1 Topic

Handling missing values during the classification process (when applying the classifier).

The treatment of missing values during the learning process has been received a lot of attention of researchers. We have published a tutorial about this in the context of logistic regression induction[1]. By contrast, the handling of the missing values during the classification process, i.e. when we apply the classifier on an unlabeled instance, is less studied. However, the problem is important. Indeed, the model is designed to work only when the instance to label is fully described. If some values are not available, we cannot directly apply the model. We need a strategy to overcome this difficulty[2].

In this tutorial, we are in the supervised learning context. The classifier is a logistic regression model. All the descriptors are continuous. We want to evaluate on various datasets from the UCI repository **the behavior of two imputations methods: the univariate approach and the multivariate approach**. The constraint is that the imputation models must rely on information from the learning sample. We consider that this last one does not contain missing values.

We note that the occurrence of the missing value on the instance to classify is "missing completely at random" in our experiments i.e. each descriptor has the same probability to be missing.

2 Two imputation approaches for the classification process

Let us take a small example to describe our problem. We use the following learning sample for the induction of the classifier.

	age	pression	cholester	maladie
Echantillon d'apprentissage	56	134	409	presence
	52	152	298	absence
	55	160	289	presence
	56	132	184	presence
	64	140	335	presence
	42	130	180	absence
	61	130	330	presence
	59	140	221	absence
	52	128	255	presence
	44	120	263	absence

Logistic regression model. We want to predict the occurrence of disease ("maladie" in French) of individuals from their various characteristics (age, blood pressure, cholesterol). We obtain the following classification model using the BINARY LOGISTIC REGRESSION component of Tanagra:

[1] http://data-mining-tutorials.blogspot.fr/2012/10/handling-missing-values-in-logistic.html

[2] We find a description of the various possible approaches in this paper: Saar-Tsechansky, Provost, "Handling Missing Values when Applying Classification Models", JMLR, 8, pp. 1625-1657, 2007.

Adjustement quality

Predicted attribute		maladie
Positive value		presence
Number of examples		10
Model Fit Statistics		
Criterion	Intercept	Model
AIC	15.46	16.29
SC	15.763	17.5
-2LL	13.46	8.29
Model Chi² test (LR)		
Chi-2		5.1703
d.f.		3
P(>Chi-2)		0.1597
R²-like		
McFadden's R²		0.3841
Cox and Snell's R²		0.4037
Nagelkerke's R²		0.5458

Attributes in the equation

Attribute	Coef.	Std-dev	Wald	Signif
constant	-11.8494	12.1207	0.9557	0.3283
age	0.2933	0.224	1.715	0.1903
pression	-0.0542	0.0815	0.4428	0.5058
cholester	0.0152	0.0215	0.5004	0.4793

The overall model is not significant at the 5% level. No variable is statistically significant. This is not really surprising because we have a very small dataset (10 instances). Nevertheless we keep this model for the rest of our paper.

Classifying an unlabeled instance. The classifier is the following:

C = -11.8494 + 0.2933 x âge – 0.0542 x pression + 0.0152 x cholestérol

For an instance ω that we want to classify, if $C(\omega) > 0$, we predict the presence of the disease, otherwise we conclude to its absence.

Let us consider an instance with the following characteristics:

	age	pression	cholester	maladie
à classer	51	130	305	**???**

We apply the classification model and we obtain

$C(\omega) = -11.8494 + 0.2933 \times 51 - 0.0542 \times 130 + 0.0152 \times 305 = 0.7086$

We consider that this is the true value of the logit for this instance since we have all the values of the predictive variables. We predict the "**presence**" of the disease (*maladie*) for this new instance.

The individual is not completely described. Now, we suppose that we have not the age of the individual. We dispose of the following description.

	age	pression	cholester	maladie
à classer	???	130	305	???

How to calculate the value of C(ω) in this context?

