or
- There are no more features available to distinguish between data items, or
- The tree has grown to a predetermined size.

All those nodes that meets the stopping condition are called leaf nodes. As the items moves from the root node to leaf nodes, they become increasingly homogeneous with respect to their class

label. If the leaf nodes are pure (all items of the same class), then that node is attached the class label of its items. If the leaf node is not a pure node, then the class label of the majority of the items currently available at that node will be assigned as the label of that node.

Decision trees use one feature at a time, depending on their importance at that particular node, to arrive at the final model. This method of partitioning the data items into groups is called **recursive partitioning**. Whatever the decision tree, learned from the data, it is depicted in the form of an inverted tree structure.

The prediction of the class label of an unseen data item begins at the root node. Depending on the split criterion in the root node, the data item will be directed either to the left node or right node. This test item is then subjected to further tests in each of the child nodes that comes in its way to leaf node. It ultimately belongs to one and only one leaf node and the leaf nodes' class label becomes the class label of the test item.

Decision trees may not be an ideal choice to model, if the data items are described by

- A large number of nominal features with many levels or
- A large number of numerical variables.

The above cases may result in an overly complex tree with a large number of decisions.

3.2 Choice of Split Variable

Having understood the working of a decision tree, we would like to know how to train a decision tree with some data. We find from the literature that there are several algorithms proposed to train a decision tree. All these algorithms answer four fundamental questions: First, for every node, including the root node, how to choose an input variable and its value to split the node into child nodes. Secondly, how to decide whether to split a node into child nodes or make that node as a leaf-node. Thirdly, how deeply to grow a tree and finally, when arrived at at leaf-node, how to decide the class label for the members of that node.There are numerous implementations for classification trees. Among them are: ID3, C4.5, and C5.0.

3.2.1 Entropy

Entropy is defined as the average number of binary digits needed to communicate information via a message as a function of the probabilities of the different symbols used. When the symbols or components of a system have equal probabilities, there will be high degree of uncertainty, but the entropy will be lower when one symbol is far likelier than the others. This has led the entropy as measure of node purity. In the binary class problem, entropy is defined by

$$Entropy = -p \log_2 p + (1 - p) \log_2 p$$

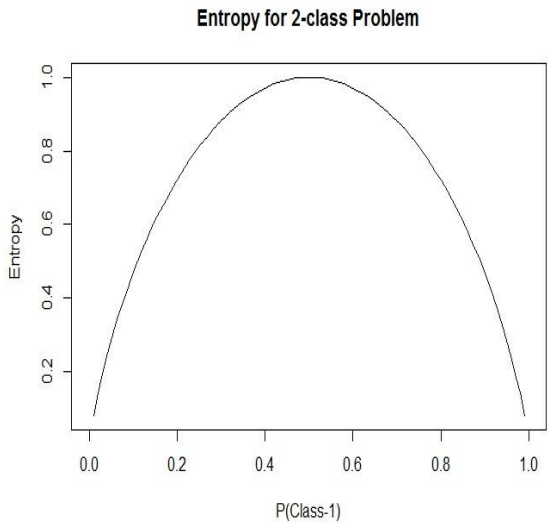


Figure 1.1: Entropy for Two-class Problem

The definition of entropy may be extended to the case where there are c classes as,

$$Entropy = -\sum_{k=1}^{c} p_k \log_2 p_k$$

Observe that, the lower the entropy, the lower the uncertainty we have about the distribution of the classes and hence, we have higher node purity. As result of this, we wish to minimize the entropy as we build a tree.

3.2.2 Information gain

The ID3 algorithm uses the weighted entropy reduction, which is also known as the information gain, as the splitting criterion. It is given by:

$$Information\ Gain \quad = \quad Entropy_{initial} - \sum_{i=1}^{p} \frac{n_i}{n} Entropy$$

This is not a good splitting criterion, since it suffers from selection bias. This algorithm tends to favour categorical variables with large number of values to continuous variable with linear range of values. To combat this, *information gain ratio* is used in an improved version of ID3 algorithm, called *C4.5*. Information gain ratio is a normalized version of information gain, normalization is with respect to a quantity known as the *spilt information value*. This in turn represents the potential increase in information that we can get just by the size of the partitions themselves. A high spilt information value occurs when we have evenly sized partitions and a low value occurs when most of the data values are concentrated in small number of the partitions. In summary, we have:

$$\text{Information Gain Ratio} \quad = \quad \frac{Information\ Gain}{Split\ Information\ value}$$

$$\text{Split information value} \quad = \quad \sum_{i=1}^{p} \frac{n_i}{n} \log_2\left(\frac{n_i}{n}\right)$$

3.2.3 Gini Index

The aim of a classification tree is to partition the data to get child nodes that are as homogeneous as possible. This partition of the data items is done based on the values of a single feature that results in groups with minimum misclassification error. As there are several features available, we are faced with the problem of choosing a feature that results in minimum misclassification error. Gini index is one measure used to decide upon the purity of a node. For a two-class problem, it is defined as

$$p_1(1 - p_1) + p_2(1 - p_2)$$

where p_1 and p_2 are class probabilities for Class-1 and Class-2, respectively. As there are only two classes by assumption, $p_1 + p_2 = 1$, and hence, the above index is equivalent to $2p_1p_2$. As p_1 increases, p_2 decreases and vice versa, the Gini index is minimized when one of the class probabilities approaches zero. In other words, as one of the class probabilities tends to zero, the node becomes pure with respect to on eof the classes. The Gini index gets maximized when $p_1 = p_2$. In this case, the node is least pure.

3.3 Data Analysis

3.3.1 Partitioning Data Set

Let us divide the data set randomly into two parts consisting of 75 percent and 25 percent. That is, we will be having two separate data frames. Call the larger part of the dataset as the training dataset and is used for building the model. The other dataset is called the test dataset and is used to evaluate the performance of the model. There are several methods using which one may evaluate the performance of a model. Cross-validation is one such method.

To ensure reproducibility of the results, use **`set.seed()`** function. Using the **`nrow()`** function, determine the number of cases in the total dataset. The value returned by the **`nrow()`** is used in the **`sample()`** function to select 75 percent of the sample cases at random, which represents our training data. The remaining cases become our test dataset.

```
set.seed(1012)

cmc.size <- nrow(cmc)

ind <- sample( cmc.size, cmc.size * 0.75)

cmc.train <- cmc[ind, ]

cmc.test <- cmc[-ind, ]
```

Let us now inspect the composition of the different choices of the contraceptives in the whole dataset, training and test datasets.

```
>nrow(cmc)

[1] 1473

>nrow(cmc.train)

[1] 1104

>nrow(cmc.test)

[1] 369
```

 # Class proportions - entire data

```
>prop.table(table(cmc$cmu))
```

```
    no-use long-term  short-term
 0.4270197 0.2260692   0.3469111
```

 # Class proportion - training data

```
>prop.table(table(cmc.train$cmu))
```

```
    no-use long-term short-term
 0.4202899 0.2264493  0.3532609
```

 # Class proportions - testing data

```
>prop.table(table(cmc.test$cmu))
```

```
    no-use   long-term  short-term
 0.4471545   0.2249322   0.3279133
```

The training dataset has 1104 cases while the test dataset has 369 cases. The proportions of classes"**non-use", "short-term"** and **"long-term"** are almost same in all the data sets.

3.3.2 Tree model

The `rpart` and `rpart.plot` packages of R programming can be used to perform a decision tree analysis. The `rpart()` function of `rpart` package is used to build decision tree. `rpart` is short form for recursive partitioning and regression trees.

The syntax of this function from the `R Help file` is found as follows:

```
rpart(formula, data, weights, subset, na.action = na.rpart,
method,model = FALSE, x = FALSE, y = TRUE, parms, control, cost,
...)
```

We know that whenever a function supports a `formula =` argument, it also supports a `data =` argument to specify the dataset to be used and this argument is optional. The formula must be specified with a response variable. `weights =` and `subset =` arguments are also optional. The `subset =` argument facilitates to use only a subset of the cases in building the model. The `na.action = na.rpart` argument will omit all those cases for which response values are missing, but includes explanatory variables with missing values. As our dataset does not have any missing values, we don't require to use this argument.

The *`method =`* argument has several values, and the only value we are interested is *`'class'`*. When *`method = 'class'`*, and the response is factor variable, then the *`rpart()`* function builds a classification model. The *`parms =`* argument takes list of optional parameters: *prior*= to specify the class `probs`, *`split =`* to specify the split function to be used; it can take two values: *`'gini' or 'information'`*, The default value is `'gini'`. Finally, the *`control =`* argument takes list of optional values whose values are usually set using the function **rpart.control()**.

```
>library(rpart)
>library(rpart.plot)
>control<- rpart.control(minsplit = 5,minbucket = round(5 / 3), ma
xdepth = 4,cp = 0)
```

Note that changing the values of **`minbucket=`** can have an impact on the choice of variables of the split. This will occur when one choice with a higher improvement result in a node with too few observations, leading to another choice being taken to meet the minimum requirements for the number of observation in a split.

While the default is to set **`minbucket=`** to be one-third of **`minsplit=,`** there is no requirement for **`minbucket=`**to be less than **`minsplit=`**.A node will always have at least **`minbucket=`**`entities`, and it will be considered for splitting if it has at least **`minsplit=`** `observation` and if on splitting each of its children has at least **`minbucket=`** `observation`.

```
rpart(formula, data=, method='class')
```

```
>fit <- rpart(cmc.train$cmu~.,data = cmc.train,method='class',
control=control)
>print(fit,digits = 4)
```

```
n= 1104

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 1104 640 no-use (0.42029 0.22645 0.35326)
 2) num.child< 0.5 77   2 no-use (0.97403 0.00000 0.02597) *
 3) num.child>=0.5 1027 638 no-use (0.37877 0.24343 0.37780)
 6) wife.edu=low,mid-low,mid-high 619 327 no-use (0.47173 0.1470
1 0.38126)
12) wife.age>=37.5 172  46 no-use (0.73256 0.12791 0.13953) *
13) wife.age< 37.5 447 235 short-term (0.37136 0.15436 0.47427)
26) num.child< 2.5 194  96 no-use (0.50515 0.14948 0.34536) *
27) num.child>=2.5 253 108 short-term (0.26877 0.15810 0.57312)*
7) wife.edu=high 408 249 long-term (0.23775 0.38971 0.37255)  14
) num.child< 2.5 188 110 short-term (0.31915 0.26596 0.41489)
28) wife.age>=41.5 11   2 no-use (0.81818 0.09091 0.09091) *
29) wife.age< 41.5 177 100 short-term (0.28814 0.27684 0.43503)*
15) num.child>=2.5 220 111 long-term (0.16818 0.49545 0.33636)
30) wife.age>=32.5 170  74 long-term (0.14706 0.56471 0.28824) *
31) wife.age< 32.5 50  25 short-term (0.24000 0.26000 0.50000) *
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node
```

The above legend indicates that every split will be provided with a node number(node), followed by split rule (split), the number of observations n at that node, the number observations that are incorrectly classified(loss), classification label for the node `(yval)` and the probability distribution of the classes in that node `(yprob)`. `rpart` follows the same order of the classes for

the probability distribution for all the nodes. A terminal node is indicated by placing a * after the class distribution. For any tree, root node is the first node and is labelled as

```
1) root 1104 640 no-use (0.42029 0.22645 0.35326)
```

All the observation in the training dataset will be available in the root node. The root node itself can be treated as a model and is usually called decision stump. It is a model that classifies every observation into the class that has the larger class size in the training dataset. The majority class for the root node `(the yval)` is **no-use**. The number 604 (technically called as loss) tells us that how many of 1104 observations will be incorrectly classified as **long-term, short-term**. Here misclassified observations are 23% and 35%.

Node-1 splitted into two sub nodes based on split value `num.child<1` as Node-2 and Node-3.Node-2 has the split expressed as `num.child<1`. At the node there are 77 observations satisfying the criterion `num.child<0.5`. This node is labelled as ***no-use***. Only 2 of 77 observations are misclassified. This represents an accuracy of 0% and 3% in predicting the observation as ***long-term and short-term***. We note that the `rpart` declared this node as an Internal Node.

```
3) num.child>=0.5 1027 638 no-use (0.37877 0.24343 0.37780)
```

At Node-3 has the split expressed as `num.child<1`. At the node there are 77 observations satisfying the criterion `num.child>=0.5`. This node is labelled as ***no-use***. Here 638 of 1027 observations are misclassified. This represents an accuracy of 24% and 38% in predicting the observation as ***long-term and short-term***.

Here node-3 splitted into two sub nodes based on split value `wife.edu=low,mid-low,mid-high` as node-6 and node-7.

```
6) wife.edu=low, mid-low, mid-high 619 327 no-use (0.47173 0.14701
0.38126)
```

Node-6 has the split expressed as `wife.edu=low,mid-low,mid-high`. At the node there are 619 observations satisfying the criterion `wife.edu=low,mid-low,mid-high`. This node is labelled as ***no-use***. Only 327 of 619 observations are misclassified. This represents an accuracy of 15% and 38% in predicting the observation as ***long-term and short-term***. Here

the node-6 is splitted into two sub nodes based on split value `wife.age>=37.5` as node-12 and node-13.

