1 Objectif

Déploiement d'un modèle prédictif sur des observations décrites de manière incomplète.

Le traitement des valeurs manquantes est un problème difficile, maintes fois étudié lorsqu'il s'agit d'analyser son impact sur les caractéristiques du modèle prédictif élaboré à partir des données d'apprentissage. Nous avons mené une expérimentation récemment[1]. Il s'agissait de comparer les mérites respectifs des différentes approches (suppression de lignes ou imputation par la moyenne/mode) sur les performances de la régression logistique. Il existe par ailleurs de très nombreuses références dont certaines sont accessibles sur le net[2].

Mais qu'en est-il lors du déploiement d'un modèle ? Curieusement, les écrits sont rares, voire très rares sur le sujet. Pourtant le problème est d'importance. Imaginons une situation concrète. Nous avons construit un super modèle à l'aide de la régression logistique. Nous l'intégrons dans notre système d'information. Une fiche client arrive, nous souhaitons le scorer pour connaître son appétence à un nouveau produit. Et là, patatras, la personne n'a pas mentionné son salaire dans sa fiche. Or, cette variable figure dans votre équation. Que faire ?

La suppression de lignes n'a absolument aucun sens dans ce cas. Cela reviendrait à refuser de classer un individu sous prétexte que les conditions d'application du modèle ne sont pas réunies. Si la posture est louable en théorie, elle n'est pas tenable en pratique. S'il fallait espérer être à chaque fois dans des conditions idéales pour agir, on n'a pas fini d'attendre.

Une autre solution simple consisterait à évacuer le problème en ignorant les variables à valeur manquante dans le modèle. Elle n'est absolument pas appropriée. En effet, le score serait systématiquement surévalué ou sous-évalué selon que les coefficients associés aux variables seraient positifs ou négatifs. Au final, la prédiction est biaisée.

Bref, il nous faut produire des valeurs de substitution pour le classement, sachant que **nous devons utiliser exclusivement les informations en provenance de l'échantillon d'apprentissage**. En effet, les observations à classer doivent être traités individuellement. L'échantillon test n'est pas exploitable comme un ensemble de données disponibles pour l'estimation des paramètres.

Dans ce tutoriel, nous supposons que le modèle prédictif a été construit selon un processus classique. La question des données manquantes n'est pas posée pour l'apprentissage. En revanche, elle est posée lors du déploiement. Nous souhaitons classer des individus dont la description est incomplète. Nous comparerons alors deux approches de substitution – l'une univariée, l'autre multivariée – de valeurs manquantes pour le déploiement. Nous montons une expérimentation sous R pour évaluer empiriquement leurs performances respectives sur plusieurs bases de données benchmark bien connues de la communauté du Data Mining.

Nous nous plaçons dans un cadre spécifique dans ce tutoriel : le modèle prédictif est issu de la régression logistique ; toutes les variables prédictives sont quantitatives ; la probabilité d'apparition d'une valeur manquante est la même pour toutes les variables décrivant l'individu à traiter.

[1] http://tutoriels-data-mining.blogspot.com/2011/12/donnees-manquantes-regression.html

[2] http://www.uvm.edu/~dhowell/StatPages/More_Stuff/Missing_Data/Missing.html

2 Deux techniques d'imputation pour le classement

Prenons un petit exemple pour préciser les idées.

	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

Construction du modèle prédictif. Nous souhaitons prédire l'occurrence d'une maladie à partir des caractéristiques des patients (âge, pression artérielle, taux de cholestérol). Nous obtenons le modèle prédictif suivant à l'aide de la régression logistique (BINARY LOGISTIC REGRESSION) de TANAGRA :

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

Le modèle dans sa globalité n'est pas significatif, aucune des variables ne l'est non plus. Ce n'est guère étonnant au vu de la très faible taille de l'échantillon (10 observations). Néanmoins, pour simplifier l'exposé, nous conservons le modèle tel quel.

Classement d'un individu. Le modèle à déployer s'écrit :

$$C = -11.8494 + 0.2933 \times \text{âge} - 0.0542 \times \text{pression} + 0.0152 \times \text{cholestérol}$$

Pour un individu ω à classer, si $C(\omega) > 0$, nous prédisons la présence de la maladie, sinon nous concluons à son absence.

Voyons ce qu'il en est pour un individu dont voici les caractéristiques :

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

Nous appliquons les coefficients de l'équation, nous obtenons :

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

Nous prédisons donc la « **présence** » de la maladie chez ce nouveau patient.

Description incomplète d'un individu à classer. Mettons maintenant que pris d'une coquetterie aussi subite qu'incompréhensible, notre patient ne veut pas avouer son âge. Nous nous retrouvons dans la configuration suivante :

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

Comment calculer la valeur de $C(\omega)$ à partir de cette description ? C.-à-d. sans disposer de la valeur de l'âge ?

2.1 Quelques mauvaises idées

Première attitude intenable disions-nous plus haut : décider qu'on ne peut rien faire pour ce patient. Le statisticien comprend très bien cette position. Après tout, on ne peut pas demander à un modèle des choses qu'il ne sait pas faire. Mais expliquer cela à un expert du domaine (un marketeur, un médecin, bref des personnes qui attendent des réponses concrètes) est bien difficile.