2.1 Some bad ideas

One strategy is simply to refuse the classification of the instance. If this attitude is understandable for a statistician - the conditions of application of the model are not fulfilled - it is less tenable in a professional context. We must try to assign a label to the individual even if the conditions are not perfect.

Another approach is simply to ignore the variables with missing value and apply the model only for the available description. This is a bad strategy. In our case, it means that we assign **0** to the age of the individual to classify. We get the following result:

$$C(\omega) = -11.8494 - 0.0542 \times 130 + 0.0152 \times 305 = -14.2521$$

The classification is bad.

In this tutorial, we evaluate the behavior of two solutions for imputation that are more or less sophisticated. We do not discuss the statistical characteristics (bias and variance) but rather the predictive performance. For that, we use an experiment scheme that we detail below (section 3).

2.2 Univariate approach (U1)

This approach is very simplistic. For a variable, we replace the missing value with its average computed on the learning sample. Thus, we neutralize the variable in the prediction. For our dataset, we obtain easily the mean for each descriptor.

	age	pression	cholester	maladie
Echantillon d'apprentissage	56	134	409	presence
	52	152	298	absence
	55	160	289	presence
	56	132	184	presence
	64	140	335	presence
	42	130	180	absence
	61	130	330	presence
	59	140	221	absence
	52	128	255	presence
	44	120	263	absence
Moyennes	**54.1**	**136.6**	**276.4**	

Then we replace the age with 54.1 for the instance that we want to classify:

	age	pression	cholester	maladie
à classer	54.1	130	305	???

When we apply the classifier, we obtain:

$$C(\omega) = -11.8494 + 0.2933 \times 54.1 - 0.0542 \times 130 + 0.0152 \times 305 = 1.6180$$

We are getting closer to the true value of C (for this example). And the conclusion is consistent with the one calculated on the full description of the individual.

When there are several missing values. This solution is operational even when several variables are missing. We simply replace each missing value by its average calculated on the learning sample.

2.3 Multivariate approach (U2): only one value is missing

The predictive variables are rarely independent from each other. We can use the existing relations to produce a better imputation value. We use a multiple linear regression as imputation model in this tutorial. But in fact, we can use any other predictive method (e.g. regression tree, etc.).

Of course, this strategy will be all the more performing that the variables are strongly related i.e. they are redundant in the model. But this is not really a good new. Indeed, we want to build parsimonious models. The predictive variables must not be collinear. The goal of the feature selection process is to remove related variables. This requirement makes the multivariate imputation of missing values less effective as we will see in the experimentation. I have not really an answer to this dilemma. This remains an open question.

We take again the individual that we want to classify above:

	age	pression	cholester	maladie
à classer	???	130	305	**???**

To get a value for age, we have previously built a regression model which explains the age from the other predictive variables using the learning sample. Here are the parameters of the model:

Age = 0.0404 x cholestérol + 0.1482 x pression + 22.6974

Thus, the imputed value for age is:

Age(ω) = 0.0404 x 305 + 0.1482 x 130 + 22.6974 = 54.3

Thereafter, we use this value for the calculation of the logit of the logistic regression model:

C(ω) = -11.8494 + 0.2933 x 54.3 – 0.0542 x 130 + 0.0152 x 305 = 1.6697

Like for the univariate imputation, the logit is different to the true value. But we predict also the presence of the disease.

Measuring the redundancy of the descriptors in a dataset. We are a little bit disappointed in our example. The multivariate imputation proposes a value (54.3) which is very similar to the simple average (54.1). It is far to the observed value of age (51). The other variables cannot be used to predict the value of age in the imputation process.

To measure the relation between the descriptors, we use the correlation matrix. Indeed, the correlations are not strong between the pairs of variables.

	age	pression	cholesterol
age	1	0.3299	0.4620
pression	0.3299	1	0.1839
cholesterol	0.4620	0.1839	1

déterminant	0.7000

To measure the overall relation between the variable, we calculate the determinant D of the correlation matrix. If the variables are highly redundant: D ≈ 0; if they are uncorrelated: D ≈ 1.