Node-12 has the split expressed as `wife.age>=37.5`. At the node there are 619 observations satisfying the criterion `wife.age>=37.5`. This node is labelled as ***no-use***. Only 46 of 172 observations are misclassified. This represents an accuracy of 13%and 14% in predicting the observation as ***long-term and short-term***. We note that the `rpart` declared this node as an Internal Node.

```
13) wife.age< 37.5 447 235 short-term (0.37136 0.15436 0.47427)
```

Node-13 has the split expressed as `wife.age>=37.5`. At the node there are 619 observations satisfying the criterion `wife.age<37.5`. This node is labelled as ***short-term***. There are 235 of 447 observations are misclassified. This represents an accuracy of 15% and 37% in predicting the observation as ***long-term and no-use***. Here the node-13 is splitted into two sub nodes based on split value `num.child<3` as node-26 and node-27.

At node-26, corresponds to the spilt value `num.child<2.5`. We have 194 observations and is labelled as **no-use.** Out of 194 observations 96 are misclassified. This corresponds to the misclassification error of 15% and 35% in predicting the observations as long-term and short term. We note that the `rpart` declared node as an Internal Node.

At Node-27 has the split expressed as `num.child<3`. At the node there are 253 observations satisfying the criterion `num.child<2.5`. This node is labelled as ***short-term***. There are 108 of 253 observations are misclassified. This represents an accuracy of 16% and 27% in predicting the observation as ***long-term and no-use***. We note that the `rpart` declared this node as an Internal Node.

```
7) wife.edu=high 408 249 long-term (0.23775 0.38971 0.37255)
```

Node-7 has the split expressed as `wife.edu=low,mid-low,mid-high`. At the node there are 408 observations satisfying the criterion wife.edu=high. This node is labelled as ***long-term***. Only 249 of 408 observations are misclassified. This represents an accuracy of 24% and 39% in predicting the observation as ***no-use and long-term***. Here the node-7 is splitted into two sub nodes based on split value `num.child<3` as node-14 and node-15.

```
14) num.child< 2.5 188 110 short-term (0.31915 0.26596 0.41489)
```

At node-14 has the split expressed as num.child<3. At the node there are 188 observations satisfying the criterion num.child<2.5. This node is labelled as ***short-term***. Only 110 of 188 observations are misclassified. This represents an accuracy of 32% and 27% in predicting the observation as ***no-use and long-term***. Here the node-14 is splitted into two sub nodes based on split value wife.age>=42 as node-28 and node-29.

Node-28 has the split expressed as wife.age>=42. At the node there are 11 observations satisfying the criterion wife.age>=41.5. This node is labelled as ***no-use***. Only 2 of 11 observations are misclassified. This represents an accuracy of 9% and 9% in predicting the observation as ***long-term and short-term***. We note that the rpart declared this node as an Internal Node.

Node-29 has the split expressed as wife.age>=42. At the node there are 177 observations satisfying the criterion wife.age>=41.5. This node is labelled as ***short-term***. Only 100 of 177 observations are misclassified. This represents an accuracy of 29% and 28% in predicting the observation as ***no-use and long-term***. We note that the rpart declared this node as an Internal Node.

```
15) num.child>=2.5 220 111 long-term (0.16818 0.49545 0.33636)
```

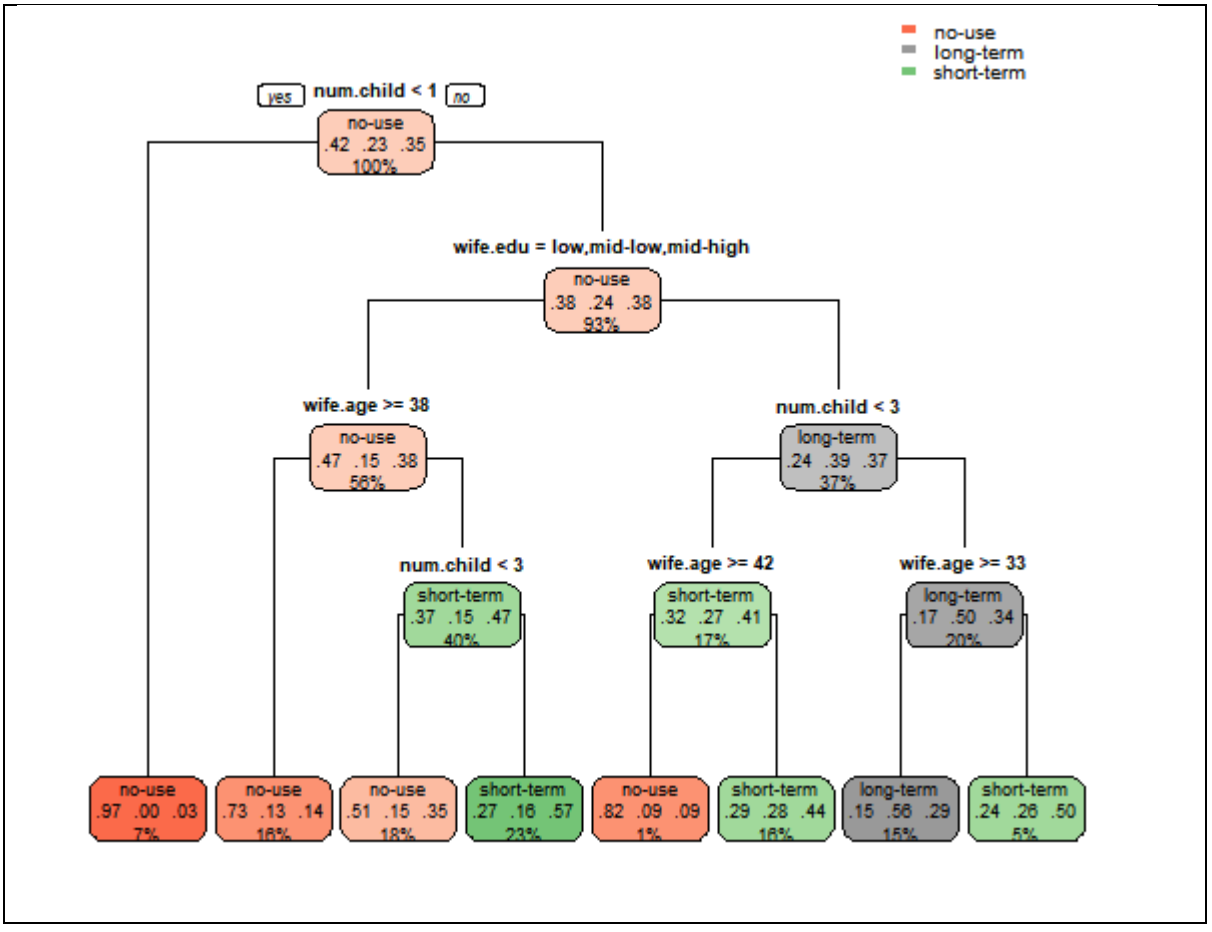
At node-15 has the split expressed as num.child<3. At the node there are 220 observations satisfying the criterion num.child>=2.5. This node is labelled as ***long-term***. Only 111 of 220 observations are misclassified. This represents an accuracy of 32% and 27% in predicting the observation as ***no-use and short-term***. Here the node-15 is splitted into two sub nodes based on split value wife.age>=33 as node-30 and node-31.

Node-30 has the split expressed as wife.age>=33. At the node there are 170 observations satisfying the criterion wife.age>=32.5. This node is labelled as ***long-term***. Only 274 of 170 observations are misclassified. This represents an accuracy of 15% and 56% in predicting the

observation as ***no-use and short-term***. We note that the `rpart` declared this node as an Internal Node.

Node-31 has the split expressed as `wife.age>=33`. At the node there are 50 observations satisfying the criterion `wife.age>=32.5`. This node is labelled as ***short-term***. Only 25 of 50 observations are misclassified. This represents an accuracy of 15% and 56% in predicting the observation as ***no-use and long-term***. We note that the `rpart` declared this node as an Internal Node.

```
>rpart.plot(fit,type=1,extra=104)
```



On basis of their variables i.e. `num.child`,`wife.edu`,`wife.age` our required decision tree is splitted.

```
>fit2 <- rpart(cmc.train$cmu~.,data = cmc.train,method='class',
```

```
parms=list(split='information'),
control=control)
```

```
>print(fit2,digits = 4)
```

```
n= 1104

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 1104 640 no-use (0.42029 0.22645 0.35326)
 2) num.child< 0.5 77   2 no-use (0.97403 0.00000 0.02597)
 4) husb.edu=mid-high,high 64   0 no-use (1.00000 0.00000 0.00000) *
 5) husb.edu=low,mid-low 13   2 no-use (0.84615 0.00000 0.15385)
10) wife.age>=21.5 10   0 no-use (1.00000 0.00000 0.00000) *
11) wife.age< 21.5 3   1 short-term (0.33333 0.00000 0.66667) *
 3) num.child>=0.5 1027 638 no-use (0.37877 0.24343 0.37780)
 6) wife.edu=low,mid-low,mid-high 619 327 no-use (0.47173 0.14708126)
12) wife.age>=37.5 172  46 no-use (0.73256 0.12791 0.13953) *
13) wife.age< 37.5 447 235 short-term (0.37136 0.15436 0.47427)
26) num.child< 2.5 194  96 no-use (0.50515 0.14948 0.34536) *
27) num.child>=2.5 253 108 short-term (0.26877 0.15810 0.57312) *
 7) wife.edu=high 408 249 long-term (0.23775 0.38971 0.37255)
14) num.child< 2.5 188 110 short-term (0.31915 0.26596 0.41489)
28) wife.age>=44 8   1 no-use (0.87500 0.12500 0.00000) *
29) wife.age< 44 180 102 short-term (0.29444 0.27222 0.43333) *
15) num.child>=2.5 220 111 long-term (0.16818 0.49545 0.33636) *
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node
```

The above legend indicates that every split will be provided with a node number(node), followed by split rule (split), the number of observations n at that node, the number observations that are incorrectly classified(loss), classification label for the node `(yval)` and the probability distribution of the classes in that node(`yprob`). `rpart` follows the same order of the classes for the probability distribution for all the nodes. A terminal node is indicated by placing a * after the class distribution. For any tree, root node is the first node and is labelled as

```
1) root 1104 640 no-use (0.42029 0.22645 0.35326)
```

All the observation in the training dataset will be available in the root node. The root node itself can be treated as a model and is usually called decision stump. It is a model that classifies every observation into the class that has the larger class size in the training dataset. The majority class for the root node (the `yval`) is ***no-use***. The number 640 (technically called as loss) tells us that how many of 1104 observations will be incorrectly classified as ***long-term and short-term***. Here misclassified observations are 23% and 35%. Node-1 splitted into two nodes based on split value `num.child<1` as Node-2 and Node-3.

```
2) num.child< 0.5 77   2 no-use (0.97403 0.00000 0.02597)
```

Node-2 has the split expressed as `num.child<1`. At the node there are 77 observations satisfying the criterion `num.child<0.5`. This node is labelled as ***no-use***. Only 2 of 77 observations are misclassified. This represents an accuracy of 0% and 3% in predicting the observation as ***long-term and short-term***. Here the node-2 is splitted into two sub nodes based on split value `husb.edu=mid-high,high` as node-4 and node-5.

Node-4 has the split expressed as `husb.edu=mid-high,high`. At the node there are 64 observations satisfying the criterion `husb.edu=mid-high,high`. This node is labelled as ***no-use***. Only 0 of 64 observations are misclassified. This represents an accuracy of 0% and 0% in predicting the observation as ***long-term and short-term***. We note that the `rpart` declared this node as an Internal Node.

```
5) husb.edu=low,mid-low 13   2 no-use (0.84615 0.00000 0.15385)
```

Node-5 has the split expressed as `husb.edu=mid-high,high`. At the node there are 13 observations satisfying the criterion `husb.edu=mid-high,high`. This node is labelled as ***no-use***. Only 2 of 13 observations are misclassified. This represents an accuracy of 0% and 15% in

predicting the observation as ***long-term and short-term***. Here the node-5 is splitted into two sub nodes based on split value `wife.age>=22` as node-10 and node-11.

Node-10 has the split expressed as `wife.age>=22`. At the node there are 10 observations satisfying the criterion `wife.age>=21.5`. This node is labelled as ***no-use***. Only 0 of 10 observations are misclassified. This represents an accuracy of 0% and 0% in predicting the observation as **long-term and short-term**. We note that the `rpart` declared this node as an Internal Node.