Deuxième solution tout aussi inappropriée : décider qu'on peut ignorer l'âge et appliquer les coefficients sur les variables restantes c.-à-d. calculer

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

Cela revient à attribuer l'âge **0** à l'individu. Le résultat est biaisé, nous concluons à l'absence de la maladie. Cette solution est d'autant plus imprécise qu'il se peut que la description soit manquante pour plusieurs variables (ex. âge ET cholestérol). Les calculs ne correspondent plus à rien dans ce cas.

Dans ce tutoriel, nous évaluons le comportement de deux solutions d'imputation plus ou moins sophistiquées. Nous en étudierons non pas les caractéristiques statistiques (biais et variance) mais plutôt les performances prédictives. Pour ce faire, nous monterons une expérimentation que nous détaillerons dans la section 3.

2.2 Solution univariée (U1)

Cette approche est très simple (simpliste). Elle consiste à remplacer la valeur manquante d'une variable par sa moyenne calculée sur l'échantillon d'apprentissage. Cela revient à neutraliser la variable dans la prédiction. Dans notre exemple ci-dessus, après avoir calculé les moyennes,

	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**	

Nous utilisons la description suivante pour la prédiction :

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

A la valeur manquante de l'âge, nous avons donc substituée sa moyenne calculée sur la base d'apprentissage, soit 54.1 (rappelons que l'âge réel de la personne est 51). En appliquant le modèle prédictif sur ces données, nous obtenons :

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

Nous nous rapprochons de la vraie valeur de C (sur cet exemple en tous les cas). Et la conclusion est conforme à celle calculée sur la description complète de l'individu.

Cas de plusieurs valeurs manquantes. Notons que cette solution est opérationnelle même lorsque plusieurs variables ne sont pas renseignées chez l'individu à classer. Il suffit de remplacer chaque valeur manquante par sa moyenne calculée sur l'échantillon d'apprentissage.

2.3 Solution multivariée (U2) : une seule valeur manquante

Les variables prédictives sont très rarement indépendantes deux à deux. Nous pouvons exploiter leurs liens mutuels pour mieux imputer la valeur prise par l'une d'entre elle pour un individu à classer. **Nous utilisons la régression linéaire multiple dans ce tutoriel**. Mais en réalité, nous pouvons utiliser tout autre technique telle que les arbres de régression (ou de décision si la variable à imputer est qualitative).

Bien évidemment, la stratégie sera d'autant plus performante que les variables sont fortement liées entre elles c.-à-d. redondantes dans le modèle servant à la prédiction de la maladie. Ce constat est un peu gênant. L'objectif est de produire des modèles prédictifs parcimonieux, avec le moins de colinéarité possible. La sélection de variables joue ce rôle là justement. Or, comme nous le montrera l'expérimentation par la suite, nous avons besoin de la redondance pour produire une imputation efficace lors du déploiement. Voilà un dilemme bien compliqué à gérer en vérité. Je me contente de le constater ici, la question reste totalement ouverte.

Reprenons notre exemple ci-dessus. L'individu à classer correspond à

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

Pour « deviner » la bonne valeur de l'âge, nous calculons au préalable la régression linéaire de l'âge sur l'ensemble des autres variables prédictives. En pratique, c'est ce que nous mettrons en place dans notre programme, toutes les régressions de chaque prédictive par rapport aux autres seront calculées une fois pour toutes, juste après la construction du modèle prédisant l'occurrence de la maladie, et stockées dans une liste.

L'équation de régression calculée sur notre échantillon d'apprentissage s'écrit :

Age = 0.0404 x cholestérol + 0.1482 x pression + 22.6974

Ainsi, la valeur imputée pour l'individu à classer est obtenue avec

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

L'estimation n'est pas très éloignée de la moyenne finalement, assez loin de la véritable valeur qui est de 51 rappelons-le.

Par la suite, pour l'individu ω, nous obtenons la valeur du LOGIT en utilisant la valeur imputée :

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

Mesurer la redondance entre les explicatives. Certes la prédiction est cohérente avec celle réalisée sur les données complètes. Mais on est un peu déçu quand même, on ne fait guère mieux que l'imputation à l'aide de la moyenne. Peut-être est-ce justement parce que les variables sont peu redondantes entre elles. La prédiction des valeurs de l'âge à partir des autres variables (et ainsi de suite, pression vs. les autres, cholestérol vs. les autres) est de mauvaise qualité.

Pour mesurer le lien entre les variables prédictives de la base, nous utilisons la matrice de corrélation. Et, effectivement, les liaisons deux à deux ne sont pas franchement marquées.

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

déterminant	0.7000

Pour évaluer le degré de liaison globale entre les variables, nous calculons le déterminant D de cette matrice. Plus sa valeur sera proche de 0 (zéro), plus forte sera la colinéarité entre les variables. Si les variables sont deux à deux orthogonales, le déterminant sera proche de 1. Dans notre exemple, avec **D = 0.7**, nous pouvons dire que les variables (âge, pression et cholestérol) sont faiblement redondantes. Nous utiliserons cette indicateur D pour évaluer l'efficacité des différentes stratégies d'imputation dans notre expérimentation.