In our dataset, we have D = 0.7. The variables are weakly related. We use this indicator D to measure the degree of redundancy of the variables in our experiments.

2.4 Multivariate imputation (U2): several values are missing

When several values are missing e.g. age and cholesterol are missing, we cannot use ***directly*** the regression model to impute the value of a variable from the other descriptors of the database.

In this tutorial, we use a very simplistic approach. When we want to impute the value of a variable from a regression model, if some values are missing, we replace them by their average computed on the learning set.

Let us consider the following individual, only the value of pressure is available.

	age	pression	cholester	maladie
à classer	???	130	???	???

When we want to impute "age", we use the linear regression model:

Age = 0.0404 x cholesterol + 0.1482 x pression + 22.6974

But the value of cholesterol is not available. We replace it by the mean calculated on the learning set i.e. 276.4. Thus, the imputed value for age is:

Age (ω) = 0.0404 x 276.4 + 0.1482 x 130 + 22.6974 = 53.1

For "cholesterol", we have the regression model:

Cholesterol = 0.2125 x pression + 4.6343 x age - 3.3389

Here, we replace the age by its mean (54.1). Thus, the imputed value for cholesterol is:

Cholestérol (ω) = 0.2125 x 130 + 4.6343 x 54.1 - 3.3389= 275.0

The description used for the classification process is then:

	age	pression	cholester	maladie
à classer	53.1	130	275.0	???

We obtain the computed logit for the instance that we want to classify,

C (ω) = -11.8494 + 0.2933 x 53.1 – 0.0542 x 130 + 0.0152 x 275.0 = 0.8740

We predict the presence of disease.

Note. We note that the univariate imputation by mean is a special case of the imputation by regression. Indeed, the best prediction of the variable with a constant is the mean of the variable.

3 Experimental comparison

3.1 Organization of the experiment

To evaluate the efficiency of the two approaches, we use several datasets from the UCI repository: Pima Indian Diabetes[3], Breast Cancer Wisconsin[4], and a binary version of Waveform[5]. The missing values have been imputed arbitrarily in order to not interfere with our experiment.

We perform the following sequence of treatments under the R software:

1. We load the database.
2. We subdivide the dataset into train and test samples according to a certain proportion (PROPORTION_TRAIN).
3. We learn the classifier parameters on the learning sample. We obtain the model (LOG.REG) that we use to classify unlabeled instances.
4. We apply LOG.REG on the test sample. We obtain the reference (base) test error rate (ERR.REF). We should not obtain a lower error rate than this value because we use a test sample without missing values at this step.
5. For each instance of the test sample, we insert “k” missing values. The concerned variables are randomly chosen for each instance.
6. We apply the classifier on this new test sample using the univariate imputation strategy (U1). We obtain a new value of test error rate (ERR.U1).
7. We apply the classifier on this test sample using the multivariate imputation strategy (U2). We obtain ERR.U2.

The steps 5 to 7 are repeated N = 200 times. We compute the mean to obtain a reliable estimation of the error rate for each value of k.

We have tried different value of “k” (k = 1, ..., M = 7) to detect the threshold from which we cannot apply the classifier to predict the class of an instance with missing values. As we see below, we observe in our experiments that ERR.U1 and ERR.U2 are more and more increased (the classification process is less efficient) when "k" increases. This is not surprising. But the evolution of ERR.U1 and ERR.U2 are not the same according to the redundancy of the predictive variables.

3.2 The main results of the experiments

Breast Cancer Wisconsin. The dataset contains 699 instances and 9 predictors. We subdivide it in equal parts (PROPORTION_TRAIN = 0.5). We learn the following model on the learning sample:

[3] http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

[4] http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

[5] http://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+%28Version+2%29

```
R Console
> print(modele.glm)

Call:  glm(formula = classe ~ ., family = binomial, data = donnees$train)

Coefficients:
(Intercept)        clump    ucellsize   ucellshape   mgadhesion       sepics
  -12.80334      0.78153      0.20599      0.04127      0.52657     -0.30412
    bnuclei   bchromatin     normnucl      mitoses
    0.58035      0.64908      0.28160      1.13158

Degrees of Freedom: 348 Total (i.e. Null);  339 Residual
Null Deviance:      445.2
Residual Deviance: 40.75        AIC: 60.75
>
```

The base error rate obtained on the test sample without missing values is **ERR.REF = 0.042857**.