Node-11 has the split expressed as `wife.age>=22`. At the node there are 3 observations satisfying the criterion `wife.age>=21.5`. This node is labelled as ***short-term***. Only 0 of 10 observations are misclassified. This represents an accuracy of 0% and 0% in predicting the observation as ***no-use and long-term***. We note that the `rpart` declared this node as an Internal Node.

```
3) num.child>=0.5 1027 638 no-use (0.37877 0.24343 0.37780)
```

At Node-3 has the split expressed as `num.child<1`. At the node there are 1027 observations satisfying the criterion `num.child<0.5`. This node is labelled as ***no-use***. Only 638 of 1027 observations are misclassified. This represents an accuracy of 24% and 38% in predicting the observation as ***long-term and short-term***. Here the node-3 is splitted into two sub nodes based on split value `wife.edu=low, mid-low, mid-high` as node-6 and node-7.

```
6) wife.edu=low, mid-low, mid-high 619 327 no-use (0.47173 0.14701
0.38126)
```

Node-6 has the split expressed as `wife.edu=low, mid-low, mid-high`. At the node there are 172 observations satisfying the criterion `wife.edu=low, mid-low, mid-high`. This node is labelled as ***no-use***. Only 46 of 172 observations are misclassified. This represents an accuracy of 15% and 38% in predicting the observation as ***long-term and short-term***. Here the node-6 is splitted into two sub nodes based on split value `wife.age>=38` as node-12 and node-13.

Node-12 has the split expressed as `wife.age>=38`. At the node there are 172 observations satisfying the criterion `wife.age>=37.5`. This node is labelled as ***no-use***. Only 46 of 172 observations are misclassified. This represents an accuracy of 13% and 14% in predicting the

observation as ***long-term and short-term***. We note that the `rpart` declared this node as an Internal Node.

13) wife.age< 37.5 447 235 short-term (0.37136 0.15436 0.47427)

At Node-13 has the split expressed as `wife.age>=38`. At the node there are 447 observations satisfying the criterion `wife.age>=37.5`. This node is labelled as ***short-term***. Only 235 of 447 observations are misclassified. This represents an accuracy of 15% and 47% in predicting the observation as ***no-use and long-term***. Here the node-13 is splitted into two sub nodes based on split value `num.child<3` as node-26 and node-27.

Node-26 has the split expressed as `num.child<3`. At the node there are 194 observations satisfying the criterion `num.child<2.5`. This node is labelled as ***no-use***. Only 96 of 194 observations are misclassified. This represents an accuracy of 15% and 35% in predicting the observation as ***long-term and short-term***. We note that the `rpart` declared this node as an Internal Node.

Node-27 has the split expressed as `num.child<3`. At the node there are 253 observations satisfying the criterion `num.child>=2.5`. This node is labelled as ***short-term***. Only 108 of 253 observations are misclassified. This represents an accuracy of 16% and 57% in predicting the observation as ***no-use and long-term***. We note that the `rpart` declared this node as an Internal Node.

7) wife.edu=high 408 249 long-term (0.23775 0.38971 0.37255)

At Node-7 has the split expressed as `wife.edu=low,mid-low,mid-high`. At the node there are 408 observations satisfying the criterion `wife.edu=high`. This node is labelled as ***long-term***. Only 46 of 172 observations are misclassified. This represents an accuracy of 15% and 38% in predicting the observation as ***no-use and short-term***. Here the node-7 is splitted into two sub nodes based on split value `num.child<3` as node-14 and node-15.

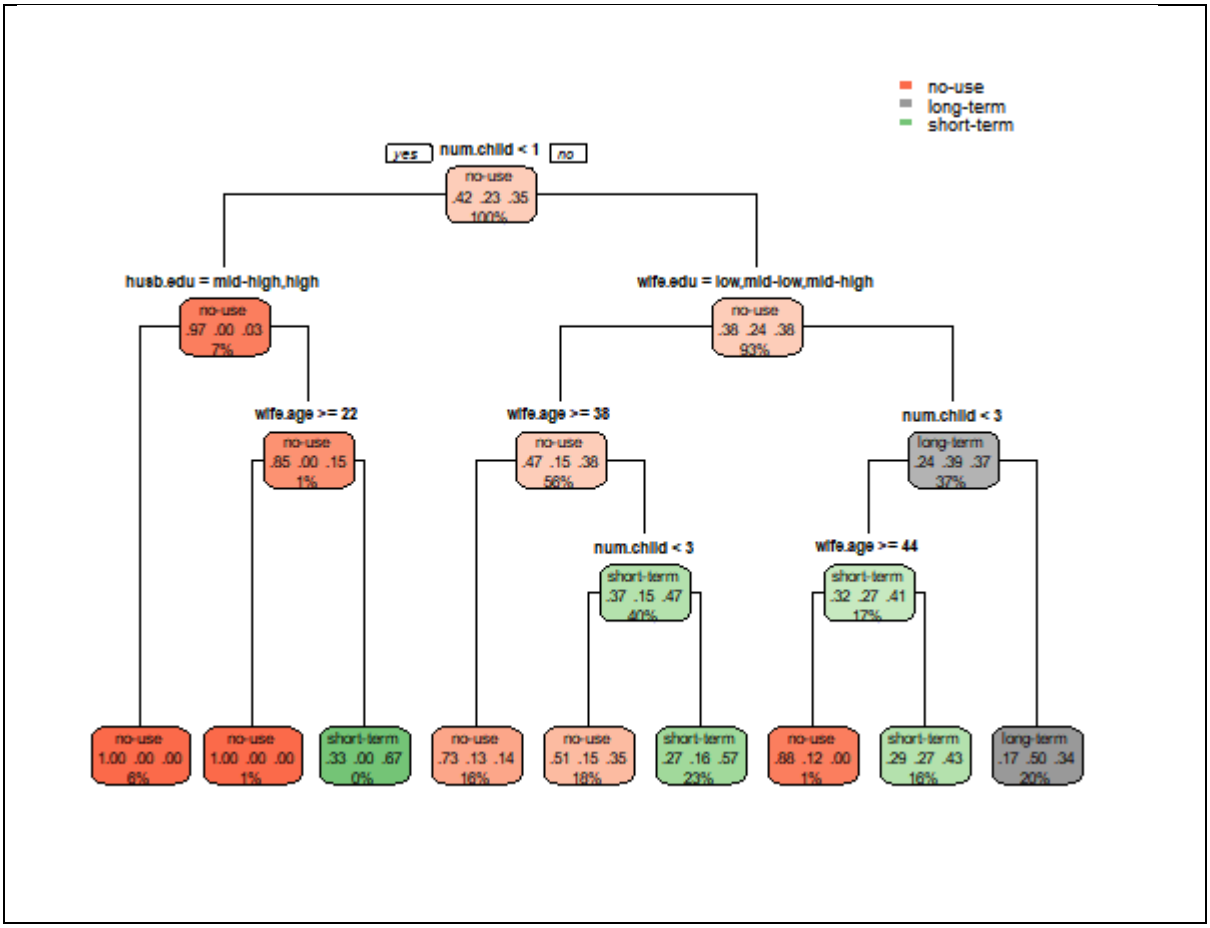
14) `num.child< 2.5 188 110 short-term (0.31915 0.26596 0.41489)`

Node-14 has the split expressed as `num.child<3`. At the node there are 172 observations satisfying the criterion `num.child<2.5`. This node is labelled as ***short-term***. Only 46 of 172 observations are misclassified. This represents an accuracy of 15% and 38% in predicting the observation as ***no-use and long-term***. Here the node-14 is splitted into two sub nodes based on split value `wife.age>=44` as node-28 and node-29.

Node-28 has the split expressed as `wife.age>=44`. At the node there are 8 observations satisfying the criterion `wife.age>=44`. This node is labelled as ***no-use***. Only 1 of 8 observations are misclassified. This represents an accuracy of 13% and 0% in predicting the observation as **long-term** and **short-term**. We note that the `rpart` declared this node as an Internal Node.

Node-29 has the split expressed as `wife.age>=44`. At the node there are 180 observations satisfying the criterion `wife.age<44`. This node is labelled as ***short-term***. Only 102 of 180 observations are misclassified. This represents an accuracy of 27% and 43% in predicting the observation as ***no-use and long-term***. We note that the `rpart` declared this node as an Internal Node.

At Node-15 has the split expressed as `num.child<3`. At the node there are 220 observations satisfying the criterion `num.child>=2.5`. This node is labelled as ***short-term***. Only 111 of 220 observations are misclassified. This represents an accuracy of 50% and 34% in predicting the observation as ***no-use and long-term***. We note that the `rpart` declared this node as an Internal Node.



On basics of these variables i.e. `husb.edu,num.child,wife.age,wife.edu` our required decision tree is splitted

Testing data by using `rpart()` and `rpart.plot()`

```
>test.fit<- rpart(cmc.test$cmu~.,data = cmc.test,
method='class',control=control)

>print(test.fit,digits = 4)
```

```
n= 369

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 369 204 no-use (0.44715 0.22493 0.32791)
 2) num.child< 0.5 20   0 no-use (1.00000 0.00000 0.00000) *
 3) num.child>=0.5 349 204 no-use (0.41547 0.23782 0.34670)
 6) wife.age>=37.5 111  55 no-use (0.50450 0.33333 0.16216)
 12) num.child< 2.5 23   3 no-use (0.86957 0.04348 0.08696) *
 13) num.child>=2.5 88  52 no-use (0.40909 0.40909 0.18182)
 26) wife.edu=low,mid-low,mid-high 49  19 no-use (0.61224 0.2449
0 0.14286) *
 27) wife.edu=high 39  15 long-term (0.15385 0.61538 0.23077) *
  7) wife.age< 37.5 238 135 short-term (0.37395 0.19328 0.43277)
 14) SOL.index< 1.5 23   6 no-use (0.73913 0.08696 0.17391) *
 15) SOL.index>=1.5 215 116 short-term (0.33488 0.20465 0.46047)
 30) husb.job=1 54  34 no-use (0.37037 0.33333 0.29630) *
 31) husb.job=2,3,4 161  78 short-term (0.32298 0.16149 0.51553)
*
```

```
  node), split, n, loss, yval, (yprob)
  * denotes terminal node
```

The above legend indicates that every split will be provided with a node number(node), followed by split rule (split), the number of observations n at that node, the number observations that are incorrectly classified(loss), classification label for the node `(yval)` and the probability distribution of the classes in that node `(yprob)`. `rpart` follows the same order of the classes for the probability distribution for all the nodes. A terminal node is indicated by placing a * after the class distribution. For any tree, root node is the first node and is labelled as

```
1) root 369 204 no-use (0.44715 0.22493 0.32791)
```

All the observation in the testing dataset will be available in the root node. The root node itself can be treated as a model and is usually called decision stump. It is a model that classifies every observation into the class that has the larger class size in the training dataset. The majority class for the root node (the `yval`) is ***no-use***. The number 204 (technically called as loss) tells us that how many of 369 observations will be incorrectly classified as ***long-term and short-term***. Here misclassified observations are 22% and 33%.

Node-2 has the split expressed as `num.child<1`. At the node there are 20 observations satisfying the criterion `num.child<0.5`. This node is labelled as `no-use`. Only 0 of 20 observations are misclassified. This represents an accuracy of 0%and 0% in predicting the observation as ***long-term and short-term***. We note that the `rpart` declared this node as an Internal Node.

```
3) num.child>=0.5 349 204 no-use (0.41547 0.23782 0.34670)
```

Node-3 has the split expressed as `num.child>1`. At the node there are 349 observations satisfying the criterion `num.child>=0.5`. This node is labelled as no-use. 204 of 349 observations are misclassified. This represents an accuracy of 24%and 35% in predicting the observation as ***long-term and short-term***. Node-3 was splited into sub-nodes 6 and 7.

```
6) wife.age>=37.5 111  55 no-use (0.50450 0.33333 0.16216)
```

Coming to Node-6, it has the split expressed as `wife.age>=38`.At that node there are 111 observations satisfying the criterion `wife.age>=37.5`. This node is labeled as no-use. 55 out of 111 are misclassified. This represents an accuracy of 33% and 16% in predicting the observation as ***long-term and short-term***. Node-6 was again splited into sub nodes 12 and 13.

Node-12, it has the split expressed as num.child<3. At that node there are 23 observations satisfying the criterion num.child<2.5. This node is labeled as no-use.3 out of 23 are misclassified.

This represents an accuracy of 4% and 9% in predicting the observation as ***long-term and short-term***. We note that the `rpart` declared this node as an Internal Node.

```
13) num.child>=2.5 88  52 no-use (0.40909 0.40909 0.18182)
```

Coming to Node-13, it has the split expressed as `num.child>3`.At that node there are 88 observations satisfying the criterion `num.child>=2.5`. This node is labeled as no-use. 52 out of 88 are misclassified. This represents an accuracy of 41% and 18% in predicting the observation as ***long-term*** and ***short-term***. Node-13 was again splited into sub nodes 26 and 27.

Node-26, it has the split expressed as `wife.edu=low,mid-low,mid-high`. At that node there are 49 observations satisfying the criterion `wife.edu=low,mid-low,mid-high`. This node is labeled as no-use.19 out of 49 are misclassified. This represents an accuracy of 24% and 14% in predicting the observation as ***long-term and short-term***. We note that the `rpart` declared this node as an Internal Node.