It seems that the predictors are highly redundant. Indeed, the determinant of the correlation matrix is very low **D = 0.000699.**

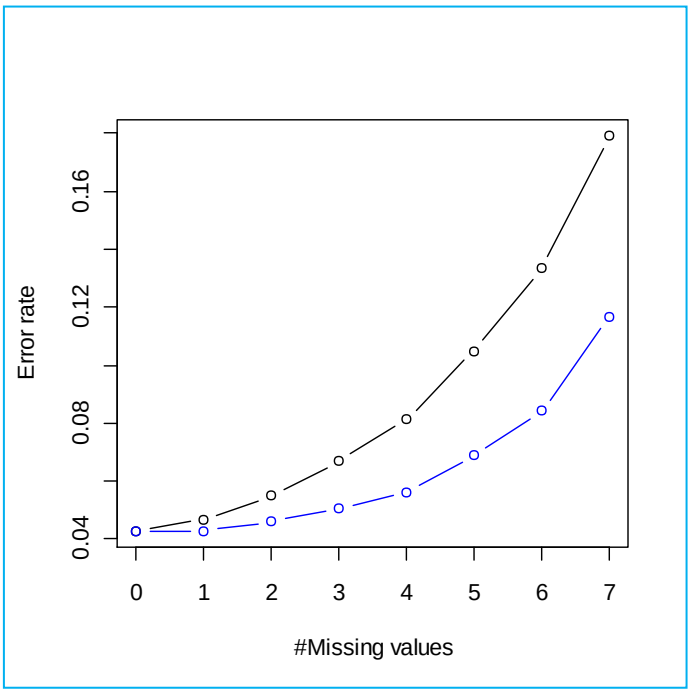

Figure 1 - BREAST: Error rate vs. number of missing features for each instance to classify

Obviously, the imputation enables to overcome the presence of missing values in classification process. But it is less and less effective as their number “k” increases (Figure 1). This is not surprising. We observe also that the multivariate approach (**U2** – in blue) is far better than the univariate one (**U1** – in black) for the high value of "k". Because the predictors are redundant, the imputation which relies on the information provided by the other variables is better.

Pima Indian Diabete. It contains 768 instances and 8 predictors. We set PROPORTION_TRAIN to 0.5. The learned model is:

```
R Console
> print(modele.glm)

Call:  glm(formula = classe ~ ., family = binomial, data = donnees$train)

Coefficients:
(Intercept)     pregnant       plasma    diastolic      triceps        serum
  -7.090494     0.104550     0.032002    -0.015782     0.001300    -0.001257
   bodymass     pedigree          age
   0.061382     1.111057     0.018395

Degrees of Freedom: 383 Total (i.e. Null);  375 Residual
Null Deviance:      491.6
Residual Deviance: 372.8        AIC: 390.8
>
```

The base test error rate is **ERR.REF = 0.2421875**. We observe that the predictors are weakly redundant (less than Breast Cancer anyway) with **D = 0.257740**.

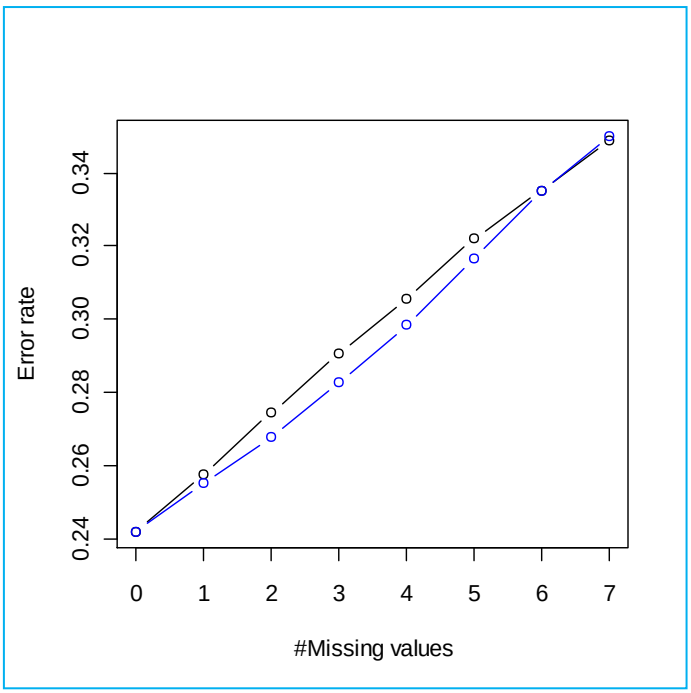

Figure 2 - PIMA: Test error rate vs. the number of missing features for each instance to classify

Here also the classification error increases when the number of missing features increases for the instance to classify. But, unlike the Breast dataset, because the redundancy between the predictors is weak, the two approaches U2 (multivariate approach, in blue) and U1 (black) have similar behavior. We cannot use the other variables for an efficient imputation of the value of the predictors.

Waveform (2 classes). This is a binary variant of the WAVEFORM dataset. There are 33367 instances and 21 predictors. We use a small part of the dataset for the learning process (PROPORTION_TRAIN = 0.01). Because the size of the test sample is high, we obtain a really reliable estimation of the error rate here. We decrease the number of repetition of the experiment (N = 20).

```
R Console

> print(modele.glm)

Call:  glm(formula = classe ~ ., family = binomial, data = donnees$train)

Coefficients:
(Intercept)           V1           V2           V3           V4
   -7.41787      0.30895      0.12913      0.22713     -0.14564
         V5           V6           V7           V8           V9
   -0.74188     -0.03857      0.01179      0.90072      0.49528
        V10          V11          V12          V13          V14
    0.66153      1.34911      0.48427      0.23143     -0.70020
        V15          V16          V17          V18          V19
   -0.37999     -1.02268     -0.18765     -0.58620     -0.47339
        V20          V21
    0.01296      0.08828

Degrees of Freedom: 332 Total (i.e. Null);  311 Residual
Null Deviance:      461.6
Residual Deviance: 96.45        AIC: 140.5
>
```

The base error rate is **ERR.REF = 0.08797**. The predictors are highly redundant **(D = 0.000038755)**.

Because of this redundancy, the results are similar to those of the Breast dataset. The behavior of the two approaches becomes very different when we increase the number of missing features for each instance (Figure 3).

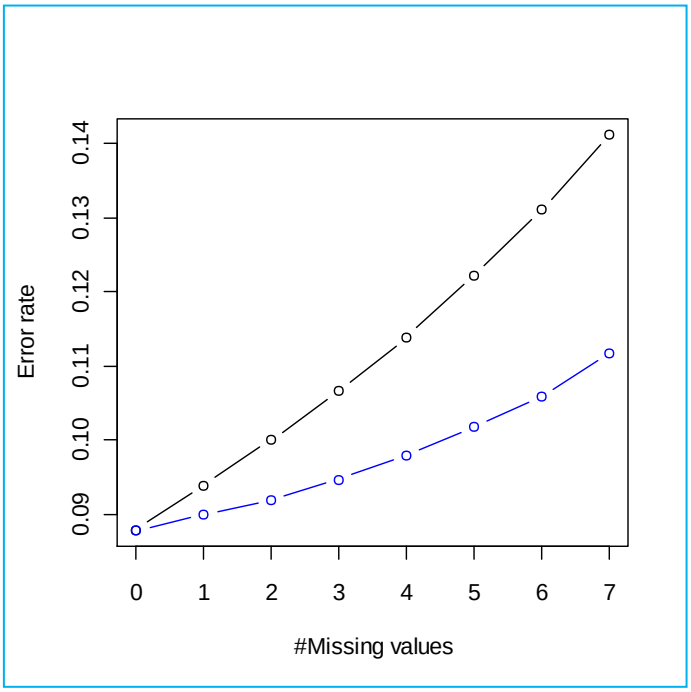

Figure 3 - WAVEFORM: Test error rate vs. the number of missing features for each instance to classify