Node-27, it has the split expressed as `wife.edu=low,mid-low,mid-high`. At that node there are 39 observations satisfying the criterion `wife.edu=low,mid-low,mid-high`. This node is labelled as no-use.15 out of 39 are misclassified. This represents an accuracy of 62% and 23% in predicting the observation as ***no-use and short-term***. We note that the `rpart` declared this node as an Internal Node.

```
7) wife.age< 37.5 238 135 short-term (0.37395 0.19328 0.43277)
```

Coming to Node-7, it has the split expressed as `wife.age<=38`.At that node there are 238 observations satisfying the criterion `wife.age<37.5`. This node is labeled as no-use. 135 out of 238 are misclassified. This represents an accuracy of 19% and 43% in predicting the observation as ***no-use and long-term***. Node-6 was again splited into sub nodes 14 and 15.

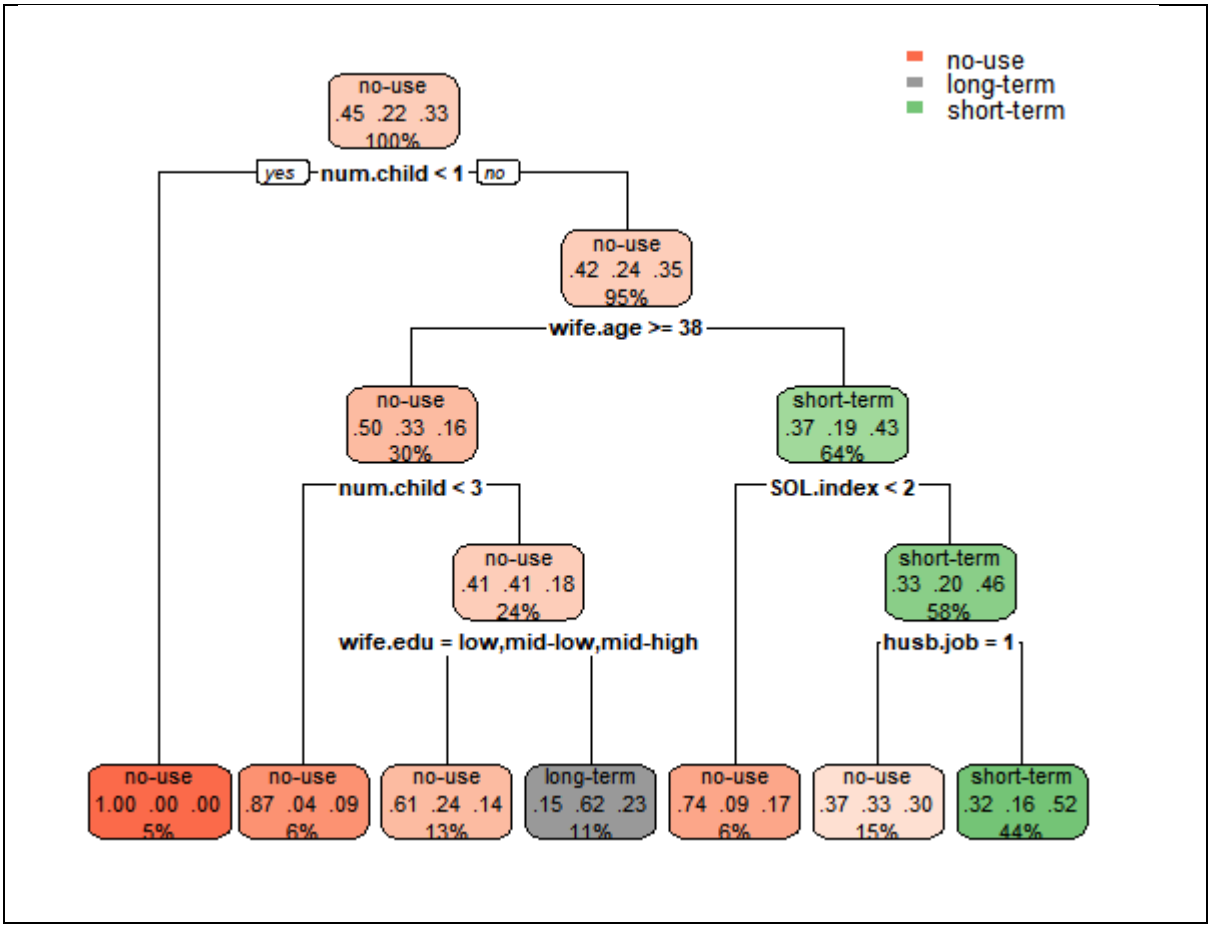
Node-14, it has the split expressed as `SOL.index<2`. At that node there are 23 observations satisfying the criterion `SOL.index<1.5`. This node is labeled as no-use.6 out of 23 are misclassified. This represents an accuracy of 9% and 17% in predicting the observation as ***long-term and short-term***. We note that the `rpart` declared this node as an Internal Node.

15) SOL.index>=1.5 215 116 short-term (0.33488 0.20465 0.46047)

Coming to Node-15, it has the split expressed as SOL.index>2. At that node there are 215 observations satisfying the criterion SOL.index>=1.5. This node is labeled as short-term. 116 out of 215 are misclassified. This represents an accuracy of 20% and 46% in predicting the observation as ***no-use and long-term***. Node-6 was again splited into sub nodes 30 and 31.

Node-30, it has the split expressed as hus.job=1. At that node there are 54 observations satisfying the criterion hus.job=1. This node is labelled as no-use.34 out of 54 are misclassified. This represents an accuracy of 33% and 30% in predicting the observation as ***long-term and short-term***. We note that the rpart declared this node as an Internal Node.

Node-31, it has the split expressed as hus.job=1. At that node there are 161 observations satisfying the criterion hus.job=1. This node is labeled as no-use.54 out of 161 are misclassified. This represents an accuracy of 16% and 52% in predicting the observation as ***no-use and long-term***. We note that the rpart declared this node as an Internal Node.



On basics of these variables i.e. `husb.job , num.child, SOL.index, wife.age, wife.edu` our required decision tree is spitted.

3.3.3 Model Performance

To know how to well the model is performed. We create a set of predictions. Based on our model using the `predict()` function. Comparing our predictions against the actual labels of the test data, we create a confusion matrix that we then use to compute our model's predictive accuracy.

Predict is a generic function for predictions from the results of various model fitting functions. The function invokes particular *methods* which depend on the class of the first argument

```
>predict.gen<- predict(fit,cmc.test,type='class')

>predict.res <- table(cmc.test$cmu,predict.gen)

>predict.res
```

```
predict.gen
             no-use long-term short-term
  no-use         102        14         49
  long-term       19        29         35
  short-term      35        17         69
```

```
>diag(predict.res)/sum(predict.res)

    no-use         long-term       short-term

  0.27642276     0.07859079      0.18699187
```

A confusion matrix is a table that is often used to describe the performance of a classification mode l on a set of test data for which the true values are known. The confusion matrix itself is relatively s imple to understand, but the related terminology can be confusing.You can compute the accuracy te st from the confusion matrix

Accuracy by outcome class

```
>accuracy.model<-sum(diag(predict.res))/sum(predict.
res)

>accuracy.model

   [1] 0.5420054
```

`sum(diag(predict.res))` represents the sum of the diagonals and `sum(predict.res)` represents the sum of the matrix.

The predictive accuracy of the model is 54 percent.

```
>printcp(fit)

Classification tree:
rpart(formula = cmc.train$cmu ~ ., data = cmc.train, method = "class
",
    control = control)

Variables actually used in tree construction:
[1] num.childwife.age  wife.edu

Root node error: 640/1104 = 0.57971

n= 1104

        CP nsplitrel error  xerrorxstd
1 0.048438      0   1.00000 1.00000 0.025626
2 0.043750      4   0.78281 0.86406 0.025958
3 0.018750      5   0.73906 0.75469 0.025755
4 0.012500      6   0.72031 0.76250 0.025783
5 0.000000      7   0.70781 0.74375 0.025711
```

The performance of the tree is calculated by the `printcp()` function, which we can go ahead and prune the tree. `printcp()` function gives cross-validation estimates of misclassification error(xerror), standard errors (`xstd`) of those estimates and the training estimates(error).

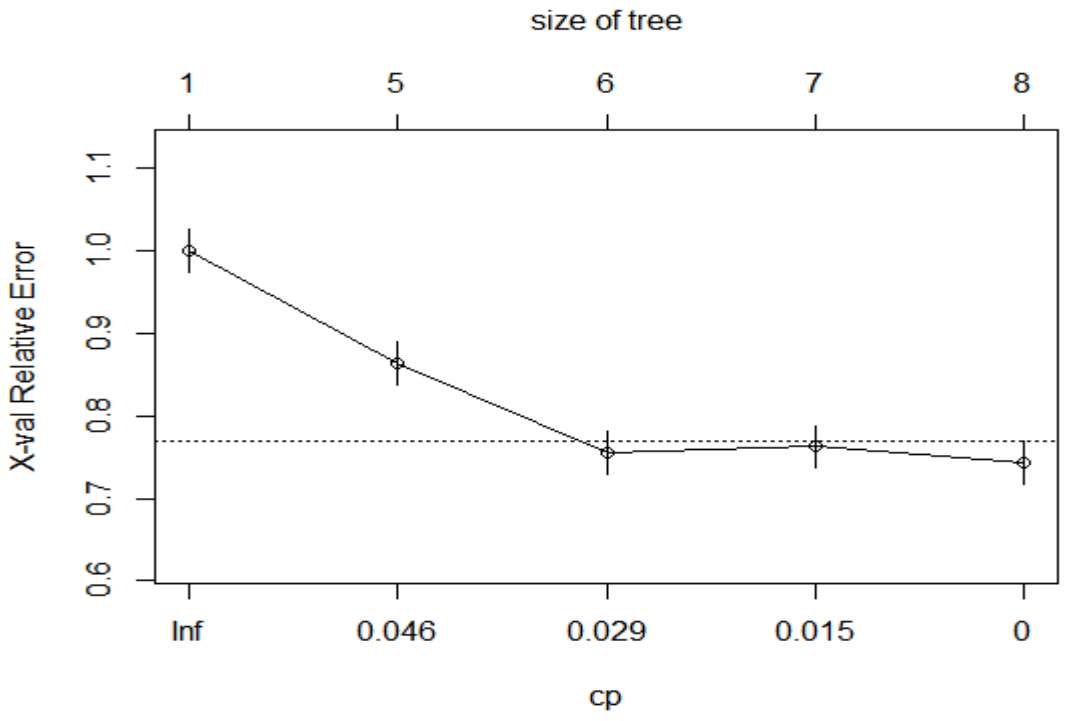
The most important column in the above table is the cross-validated error, which is the **`xerror`** column of the table. Recall that, in the root node, we have 1104 observations, out of which there are 640 long-term and short-terms. The decision tree algorithm named this node as long-term and short-term because their size 640, outnumber 461 are the no-use. If we were to classify every observation as long-term and short-term, the error over the entire dataset is 0.57971 or 57%. Treating this as baseline error (i.e. as 1.00), the above table shows the relative reduction in the error (and the cross-validation-based-error) as we build the tree. From the second line we observe that after the split of the dataset, by building a model we have reduced the cross-validation based error to 86% of the

original error. The error on the training dataset is reduced to 78%. From this we note that the cross-validation error is usually reduced slowly than the error on the training dataset (error column of the table). After the fourth split, the reduction in the cross-validation based error is very marginal i.e., from76% to 74% whereas the error on the training dataset is reduced from 72% to 70%. To conclude, building the decision tree model we are able to reduce the cross-validation error from 100% to 74%.

```
>plotcp(fit)
```

The complexity parameter is used to control the size of the decision tree and to select an optimal tree. The complexity parameter controls the process of pruning a decision tree.

By setting the parameter values of complexity, Min Split and Min Bucket all to 0, we can build a full decision tree. However, without pruning, a decision tree model can over fit the training data and hence, may not perform well on unseen data. A general rule is that, the more the complex the models is, the less it will generalize. By setting the CP parameter to an appropriate value we can avoid splits that add little value to the model. The `plotcp()` command is useful in visualizing the progression of CP values. By default the Rattle package sets this value to 0.0100. For our model, we have obtained the CP plot for the pruned tree as shown in figure.



4.1 C5.0 decision tree

There are quite a number of implementation for decision tree. One of the most popular implementation is C5.0 algorithm. This algorithm was due to Quinlan, which is an improved version of his earlier algorithm for decision tree called C4.5. The C4.5 algorithm itself is an improvement over the ID3 algorithm which is also due to Quinlan. In R this algorithm is available through the package C5.0.

This algorithm has become industry standard for developing decision trees. It can handle both small and large datasets. The learning process is highly automatic and handle numeric as well as nominal variables. It can handles the missing data efficiently. The results of C5.0 algorithm are easy to interpret.