We observe some interesting facts in the results of our experiment:

- These imputation techniques (U2 and U1) enable to overcome the missing feature problem when we apply the classification model on an unlabeled instance.

- But their efficiency decreases when the number of missing features increases.
- The multivariate approach, which uses the other variables, is better than the univariate one when the predictors are redundant. By contrast, the two methods have similar performances when the predictors are weakly correlated.

Anyway, these methods U1 and U2 at least have the merit to propose a practical solution for the missing feature problem during the prediction process.

4 Details of the program for the experiment

4.1 Conditions of use

The program that we describe in this section can be used if we fulfill the following requirements:

1. We have a binary class attribute.
2. The class attribute name must be “classe”.
3. All the predictors are continuous.
4. There are no missing values.

Then, the program enables to conduct the experiment described in this tutorial.

4.2 Settings

NOM_FICHIER enables to set the data file name. PROPORTION_TRAIN is the proportion of the dataset which is dedicated to the learning phase. M is the maximal number of missing features for each instance to classify. N is the number of repetition of each experiment to obtain a reliable value of the test error rate of each value “k” of missing features.

For instance, we set the following characteristics for the BREAST.TXT data file.

```
#data file name
NOM_FICHIER <- "breast.txt"
#proportion of the instances used for the learning phase
PROPORTION_TRAIN <- 0.50
#maximum number of missing features
M <- 7
#number of repetition for each experiment
N <- 200
```

4.3 Main steps of the program

4.3.1 Data partition, model learning phase, base error rate

We load the dataset. We subdivide it in train and test samples. We learn the model on the first one. Then we apply it on the second one (test sample without missing values) to obtain the base test error rate.

```
#set the working directory
setwd("...")
#load the data file
donnees.all <- read.table(file=NOM_FICHIER,sep="\t",header=T,dec=".")
#set the random number seed
```

```
set.seed(100)
#partitioning the data into train (donnees$train) and test (donnees$test) samples
donnees <- partition(donnees.all,prc=PROPORTION_TRAIN)
#construction of the model on the learning sample (donnees$train)
modele.glm <- glm(classe ~ ., data = donnees$train,family=binomial)
#calculation of the error rate on the test sample(donnees$test)
err.ref <- pred_and_confusion_matrix(donnees$test,modele.glm)
print(paste("Erreur sur observations completes =",as.character(err.ref)))
```

Two functions are written for this process: « **partition** » is used for the subdivision of the dataset.

```
#partitioning the dataset into train and test samples
#default proportion (train = 70%, test = 30 %)
partition <- function(donnees,prc=0.7){
      #samples size
      n <- nrow(donnees)
      n.train <- trunc(n*prc)
      n.test <- n - n.train
      #index
      alea <- runif(n)
      index <- rank(alea,ties.method="random")
      #subdivision
      donnees.train <- subset(donnees,subset=(index <= n.train))
      donnees.test <- subset(donnees,subset=(index > n.train))
      #return
      return(list(train=donnees.train,test=donnees.test))
}
```

« **pred_and_confusion_matrix** » enables to apply the model on the test set to obtain the prediction column. We derive the confusion matrix and the test error rate.

```
#calculation of the confusion matrix from a model
#new.dataset is the test set
#model.glm is the prediction model
pred_and_confusion_matrix <- function(new.dataset,model.glm){
      #predicted probabilities for each new instance
      data.pred.prob <- predict(model.glm,newdata = new.dataset)
      #class assignment
      data.pred.class <- ifelse(data.pred.prob > 0.5,"B","A")
      data.pred.class <- as.factor(data.pred.class)
      #confusion matrix (observed values vs. predicted values)
      mc <- table(new.dataset$classe,data.pred.class)
      #error rate
      err.rate <- 1.0 - (mc[1,1]+mc[2,2])/sum(mc)
      return(err.rate)
}
```

Note: This function fails if the model assigns the same value for all the instances. The dimension of the confusion matrix is different. The calculation of the error rate is not possible with our program.