Before we actually use the C5.0 () function of C50 package we must install the package using the command

```
>install.package("C50")
```

This can achieved using the following command.

```
>library(C50)
```

We now load the Contraceptive method Choice data set.

```
setwd("F:/new project")

  cmc=read.csv(file="data.txt",sep=",",header = FALSE)

   colnames(cmc) <-c("wife.age",
                     "wife.edu",
                     "husb.edu",
                     "num.child","islam",
                     "wife.working",
                     "husb.job",
                     "SOL.index",
                     "media.exposure",
                      "cmu")
```

And the above dataset column names should be added. And we convert the education variables for both wife and husband as well as `Sol.index and husband.job` into factors.

```
cmc$wife.edu <- as.factor(cmc$wife.edu)

cmc$husb.edu <- as.factor(cmc$husb.edu)

cmc$SOL.index <- as.factor(cmc$SOL.index)

cmc$husb.job <- as.factor(cmc$husb.job)
```

The response variable should also be converted into a categorical variables having three levels whose labels being " no-use"," long-term" " short-term"

```
cmc$cmu <- factor(cmc$cmu, labels = c("no-use", "long-term", "short-term"))
```

We done all the necessary transformation to the dataset. The structure of the data by using command `str()`

```
> str(cmc)
'data.frame': 1473 obs. of  10 variables:
 $ wife.age      : int  24 45 43 42 36 19 38 21 27 45 ...
 $ wife.edu      : Factor w/ 4 levels "1","2","3","4": 2 1 2 3 3 4
2 3 2 1 ...
 $ husb.edu      : Factor w/ 4 levels "1","2","3","4": 3 3 3 2 3 4
3 3 3 1 ...
 $ num.child     : int  3 10 7 9 8 0 6 1 3 8 ...
 $ islam         : int  1 1 1 1 1 1 1 1 1 1 ...
 $ wife.working  : int  1 1 1 1 1 1 1 0 1 1 ...
 $ husb.job      : Factor w/ 4 levels "1","2","3","4": 2 3 3 3 3 3
3 3 3 2 ...
 $ SOL.index     : Factor w/ 4 levels "1","2","3","4": 3 4 4 3 2 3
2 2 4 2 ...
 $ media.exposure: int  0 0 0 0 0 0 0 0 0 1 ...
 $ cmu           : Factor w/ 3 levels "no-use","long-term",..: 1 1
1 1 1 1 1 1 1 1 …
```

In order to the dataset contain 1473 observations. And this dataset converting into training data and testing data and we set the random seed 1012.

```
set.seed(1012)

 cmc.size <- nrow(cmc)

ind <- sample( cmc.size, cmc.size * 0.75)

cmc.train <- cmc[ ind, ]

cmc.test <- cmc[ -ind, ]
```

Using `nrow()` function determine the number of cases in the total dataset. The value returned by the `nrow()` is used in the `sample()` function to select 75 percent of the sample cases at random, which represent our training data. The remaining cases become our test dataset.

#class proportions - entire data

```
> table( cmc$cmu )

    no-use  long-term short-term
       629        333        511

> prop.table( table( cmc$cmu ) )

    no-use   long-term short-term
 0.4270197    0.2260692  0.3469111
```

From the above outputs, out of 1473 entire data values. 629 observations are no-use, 333 observations are long-term and 511 observations are short-term. They are approximately in the ratio s 43%, 22% and 35%.

 # Class proportions - training data

```
> table( cmc.train$cmu )

    no-use  long-term short-term
       464        250        390
> prop.table( table( cmc.train$cmu ) )

    no-use  long-term short-term
 0.4202899  0.2264493  0.3532609
```

Out of 1104 training data values, 464 observations are of type no-use, 250 observations are of type long-term and 390 observations are of type short-term. They are approximately in ratios 42%, 23% and 35%. The remaining data values will be considered are test sample.

 # Class proportions - test data

```
> table( cmc.test$cmu )

    no-use  long-term short-term
       165         83        121
> prop.table( table( cmc.test$cmu ) )

    no-use  long-term short-term
 0.4471545  0.2249322  0.3279133
```

4.2 Training the Model

We would like to build a decision tree using C5.0 algorithm. To do so, we have to use the `C5.0( )` function and `C5.0control( )` function of the C50 package. A simple syntax of the `C5.0( )` function is given by:

```
C5.0(x, y, trials = 1, rules = FALSE . . .)
```

Where

x	a data frame or matrix of predictors
y	a factor vector with 2 or more levels
trails	an integer specifying the number of boosting iterations. A value of one indicates that a single model is used
rules	A logical: should the tree be decomposed into a rule-based model?

Syntax of the C5.0control( ) function is given by:

```
C5.0Control( subset = TRUE, bands = 0, winnow = FALSE,noGlobalPruning = FALSE,CF = 0.25, minCases = 2,fuzzyThreshold = FALSE,sample = 0,seed = sample.int(4096, size = 1) - 1L,earlyStopping = TRUE,label = "outcome")
```

Where

subset	A logical: should the model evaluate groups of discrete predictors for splits? Note: the C5.0 command line version defaults this parameter to FALSE, meaning no attempted groupings will be evaluated during the tree growing stage.
bands	An integer between 2 and 1000. If TRUE, the model orders the rules by their effect on the error rate and groups the rules into the specified number of bands. This modifies the output so that the effect on the error rate can be seen for the groups of rules within a band. If this options is selected

	and rules = FALSE, a warning is issued and rules is changed to TRUE.
winnow	A logical: should predictor winnowing (i.e. feature selection) be used?
noGlobalPruning	A logical to toggle whether the final, global pruning step to simplify the tree.
CF	A number in (0, 1) for the confidence factor.
minCases	An integer for the smallest number of samples that must be put in at least two of the splits.
fuzzyThreshold	A logical toggle to evaluate possible advanced splits of the data. See Quinlan (1993) for details and examples.
sample	A value between (0, .999) that specifies the random proportion of the data should be used to train the model. By default, all the samples are used for model training. Samples not used for training are used to evaluate the accuracy of the model in the printed output.
seed	An integer for the random number seed within the C code.
earlyStopping	A logical to toggle whether the internal method for stopping boosting should be used.
label	A character label for the outcome used in the output.

For a complete syntax of the function, R Help may be consulted. This function will return a C5.0 model object which can in turn be used for predictions. Let us first create `C5.0control ( )` model. We would like to express the function. Syntax in a readable form as follows:

```
>C5.0Control( minCases = 5,)
```

The command for various parameters that control aspects of the C5.0 fit.

4.3 Particular tree models

4.3.1 Decision tree by using the single variable

```
> tree.1 <- C5.0(cmu ~ num.child, data=cmc.train)
```

```
> tree.1
```

```
Call:
C5.0.formula(formula = cmu ~ num.child, data = cmc.train)

Classification Tree
Number of samples: 1104
Number of predictors: 1

Tree size: 4

Non-standard options: attempt to group attributes
```

The above output reminds us the function call made, whether it is the classification tree or regression tree that was built, the number of examples used in learning and the predictor variable (`num.child`) used in building the classification tree it also has a mention about the size of the tree. This indicates how much depth the tree was grown. As per the output, it is 4 decisions deep. To know the various decisions made by the tree to arrive at the final mode, let us call the `summary ( )` function on the model object. The following is the output generated as result of the `summary ( )` function call.

```
> summary(tree.1)
```

```
Call:
C5.0.formula(formula = cmu ~ num.child, data = cmc.train)



C5.0 [Release 2.07 GPL Edition]    Mon Jun 01 14:35:31 2020
```

```
--------------------------------

Class specified by attribute `outcome'

Read 1104 cases (2 attributes) from undefined.data

Decision tree:

num.child <= 0: no-use (77/2)
num.child > 0:
:...num.child <= 1: no-use (208/101)
    num.child > 1:
    :...num.child <= 7: short-term (749/444)
        num.child > 7: no-use (70/31)


Evaluation on training data (1104 cases):

           Decision Tree
         ----------------
         Size      Errors

            4     578(52.4%)   <<


          (a)   (b)   (c)    <-classified as
         ----  ----  ----
          221         243    (a): class no-use
           49         201    (b): class long-term
           85         305    (c): class short-term

       Attribute usage:

       100.00%       num.child

Time: 0.0 secs
```

Interpreting the Output

As the size of the tree is very small, the C5.0 algorithm outputs all the decision made to grow the tree. The lines under the heading Decision tree: can be interpreted as follows:

1. If `num.child<=0`, then classify the people as no-use. Observe that there are two numbers mentioned within a pair of parenthesis. The upper number indicates the size of the node or the number of examples satisfying the condition. The class label indicates which class type observations are more in number at that node. Note that this node is labelled as no-use. This means that most of the observations (or all) are of the type no-use. How many observations of the other class, short-term, are there at this node? The lower integer in the parenthesis indicates the size of the short-term meeting this condition. In other words, 2 of the short-term were classified as no-use at this node.
2. Otherwise, `num.child>0`.
3. If `num.child>0 and num.child<=1`, then classify the people as no-use. There are 208 Peoples in the Dataset meeting this condition. The node is labelled as no-use. Of the 208 peoples, 101 people were misclassified.
4. Otherwise `num.child>1`.
5. If `num.child>1 and num.child<=7`, then classify the people as short-term. There are 749 customers in the dataset meeting this condition. The node is labelled as short-term. Of the 749 peoples, 444 were misclassified.
6. If `num.child>1 and num.chld>7`, then classify the people as no-use. There are 70 peoples in the dataset meeting this condition. The node is labelled as no-use. Of the 70 peoples, 31 were misclassified.

The summary () function also indicates the performance of the C5.0 algorithm on the training dataset made out of `cmc` Dataset in the form of confusion matrix. The performance on the training data shows that the algorithm could not learn 134 of the no-use and 444 of the short-term. The overall error rate is 52.4 %(578/1104*100=52.355≡52)

Performance Evaluation

Classification trees have a tendency to over fit the model to the training data and hence, the error rate as reported by the tree model is overly optimistic. For this reason we have only used 75% of the data set for training model. The purpose of leaving behind 25% of the dataset is to be evaluate the performance of the model built on unseen data or generalization of the model.

After applying the C5. 0 model to the train data, we do not know the outcome classes for the new data i.e., test data (`cmc.test`), so we predict the class for the new data instances using our model. To predict we use predict. C5. 0() function.

table( ) is used to build a contingency table of the counts at each combination of factors using the cross classifying factors

```
> pred.1 <- predict.C5.0(tree.1,newdata = cmc.test)

> addmargins(table(cmc.test$cmu,pred.1))

            pred.1
             no-use long-term short-term Sum
  no-use          70         0         95 165
  long-term       14         0         69  83
  short-term      26         0         95 121
  Sum            110         0        259 369
```

The above table builds contingency table of two objects `cmc.test` and pred. 1, here there are three factors and the numbers classify the factor count or the combination of factors count. We added add margins to our command to specify the each factor total of rows and columns

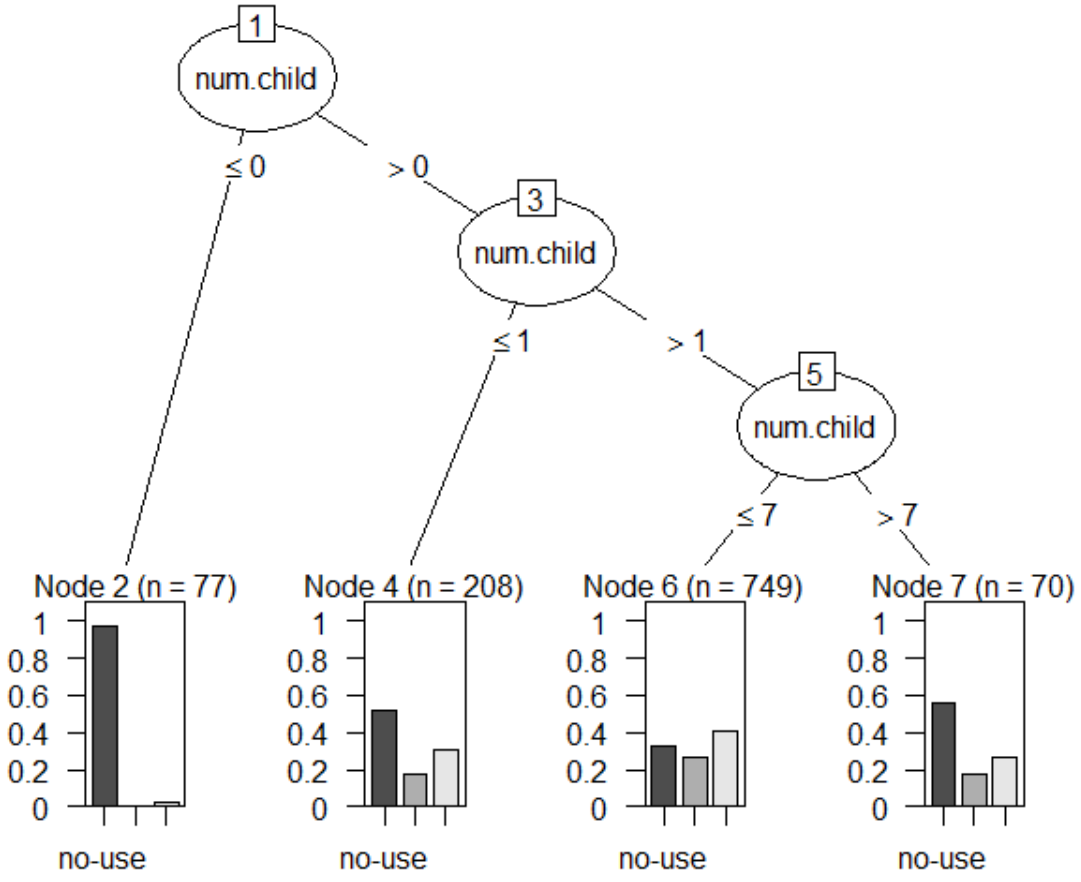
Here rows represent the predicted values and the columns is our test data factor levels. From the above we can easily identify our predicted values.