4.3.2 Correlation matrix and the D criterion

All the predictors are numeric. We can compute the correlation matrix and its determinant D.

```
#predictive columns
donnees.train.numeric <- subset(donnees$train,select = -classe)
#correlation matrix and its determinant
d <- det(cor(donnees.train.numeric))
print(paste("Determinant matrice de correlation =",as.character(d)))
```

4.3.3 Imputation model

The U1 strategy is really simplistic: the imputation model is the mean of the variables computed on the learning sample. For the U2 approach, we must implement the multiple linear regression of each variable on the others.

```
#calculation of the mean of each variable
donnees.numeric.mean <- sapply(donnees.train.numeric,mean)
#multiple linear regression for each predictor
modeles.replace.by.reg <- lapply(donnees.train.numeric,modele.replace.by.reg.for.lapply,donnees.train.numeric)
```

I have tried two solutions for the U2 strategy. The first one is based on a loop which adds each imputation model in a list.

```
#construction of the list of imputation model
#based on a multiple linear regression
modele.replace.by.regression <- function(donnees){
      modele.replace <- list()
      for (j in 1:ncol(donnees)){
            formule <- paste(names(donnees)[j]," ~ .",sep="")
            modele.replace[[j]] <- lm(as.formula(formule),data=donnees)
      }
      names(modele.replace) <- names(donnees)
      return(modele.replace)
}
```

The second one uses the useful **lapply()** R function[6] with is more efficient if we deal with a large database (large number of predictors). It calls a call back function which implements the regression. The main (only) difficulty here is to obtain the variable number of the target into the regression[7].

```
#call back function for lapply()
#regression of x on the other variables
modele.replace.by.reg.for.lapply <- function(x,donnees){
      #obtaining the number of x into donnees
      numero <- as.numeric(gsub("\\D","", deparse(substitute(x)), perl=T))
      #regression of x on the other variables
      modele <- lm(x ~ ., data = subset(donnees,select = -numero))
      return(modele)
}
```

[6] http://www.ats.ucla.edu/STAT/r/library/advanced_function_r.htm

[7] http://forums.cirad.fr/logiciel-R/viewtopic.php?p=15851&sid=60c7e747fa7c9778c915ed91443a6615

The solution is more elegant and we will use it intensely in the imputation procedures developed subsequently.

4.3.4 The main loop of the experiment

We can start the experiments by varying the number of missing values for each individual to classify. For each value of "k", we repeat N times the experiment. The results are stacked into the 2-columns matrix "resultats".

```
#*********************************************************************
#launch the experiment
#k is the number of missing features for each instance to classify
#k = 1,..., M
#Repeat N times each experiment
#*********************************************************************

#the results are stored into resultats
resultats <- c()
#experiments (>> k << missing features for each instance)
for (k in 1:M){
      erreurs <- replicate(N,experiments(donnees$test,modele.glm,k))
      moyennes <- apply(erreurs,1,mean)
      resultats <- rbind(resultats,moyennes)
}
#add the result for the test set without missing values
resultats <- rbind(c(err.ref,err.ref),resultats)
rownames(resultats) <- 0:M
print(resultats)
#plotting
plot(0:M,resultats[,1],type="b",col="black",xlim=c(0,M),ylim=c(min(resultat
s),max(resultats)),xlab="#Missing values",ylab="Error rate")
points(0:M,resultats[,2],type="b",col="blue")
```