70 observations are classified to no-use factor, and 95 observations to short - term, these are predicted to the same, factor level, and the remaining observations are the combinations of two factors, means the observations are predicted by their combination of factors.

```
> round(sum(cmc.test$cmu==pred.1)/length(pred.1)*100,digits=2)

[1] 44.72
```

Tree plot

```
> plot(tree.1)
```



4.3.2 Decision tree build with by using two variables

```
> tree.2<-C5.0(cmu~num.child+wife.edu,data=cmc.train)

> tree.2
```

```
Call:
C5.0.formula(formula = cmu ~ num.child + wife.edu, data = cmc.
train)

Classification Tree
Number of samples: 1104
Number of predictors: 2

Tree size: 9

Non-standard options: attempt to group attributes
```

The above output reminds us the function call made, whether it is the classification tree or regression tree that was built, the number of examples used in learning and the predictor variable (`num.child, wife.edu`) used in building the classification tree it also has a mention about the size of the tree. This indicates how much depth the tree was grown. As per the output, it is 9 decisions deep.

To know the various decisions made by the tree to arrive at the final mode, let us call the `summary( )` function on the model object. The following is the output generated as result of the `summary( )` function call.

```
> summary(tree.2)
```

```
Call:
C5.0.formula(formula = cmu ~ num.child + wife.edu, data = cmc.
train)


C5.0 [Release 2.07 GPL Edition]       Tue Jun 02 15:36:20 2020
-------------------------------

Class specified by attribute `outcome'

Read 1104 cases (3 attributes) from undefined.data

Decision tree:

wife.edu = 4:
:...num.child <= 0: no-use (28)
:   num.child > 0:
:   :...num.child <= 2: short-term (188/110)
:       num.child > 2: long-term (220/111)
wife.edu in {1,2,3}:
:...wife.edu = 1: no-use (105/35)
    wife.edu = 3:
    :...num.child <= 1: no-use (77/25)
    :   num.child > 1: short-term (232/132)
    wife.edu = 2:
    :...num.child <= 2: no-use (104/31)
        num.child > 2:
        :...num.child <= 7: short-term (129/65)
            num.child > 7: no-use (21/6)


Evaluation on training data (1104 cases):

          Decision Tree
```

```
        ----------------
        Size      Errors

           9  515(46.6%)   <<


         (a)   (b)   (c)    <-classified as
        ----  ----  ----
         238    37   189    (a): class no-use
          23   109   118    (b): class long-term
          74    74   242    (c): class short-term


      Attribute usage:

      100.00%      wife.edu
       90.49%      num.child


Time: 0.0 secs
```

Interpreting the Output

1. If `wife.edu=4 and num.child<=0`, then classify the people as no-use. Observe that there are two numbers mentioned within a pair of parenthesis. The upper number indicates the size of the node or the number of examples satisfying the condition. The class label indicates which class type observations are more in number at that node. Note that this node is labelled as no-use. This means that most of the observations (or all) are of the type no-use. How many observations of the other class, short-term, are there at this node? The lower integer in the parenthesis indicates the size of the short-term meeting this condition. In other words, 0 of the short-term were classified as no-use at this node.
2. If `num.child>0 and num.child<=2`, then classify the people as short-term. There are 188 people satisfying this condition. The node is labeled as short-term. Of the 188 people, 110 people were misclassified.

3. If num.child>0 and num.child>2, then classify the people as long-term. There are 220 people satisfying this condition. The node is labeled as long-term. Of the 220 people, 111 people were misclassified.
4. If wife.edu in {1, 2, 3} and wife.edu=1, then classify the people as no-use. There are 105 people satisfying this condition. The node is labeled as no-use. Of the 105 people, 35 people were misclassified.
5. If wife.edu=3 and num.child<=1, then classify the people as no-use. There are 77 people satisfying this condition. The node is labeled as no-use. Of the 77 people, 25 people were misclassified.
6. If num.child=3 and num.child>1, then classify the people as short-term. There are 232 people satisfying this condition. The node is labeled as short-term. Of the 232 people, 132 people were misclassified.
7. If wife.edu=2 and num.child<=2, then classify the people as no-use. There are 104 people satisfying this condition. The node is labeled as no-use. Of the 104 people, 31 people were misclassified.
8. If num.child>2 and num.child<=7, then classify the people as short-term. There are 129 people satisfying this condition. The node is labeled as short-term. Of the 129 people, 65 people were misclassified.
9. If num.child>2 and num.child>7, then classify the people as no-use. There are 21 people satisfying this condition. The node is labeled as no-use. Of the 21 people, 6 people were misclassified.

The summary() function also indicates the performance of the C5.0 algorithm on the training dataset made out of cmc Dataset in the form of confusion matrix. The performance on the training data shows that the algorithm could not learn 307of the short-term and 111of the long-term and 97 of the no use. The overall error rate is 46.6 %(515/1104*100=46.6)

Performance Evaluation:

The size of the tree is 9 and error is 46.6 and the attribute usage for the wife.edu and num.child Comparing to other decision trees, decision tree with two variables gives more effective tree.

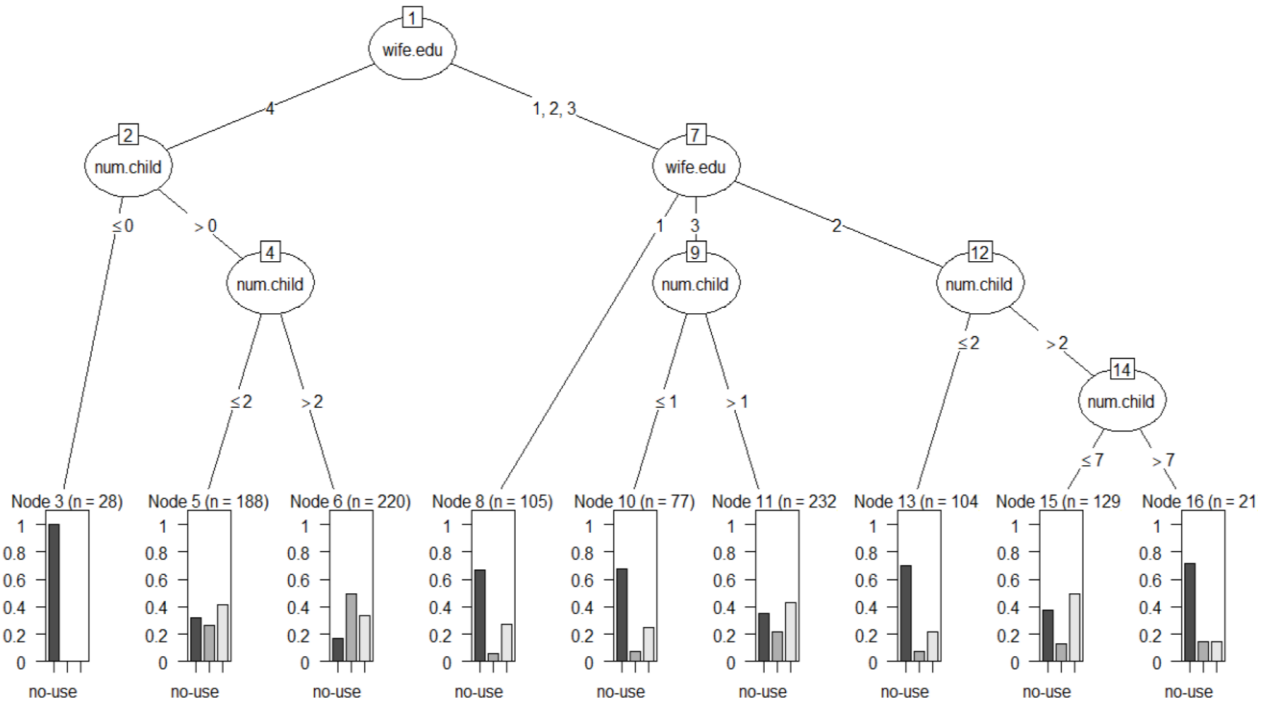
```
> pred.2=predict.C5.0(tree.2,newdata = cmc.test)

> addmargins(table(cmc.test$cmu,pred.2))
            pred.2
             no-use long-term short-term Sum
  no-use         74        17         74 165
  long-term       9        35         39  83
  short-term     32        26         63 121
  Sum           115        78        176 369
```

From the above table we understand that 74 observations are classified to no-use factor, and 35 observations to long - term, and 63 observations are short term these are predicted to the same, factor level, the remaining observations are the combinations of two factors, means the observations are predicted by their combination of factors.

```
> round(sum(cmc.test$cmu==pred.2)/length(pred.2)*100,digits=2)

[1] 46.61
```

Tree Plot

```
> plot(tree.2)
```



Variables importance

The C5.0 algorithm determines the importance of variables this algorithm class this as alternative us age in two forms and this is determine by an optional parameter metric.it has two values `"usage"` `and "splits"`. Importance of a particular feature is determined by computing the percentage of training set samples that fall into all the terminal nodes after the split. This is the default method. In other words metric option "usage" by default.

The syntax of the function `C5imp()` is:

```
C5imp(object, metric = "usage", pct = TRUE, ...)
```

When `metric = "splits"` the percentage of splits associated with each predictor is calculate d.

```
> C5imp(tree.2)
            Overall
  wife.edu    100.00
  num.child    90.49
 > C5imp(tree.2,metric = "splits")
             Overall
  num.child 71.42857
  wife.edu  28.57143
```

4.3.3 Decision tree build with by using four variables

```
> tree.4 <- C5.0(cmu ~ wife.age + wife.edu +
                         husb.edu + num.child,
                          data=cmc.train)
> tree.4
```

```
Call:
C5.0.formula(formula = cmu ~ wife.age + wife.edu + husb.edu +
num.child, data= cmc.train)

Classification Tree
Number of samples: 1104
Number of predictors: 4

Tree size: 21
Non-standard options: attempt to group attributes
```

The above output reminds us the function call made, whether it is the classification tree or regression tree that was built, the number of examples used in learning and the predictor variable `num.child`, `wife.age`,`wife.edu`,`husb.edu` used in building the classification tree it also has a mention about the size of the tree. This indicates how much depth the tree was grown. As per the output, it is 4 decisions deep.