For each experiment session, we make the following steps: insert the missing values in each row of the test set; make the imputation with the first strategy U1; calculate the test error rate **e1** on the test set with imputed values; make the imputation with the second strategy U2; calculate the test error rate **e2**.

```
#one experiment
experiments <- function(donnees.test, modele.glm, k){
      #retrieve all the columns unless the class attribute
      donnees.numeric <- subset(donnees.test,select= -classe)

      #insert the missing values
      donnees.na <- insert.na.data.frame(donnees.numeric,k)

      #replace NA by means
      donnees.test.mean <- as.data.frame(lapply(donnees.na,replace.by.mean.one,donnees.numeric.mean))
      donnees.for.test <- cbind(donnees.test.mean,donnees.test$classe)
      names(donnees.for.test) <- names(donnees.test)
```

```
        e1 <- pred_and_confusion_matrix(donnees.for.test,modele.glm)

        #replace NA by regression models
donnees.test.reg <- as.data.frame(lapply(donnees.na,replace.by.reg.one,modeles.replace.by.reg,donnees.test.mean))
        donnees.for.test <- cbind(donnees.test.reg,donnees.test$classe)
        names(donnees.for.test) <- names(donnees.test)
        e2 <- pred_and_confusion_matrix(donnees.for.test,modele.glm)

        #return the error rates
        return(c(e1,e2))
}
```

4.3.5 Inserting the missing values into each row of the test set

The treatment is done row by row. We transform the data.frame in a matrix then we call the **apply()** function.

```
#insert randomly k NA in each row of the data.frame
#we consider that all the columns are numeric
insert.na.data.frame <- function(donnees,k){
        #transform the data.frame in a matrix
        #all the columns must be numeric!!!
        matrice <- as.matrix(donnees)
        #apply the modification for each row
        new.matrice <- apply(matrice,1,insert.na.row,k)
        #transposition of the matrix
        new.matrice <- t(new.matrice)
        #transformation in a data.frame
        new.donnees <- as.data.frame(new.matrice)
        return(new.donnees)
}
```

We use **insert.na.row(.)** to insert k missing values in each row. The insertion is made randomly for each row. Thus, different columns are concerned for the various rows of the dataset.

```
#inserting k NA randomly in a row
insert.na.row <- function(ligne,k){
        #number of column in the row
        J <- length(ligne)
        #number of the columns to modify
        index <- rank(runif(J))[1:k]
        #inserting the value NA
        y <- ligne
        y[index] <- NA
        return(y)
}
```

4.3.6 Univariate imputation

The function searches the number of the variable and then retrieves the corresponding mean into the “means” parameter (vector of the means for all the variables) of the function.

```
#replace the NA by the mean of the variable
replace.by.mean.one <- function(x, means){
      numero <- as.numeric(gsub("\\D","", deparse(substitute(x)), perl=T))
      y <- ifelse(is.na(x)==T, means[numero],x)
      return(y)
}
```

4.3.7 Multivariate imputation by regression

For the multivariate imputation, we need to the list of the imputation models and the means of all the variables (required if we have more than one missing values - see section 2.4).

```
#replace NA using a regression model
replace.by.reg.one <- function(x, modeles, donnees.mean){
      numero <- as.numeric(gsub("\\D","", deparse(substitute(x)), perl=T))
      pred <-predict(modeles[[numero]],newdata=subset(donnees.mean,select=-numero))
      y <- ifelse(is.na(x) == T,pred,x)
      return(y)
}
```

Here also we need the number of the variable to obtain the right model for the imputation.

5 Conclusion

In this tutorial, we studied the behavior of two imputation techniques when deploying a classification model on partially described individuals (some variables are missing). Their first merit is they are operational solutions. We note that the multivariate strategy, based on the regression, is more efficient than the univariate one when variables are correlated. Otherwise, when the predictors are weakly correlated, the two approaches give similar results.

However, experiments show that these solutions are reliable only if, for each individual to classify, the number of missing values remains relatively low compared with the overall number of variables.