To know the various decisions made by the tree to arrive at the final mode, let us call the `summary ( )` function on the model object. The following is the output generated as result of the `summary ( )` function call.

```
> summary(tree.4)
```

```
Call:
C5.0.formula(formula = cmu ~ wife.age + wife.edu + husb.edu + num.child, da
ta
 = cmc.train)


C5.0 [Release 2.07 GPL Edition]       Mon Jun 01 16:55:54 2020
-------------------------------

Class specified by attribute `outcome'

Read 1104 cases (5 attributes) from undefined.data

Decision tree:

num.child <= 0: no-use (77/2)
num.child > 0:
:...wife.edu in {1,2,3}:
    :...wife.age > 37: no-use (172/46)
    :   wife.age <= 37:
    :   :...num.child > 2: short-term (253/108)
    :       num.child <= 2:
    :       :...wife.age <= 25: no-use (116/61)
    :           wife.age > 25:
    :           :...num.child <= 1: no-use (29/6)
    :               num.child > 1:
    :               :...wife.age <= 28: long-term (23/13)
    :                   wife.age > 28: no-use (26/10)
    wife.edu = 4:
    :...num.child <= 2:
        :...husb.edu = 2: long-term (1)
        :   husb.edu in {1,3}: short-term (8)
        :   husb.edu = 4:
        :   :...wife.age > 29:
        :       :...num.child <= 1: no-use (22/6)
        :       :   num.child > 1: short-term (49/26)
        :       wife.age <= 29:
        :       :...num.child <= 1: long-term (72/46)
        :           num.child > 1:
        :           :...wife.age <= 27: short-term (28/12)
        :               wife.age > 27: long-term (8/2)
```

```
        num.child > 2:
        :...wife.age <= 42:
            :...wife.age > 35: long-term (76/27)
            :   wife.age <= 35:
            :   :...wife.age <= 32: short-term (50/25)
            :       wife.age > 32: long-term (36/15)
            wife.age > 42:
            :...num.child > 10: short-term (3)
                num.child <= 10:
                :...wife.age > 48: no-use (6/1)
                    wife.age <= 48:
                    :...wife.age <= 43: no-use (9/4)
                        wife.age > 43: long-term (40/17)


Evaluation on training data (1104 cases):

           Decision Tree
         ----------------
         Size      Errors

           21  427(38.7%)   <<


          (a)   (b)   (c)    <-classified as
         ----  ----  ----
          321    41   102    (a): class no-use
           45   136    69    (b): class long-term
           91    79   220    (c): class short-term


        Attribute usage:

        100.00% num.child
         93.03% wife.edu
         92.21% wife.age
         17.03% husb.edu


Time: 0.0 secs
```

Interpreting the Output

1. If `num.child<=0`, then classify the people as no-use. Observe that there are two numbers mentioned within a pair of parenthesis. The upper number indicates the size of the node or the number of examples satisfying the condition. The class label indicates which class type observations are more in number at that node. Note that this node is labelled as no-use. This means that most of the observations (or all) are of the type no-use. How many observations of the other class, short-term, are there at this node? The lower integer in the parenthesis indicates the size of the short-term meeting this condition. In other words, 2 of the short-term were classified as no-use at this node.
2. Otherwise, `num.child>0`.
3. If wife.edu `in {1, 2, 3} and wife.age>37`, then classify the people as no-use. There are 172 people satisfying this condition. The node is labeled as no-use. Of the 172 people, 46 people were misclassified.
4. If `wife.age<=37 and num.child>2`, then classify the people as short-term. There are 253people satisfying this condition. The node is labeled as short-term. Of the 253 people, 108 people were misclassified.
5. If `num.child>2 and wife.age<=25`, then classify the people as no-use. There are 116 people satisfying this condition. The node is labeled as no-use. Of the 116 people, 61 people were misclassified.
6. If `wife.age<=25 and num.child<=1`, then classify the people as no-use. There are 29 people satisfying this condition. The node is labeled as no-use. Of the 29 people, 6 people were misclassified.
7. If `num.child>1 and wife.age<=28`, then classify the people as long-term. There are 23 people satisfying this condition. The node is labeled as long-term. Of the 23 people, 13 people were misclassified.
8. If `num.child>1 and wife.age>28`, then classify the people as no-use. There are 26 people satisfying this condition. The node is labeled as no-use. Of the 26 people, 10 people were misclassified.
9. If `wife.age>29 and num.child<=1`, then classify the people as no-use. There are 22 people satisfying this condition. The node is labeled as no-use. Of the 22 people, 6 people were misclassified.

10. If `wife.age>29 and num.child>1`, then classify the people as shor-term. There are 49 people satisfying this condition. The node is labeled as short-term. Of the 49 people, 26 people were misclassified.
11. If `wife.age<=29 and num.child<=1`, then classify the people as long-term. There are 72 people satisfying this condition. The node is labeled as long-term. Of the 72 people, 46 people were misclassified.
12. If `num.child>1 and wife.age<=27`, then classify the people as short-term. There are 28 people satisfying this condition. The node is labeled as short-term. Of the 28 people, 12 people were misclassified.
13. If `num.child>1 and wife.age>27`, then classify the people as long-term. There are 8 people satisfying this condition. The node is labeled as long-term. Of the 8 people, 2 people were misclassified.
14. If `num.child>2.`
15. If `wife.age<=42 and wife.age>35`, then classify the people as long-term. There are 76 people satisfying this condition. The node is labeled as long-term. Of the 76 people, 27 people were misclassified.
16. If `wife.age<=35 and wife.age<=32`, then classify the people as short-term. There are 50 people satisfying this condition. The node is labeled as short-term. Of the 50 people, 25 people were misclassified.
17. If `wife.age<=35 and wife.age>32`, then classify the people as long-term. There are 36 people satisfying this condition. The node is labeled as long-term. Of the 36 people, 15 people were misclassified.
18. If `wife.age>42 and num.child>10`, then classify the people as short-term. There are 3 people satisfying this condition. The node is labeled as short-term. Of the 3 people, no people were misclassified.
19. If `num.child<=10 and wife.age>48`, then classify the people as no-use. There are 6 people satisfying this condition. The node is labeled as no-use. Of the 6 people, 1 people were misclassified.
20. If `wife.age<=48 and wife.age<=43`, then classify the people as no-use. There are 9 people satisfying this condition. The node is labeled as no-use. Of the 9 people, 4 people were misclassified.

21. If `wife.age<=48 and wife.age>43`, then classify the people as long-term. There are 40 people satisfying this condition. The node is labeled as long-term. Of the 40 people, 17 people were misclassified.

The `summary ()` function also indicates the performance of the C5.0 algorithm on the training dataset made out of `cmc` Dataset in the form of confusion matrix. The performance on the training data shows that the algorithm could not learn 136 of the no-use and 171 of the short-term and 120 of the long-term. The overall error rate is 38.7 %(427/1104*100=38)

Performance Evaluation:

The size of the tree is 21 and error is 38.7 and the attribute usage for the `wife.edu and num.child, wife.age, husb.edu` decision tree must be very large.

The predict function produces predicted classes or confidence value from a C5.0 model

When `type="class"` a factor vector is returned when `ype="prob"` a matrix of confidence values is returned (one columns per class)

```
> pred.4 <- predict.C5.0(tree.4,newdata = cmc.test)

> addmargins(table(cmc.test$cmu,pred.4))
            pred.4
             no-use long-term short-term Sum
  no-use        104        22         39 165
  long-term      20        35         28  83
  short-term     36        26         59 121
  Sum           160        83        126 369
```

From the above table we understand that 104 observations are classified to no-use factor, and 35 observations to long - term, and 59 observations are short term these are predicted to the same, factor level, the remaining observations are the combinations of two factors, means the observations are predicted by their combination of factors

```
> round(sum(cmc.test$cmu==pred.4)/length(pred.4)*100,digits=2)

[1] 53.66
```

Variables importance

The C5.0 algorithm determines the importance of variables this algorithm class this as alternative usage in two forms and this is determine by an optional parameter metric.it has two values "`usage` "and "`splits`". Importance of a particular feature is determined by computing the percentage of training set samples that fall into all the terminal nodes after the split. This is the default method. In other words metric option "usage" by default.

The syntax of the function C5imp() is:

```
C5imp(object, metric = "usage", pct = TRUE, ...)
```

When `metric = "splits"` the percentage of splits associated with each predictor is calculated.

```
    > C5imp(tree.4)
          Overall
  num.child  100.00
 wife.edu    93.03
 wife.age    92.21
 husb.edu    17.03
> C5imp(tree.4,metric = "splits")
           Overall
wife.age  52.631579
num.child 36.842105
husb.edu   5.263158
wife.edu   5.263158
```

4.4 Boosted Tree

Boosting is a powerful feature incorporated in C5.0 .the idea is to generate several classifier (either decision tree or ruleset) rather than just one. When a new case is to be classified, each classifier votes for its predicted class and the votes are counted to determine the final class.

```
> c5 <- C5.0Control(minCases = 5)
> boost.tree <- C5.0(cmu ~ wife.age + wife.edu +
+                          num.child,
+                    data=cmc.train,control=c5,
+                    trials=10)
> boost.tree
```

```
Call:
C5.0.formula(formula = cmu ~ wife.age + wife.edu +
num.child, data =
 cmc.train, control = c5, trials = 10)

Classification Tree
Number of samples: 1104
Number of predictors: 3

Number of boosting iterations: 10 requested;  3 used
due to early stopping
Average tree size: 9.7

Non-standard options: attempt to group attributes,
minimum number of cases: 5
```

By default, the C5.0 function takes trails=1. And taking the values between 1 to 100 you get a boosted tree with a single independent variable such as `num.child` or `wife.age` variables you can get a simple tree otherwise tree is very large.

```
> summary(boost.tree)
```

```
Call:
C5.0.formula(formula = cmu ~ wife.age + wife.edu + num.child, data =
 cmc.train, control = c5, trials = 10)
```

```
C5.0 [Release 2.07 GPL Edition]  Mon Jun 01 16:06:05 2020
-------------------------------

Class specified by attribute `outcome'

Read 1104 cases (4 attributes) from undefined.data

-----  Trial 0:  -----

Decision tree:

num.child <= 0: no-use (77/2)
num.child > 0:
:...wife.edu = 4:
    :...num.child <= 2:
    :   :...num.child > 1: short-term (90/45)
    :   :   num.child <= 1:
    :   :   :...wife.age <= 31: short-term (79/50)
    :   :       wife.age > 31: no-use (19/4)
    :   num.child > 2:
    :   :...wife.age <= 42:
    :       :...wife.age > 35: long-term (76/27)
    :       :   wife.age <= 35:
    :       :   :...wife.age <= 32: short-term (50/25)
    :       :       wife.age > 32: long-term (36/15)
    :       wife.age > 42:
    :       :...num.child > 7: short-term (8/4)
    :           num.child <= 7:
    :           :...wife.age <= 48: long-term (45/21)
    :               wife.age > 48: no-use (5)
    wife.edu in {1,2,3}:
    :...wife.age > 37: no-use (172/46)
        wife.age <= 37:
```

```
        :...num.child > 2: short-term (253/108)
            num.child <= 2:
            :...num.child > 1: no-use (99/56)
                num.child <= 1:
                :...wife.age > 23: no-use (46/11)
                    wife.age <= 23:
                    :...wife.age <= 20: no-use (22/10)
                        wife.age > 20: short-term (27/10)

-----  Trial 1:  -----

Decision tree:

wife.edu = 4:
:...num.child <= 2: no-use (219.2/128.8)
:   num.child > 2: long-term (221.2/118.5)
wife.edu in {1,2,3}:
:...wife.age > 37: no-use (174.5/52.3)
    wife.age <= 37:
    :...num.child <= 1: no-use (131.4/43.9)
        num.child > 1: short-term (357.7/184.6)

-----  Trial 2:  -----

Decision tree:

num.child <= 0: no-use (69.3/2.3)
num.child > 0:
:...wife.edu = 4:
    :...wife.age > 42: no-use (67.3/37.6)
    :   wife.age <= 42:
    :   :...num.child <= 2: short-term (186.8/113.8)
    :       num.child > 2: long-term (163/86)
    wife.edu in {1,2,3}:
    :...wife.age > 36: no-use (185.1/61.9)
```

```
        wife.age <= 36:
        :...wife.edu in {1,3}: short-term (258.8/137.2)
            wife.edu = 2:
            :...wife.age <= 27: no-use (89.1/40.5)
                wife.age > 27: short-term (84.6/46.7)

-----  Trial 3:  -----

Decision tree:

wife.edu = 4:
:...num.child <= 2: no-use (220.8/131.6)
:   num.child > 2: long-term (222.6/123)
wife.edu in {1,2,3}:
:...wife.age > 37: no-use (171.6/54.6)
    wife.age <= 37:
    :...num.child <= 1: no-use (130.4/45)
        num.child > 1: short-term (358.7/191.4)

*** boosting reduced to 3 trials since last classifier is very
inaccurate


Evaluation on training data (1104 cases):

Trial      Decision Tree
-----    ----------------
         Size      Errors

   0       16  434(39.3%)
   1        5  498(45.1%)
   2        8  488(44.2%)
boost          447(40.5%)   <<
```

```
          (a)    (b)    (c)    <-classified as
         ----   ----   ----
          288     30    146    (a): class no-use
           33    107    110    (b): class long-term
           58     70    262    (c): class short-term


       Attribute usage:

       100.00%       wife.edu
       100.00%       num.child
        97.46%       wife.age


Time: 0.0 secs
```

4.4.1 Interpretation of boosted tree:

In this example, our command causes three decision trees to be generated. The summary of the trees individual and aggregated performance on the 1104 test cases is:

```
Trial       Decision Tree
-----     ----------------
          Size      Errors

   0        16  434(39.3%)
   1         5  498(45.1%)
   2         8  488(44.2%)
boost           447(40.5%)   <<


           (a)   (b)   (c)    <-classified as
          ----  ----  ----
           288    30   146    (a): class no-use
            33   107   110    (b): class long-term
            58    70   262    (c): class short-term
```

The performance of the classifier constructed at each trail is summarized on a separate line, while the line labelled boost shows the result of voting all the classifiers.

The decision tree constructed on Trial 0 is identical to that produced without the -b option. Some of the subsequent trees produced by paying more attention to certain cases have relatively high overall error rates. Nevertheless, when the trees are combined by voting, the final predictions have a lower error rate of 39.3% on the test cases.

```
> pred.boost <- predict.C5.0(boost.tree,newdata = cmc.test)

> table(cmc.test$cmu,pred.boost)
```

	pred.boost		
	no-use	long-term	short-term
no-use	91	16	58
long-term	16	35	32
short-term	23	26	72

```
>round(sum(cmc.test$cmu==pred.boost)/length(pred.boost)*100,digits=2)

[1] 53.66
```

4.5 Rule-Based Tree:

Rule-based decision tree methods handle manipulating in the data through the rules induced from the data not the decision tree itself. A declarative representation such as a set of decision rule is much easier to modify and adapt to different situation than a procedural one. This easiness is due to the absence of constraints on the order of evaluating the rule to create a rule based tree use rules=TRUE option.

```
> rule.tree <- C5.0(x = cmc.train[, c(1,2,4)],
+                   y = cmc.train$cmu,
+                   rules = TRUE)
```

```
> rule.tree

Call:
C5.0.default(x = cmc.train[, c(1, 2, 4)], y = cmc.train$cmu, rules =
TRUE)

Rule-Based Model
Number of samples: 1104
Number of predictors: 3

Number of Rules: 14

Non-standard options: attempt to group attributes
> summary(rule.tree)
```

```
Call:
C5.0.default(x = cmc.train[, c(1, 2, 4)], y = cmc.train$cmu, rules = TRUE)


C5.0 [Release 2.07 GPL Edition]       Mon Jun 01 17:34:18 2020
-------------------------------

Class specified by attribute `outcome'

Read 1104 cases (4 attributes) from undefined.data

Rules:

Rule 1: (77/2, lift 2.3)
        num.child <= 0
        ->  class no-use  [0.962]

Rule 2: (57/5, lift 2.1)
        wife.age > 31
        num.child <= 1
        ->  class no-use  [0.898]
```

```
Rule 3: (180/46, lift 1.8)
        wife.age > 37
        wife.edu in {1, 2, 3}
        ->  class no-use  [0.742]

Rule 4: (163/59, lift 1.5)
        wife.age > 42
        num.child <= 10
        ->  class no-use  [0.636]

Rule 5: (277/104, lift 1.5)
        wife.edu in {1, 2, 3}
        num.child <= 2
        ->  class no-use  [0.624]

Rule 6: (40/17, lift 2.5)
        wife.age > 43
        wife.age <= 48
        wife.edu = 4
        num.child > 2
        num.child <= 10
        ->  class long-term  [0.571]

Rule 7: (220/111, lift 2.2)
        wife.edu = 4
        num.child > 2
        ->  class long-term  [0.495]

Rule 8: (3, lift 2.3)
        wife.edu = 4
        num.child > 10
        ->  class short-term  [0.800]

Rule 9: (15/5, lift 1.8)
        wife.age > 24
        wife.age <= 25
        num.child > 1
        num.child <= 2
        ->  class short-term  [0.647]

Rule 10: (27/10, lift 1.8)
```

```
        wife.age > 20
        wife.age <= 23
        wife.edu in {1, 2, 3}
        num.child > 0
        num.child <= 1
        ->  class short-term  [0.621]

Rule 11: (39/15, lift 1.7)
        wife.age <= 22
        num.child > 1
        ->  class short-term  [0.610]

Rule 12: (212/86, lift 1.7)
        wife.age <= 32
        num.child > 2
        ->  class short-term  [0.593]

Rule 13: (447/235, lift 1.3)
        wife.age <= 37
        wife.edu in {1, 2, 3}
        num.child > 0
        ->  class short-term  [0.474]

Rule 14: (188/110, lift 1.2)
        wife.edu = 4
        num.child > 0
        num.child <= 2
        ->  class short-term  [0.416]

Default class: no-use


Evaluation on training data (1104 cases):

               Rules
          ----------------
            No      Errors

            14  424(38.4%)   <<
```

```
           (a)    (b)    (c)    <-classified as
          ----   ----   ----
           305     15    144    (a): class no-use
            51     93    106    (b): class long-term
            64     44    282    (c): class short-term


       Attribute usage:

        97.46% wife.edu
        94.47% num.child
        70.11% wife.age
```

From the above summary we understand the model ,there are 1104cases and 4 attributes .And the rules are explained as

Rule 1: (77/2, lift 2.3)

```
num.child <= 0
->  class no-use  [0.962]
```

The statistics (77/2,lift 2.3) that summarize the performance of the rule

1. The condition is num.child<=0 then class is no-use with 0.962 rule's estimated accuracy.

2. Among the 1104 training cases there are 77 the number of training cases covered by the rule and 2 of them do not belong to the class predicted by the rule ,that satisfy the condition, lift 2.3 is the result of dividing the rule's estimated accuracy by the relative frequency of the predicted class In the training data set.

Rule 2: (57/5, lift 2.1)

```
wife.age > 31
num.child <= 1
```

```
    ->  class no-use  [0.898]
```

there are 57 number of training cases covered by the rule and 5 of them do not belong to the class predicted by the rule ,that satisfy the two conditions, lift is 2.1 is the result of dividing the rule's estimated accuracy by the relative frequency of the predicted class In the training data set.

1. The condition is if wife.age>31 and num.child<=1 then the class is no-use with rule's estimated accuracy 0.898
2. there are 57 number of training cases covered by the rule and 5 of them do not belong to the class predicted by the rule ,that satisfy the two conditions, lift is 2.1 is the result of dividing the rule's estimated accuracy by the relative frequency of the predicted class In the training data set.

Rule 3: (180/46, lift 1.8)

```
    wife.age > 37
wife.edu in {1, 2, 3}
->  class no-use  [0.742]
```

1. The condition is if wife.age>37 and wife.edu in {1,2,3} there is three conjunct's the minimizes the entropy measure, then the class is no-use with rule's estimated accuracy 0.742
2. There are 57 number of training cases covered by the rule and 5 of them do not belong to the predicted class that satisfy the two conditions, lift is 1.8.

Rule 4: (163/59, lift 1.5)

```
wife.age > 42
   num.child <= 10
   ->  class no-use  [0.636]
```

1. The condition is if wife.age>42and num.child<=10,then the class is no-use with rule's estimated accuracy 0.636
2. There are 163number of training cases and 59 of them do not belong to the predicted class that satisfy the two conditions, and the lift is 1.5.

Similarly, the remaining rules are, when the rule's estimated accuracy is 1 then it is perfect rule, here rule 1 is nearer to 1,then we assume it is as a perfect rule.

Attribute usage:

3. The first column shows the percentage of cases for which the named attribute appears in a condition of an applicable rule,i.e.,wife education is used in the condition part of the rules

That cover 97.46% of cases, similarly 94.47% of cases are covered by the num.child, and 70.11% of cases are covered by wife. 98% of cases. 38.4% of error occurred

```
> pred.rule <- predict(rule.tree,newdata = cmc.test)

> table(cmc.test$cmu,pred.rule)
           pred.rule
            no-use long-term short-term
  no-use         99        13         53
  long-term      19        28         36
  short-term     29        15         77

> round(sum(cmc.test$cmu==pred.rule)/length(pred.rule)*100,digits=2)

[1] 55.28
```

5.1 Need for Decision Making

As a part of their day to day work, different organizations such as business firms, banks, hospitals, research institutes and so on collect huge amounts of data. These organizations require analysis of the data so collected in a timely manner either for making important decisions, or to discover important patterns in the data unseen or unknown earlier. The important decisions involved in a particular situation may be as follows: in the case of a bank, the important decision may be, based on the parameter as collected by the bank, whether to sanction a loan to a prospective customer or to offer a credit card, or to issue an insurance policy and so on; in the case of a medical situation, a doctor has to decide whether the patient has a particular disease on the basis of the physical symptoms observed and the reports of the medical tests conducted on the patient; on the basis of the data collected on a patient or doctor may also be faced with making several decisions such as: whether to treat the patient as in-patient or out-patient, whether to conduct an operation or to prescribe medicines only and the list goes on. In a business situation, a marketing manager, having the huge number of customer records describing large number of customer records describing large number of features of each customer, is faced with how to segment the customer database for targeting a particular marketing campaign. There are quite a large number of situations, one is required analyse large amounts of data quickly and correctly, as these both factors may have an influence on their business.

5.2 Sources of Solution Methods

Machine Learning and data mining techniques will help to achieve this. We have mentioned only one type of prediction problem, known as classification problem, in all our problems, in all our example situations above as it is our project theme. It will be so helpful to have a simple and efficient models to make decisions instead of analysing the entire data again and again. Machine learning algorithms help use learn the data and come up with simple models. Decision tree is a particular machine learning tool that will be useful in making predictions. The method is simple to implement and the results are easy to interpret. Decision trees can further be broadly classified as Classification Trees and Regression Trees. If the target variable to be predicted is categorical variable, then the decision tree involved is called a Classification tree. In Regression trees, the target variable is a continuous variable.

There are quite a number of algorithms out there to learn a Classification tree. Some of the popular learning algorithms are ID3, C4.5, C5.0, all due to Ross Quinlan, from machine learning community and CART (an acronym for Classification and Regression Trees) due to a team of

statisticians. All these models are simple in the sense that they build only one classification tree for the prediction purpose. To improve the performance of these models in different directions, researchers have proposed various methods, which resulted in various classes of techniques: Bagging (Bootstrap Aggregation), Boosting, Random Forests and many more. The underlying basic idea in all these models is to build several models and aggregate the outputs to come out with a single decision. These methods are called ensemble methods.

5.3 Software used

There were quite a number of software, some are proprietary, that have implementations for the above algorithms. To build predictive models, IBM has developed software called IBM SPSS Modeller, SAS has data mining module called SAS Enterprise Miner and several others. The software just mentioned is proprietary. We don't have access to them. Among the free software that have implementations for the above mentioned algorithms are R and Weka. Our choice is R.

Packages used

`rpart()`	Recursive partitioning and regression tree
`rpart.plot()`	Plot '`rpart`' models: an Enhanced version of '`plot.rpart`'
`C5.0`	C5.0 Decision tree and Rule-Based models
`DescTools()`	Tools for Descriptive statistics
`vcd()`	Visualizing Categorical Data
`rcompanion()`	Function to Support Extension Education Program

5.4 Execution of the Project work

We would like to mention only about the content of the various previous Chapters in this project work, as the objective is to get an insight into those machine learning algorithms. The purpose behind the choice of the dataset is to have a simple dataset that has small number of variables and to understand the intent of the variables easily.

Chapter-1 we discussed the classification and regression problem using a decision tree in general, and its application in the various practical situations. The objectives of the project work are specified in clear cut terms. The dataset is downloaded from the Machine learning data repository of UCI. It also contains an indication of what is expected in different Chapters that come up later.

In Chapter-2 we explore the data. The purpose behind the choice of the dataset is to have a simple dataset that has small number of variables and to understand the intent of the variables easily. We explored the explanatory variables that are related to the class variables in this association and also we used **`DescTools`** package and **`plotcirc()`** to visualizing the data.

In Chapter-3 introduces the problem mentioned in the opening paragraph of this Chapter A study of the general characteristics of the data set is explored. It explores the Contraceptive method choice dataset using simple classification trees. By simple classification trees we mean a single model in making decision. The primary package used is `rpart`, which implements several concepts discussed in the book by Breiman et al (1984). The classification tree has an overall error rate of 57% on the training dataset and 55% on the test dataset. Only three variables have participated in the construction of the model, namely, `num.child, wife.age, wife.edu` in the training dataset and five variables have participated in the construction of the model, namely, `hus.job, num.child, SOL.index, wife.age` and `wife.edu` in the test dataset.

In Chapter-4, we have also used Ross Quinlan's C5.0 algorithm implemented in C50 package for building a simple classification tree. The classification tree using two variables give to C5.0 algorithm about 52.74. As per this algorithm important attribute is `num.child`. The classification tree using four variables give to C5.0 algorithm about 46.6% as per this algorithm important attributes are `num.child, wife.edu, wife.age, hus.edu.` Boosting is a powerful feature incorporated in C5.0 .In the boosted tree the final predictions have a lower error rate of 39.3% on the test cases. In Rule based tree the error rate is 38.4%

And finally, in this Chapter, Chapter-5, we summarize the objects of the project work, dataset used and a brief description of the various Chapters.

Bibliography

1. Grahham Williams(2011).Data Mining with Rattle and R-The Art of Exacting Data for Knowledge Discovery Springer
2. Daniel T.Larose Chantal D.Larose Data Mining and Predictive Analytics-Wiley Series on Methods and Application in Data Mining WILEY
3. https://cran.r-project.org/
4. Recursive partitioning for classification, regression and survival trees. An implementation of most of the functionality of the 1984 book by Breiman, Friedman, Olshen and Stone.

   https://CRAN.R-project.org/package=rpart

5. C5.0 decision trees and rule-based models for pattern recognition that extend the work of Quinlan (1993)

   https://CRAN.R-project.org/package=C50

6. https://CRAN.R-project.org/package=rpart.plot
7. https://CRAN.R-project.org/package=DescTools
8. Functions and datasets to support "Summary and Analysis of Extension Program Evaluation in R"
9. https://CRAN.R-project.org/package=rcompanion
10. https://cran.r-project.org/web/packages/C50/vignettes/C5.0.html
11. https://topepo.github.io/C5.0/reference/C5.0.html
12. https://www.rulequest.com/see5-unix.html
13. https://rviews.rstudio.com/2020/05/21/modern-rule-based-models/
14. https://www.rulequest.com/cubist-unix.html