2.4 Solution multivariée (U2) : plusieurs valeurs manquantes

Comment faire lorsque plusieurs valeurs sont manquantes ? Mettons que l'âge et le cholestérol ne sont pas renseignés chez l'individu à classer. Avec le système de régression, pour imputer l'âge, nous avons besoin de la valeur du cholestérol, et inversement. On ne s'en sort pas.

On peut toujours imaginer des systèmes sophistiqués. Dans ce didacticiel, nous avons choisi une solution simple : pour imputer la valeur d'une variable, nous utilisons les valeurs observées des autres variables ou, à défaut, de leur moyenne calculée sur l'échantillon d'apprentissage.

Mettons que nous souhaitons classer un individu décrit de la manière suivante :

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

Il nous faut tout d'abord imputer la valeur de l'âge. Pour ce faire, nous utilisons l'équation de régression

Age = 0.0404 x cholestérol + 0.1482 x pression + 22.6974

Nous ne disposons pas de la valeur du cholestérol. Nous utilisons la moyenne calculée sur l'échantillon d'apprentissage, à savoir 276.4. Ainsi, la valeur imputée de l'âge est :

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

Pour cholestérol, nous estimons son équation de régression linéaire sur les autres variables

Cholestérol = 0.2125 x pression + 4.6343 x age - 3.3389

L'âge n'étant pas disponible, nous utilisons sa moyenne 54.1. Ainsi la valeur imputée de cholestérol devient :

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

La description utilisée pour le classement sera donc :

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

Nous en déduisons le LOGIT

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

Nous prédisons la « présence » de la maladie chez le malade ainsi décrit.

Remarque. Notons que l'imputation par la moyenne est un cas particulier de l'imputation par la régression. En effet, le meilleur prédicteur endogène d'une variable est sa propre moyenne c.-à-d. dans la régression x = b + ε, l'estimateur des moindres carrés de la constante est $\hat{b} = \bar{x}$.

3 Evaluation expérimentale des solutions d'imputation

3.1 Principe de l'expérimentation

Pour évaluer l'efficacité de ces deux solutions, nous avons mis au point une expérimentation sur des bases ultra-connues du serveur UCI : Pima Indian Diabetes[3], Breast Cancer Winconsin[4], et une

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

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

variante binaire de Waveform[5]. Les valeurs manquantes ont été imputées en amont lorsque la base en comporte.

Nous effectuons alors la séquence de traitements suivante :

1. Charger la base.
2. La partitionner en échantillons d'apprentissage et de test selon une proportion à spécifier (PROPORTION_TRAIN).
3. Le modèle prédictif est construit sur l'échantillon d'apprentissage en utilisant la régression logistique. C'est le modèle (LOG.REG) que nous utilisons pour le déploiement.
4. LOGERG est appliqué sur l'échantillon test (toutes les observations étant complètement décrites). Nous mesurons le taux d'erreur de référence ERR.REF.
5. Dans l'échantillon test, nous insérons « k » valeurs manquantes par ligne. Les variables concernées sont choisies au hasard.
6. Classement des individus de l'échantillon test avec LOGR.EG en utilisant la technique d'imputation U1. Nous mesurons le taux d'erreur ERR.U1.
7. Classement des individus de l'échantillon test avec LOG.REG en utilisant la technique d'imputation U2. Nous mesurons le taux d'erreur ERR.U2.

L'imputation serait particulièrement efficace si ERR.U1 (ou ERR.U2) restait suffisamment proche de ERR.REF. Ce ne sera pas le cas. Les performances se dégradent fortement à mesure que l'on augmente « k », le nombre de valeurs manquantes pour un individu à classer. Pas de la même manière cependant chez U1 et U2, selon que les variables de la base sont corrélées ou non.

Pour éviter les artefacts, les étapes 5 à 7 sont répétées N fois (N =200 sauf mention contraire). Nous calculons les moyennes des erreurs ERR.U1 et ERR.U2 pour chaque valeur de « k ». Nous réduisons ainsi la variabilité associée à « k ».

Nous ne l'avons pas fait dans cette expérimentation, mais nous aurions pu tout aussi bien introduire la répétition des étapes 2 à 7, dans laquelle nous réitérons N fois les étapes 5 à 7. Nous tenons compte alors de la variabilité associée à la modélisation (nous testons différentes configurations des échantillons d'apprentissage et de test) ET associée à « k ». L'ennui est que les résultats deviennent très vite inextricables. Pour ceux qui souhaitent s'y lancer, l'adaptation du programme ne pose aucun souci quoiqu'il en soit.

Enfin, nous avons testé différentes valeurs de « k » allant de 1 à M (M = 7) pour détecter le seuil à partir duquel l'imputation n'est plus viable. A l'extrême, toutes les valeurs sont manquantes. Dans ce cas, tous les individus se voient imputées les mêmes valeurs.

3.2 Les principaux résultats

Breast Cancer Wisconsin. Elle comporte 699 observations et 9 variables prédictives. Nous l'avons scindé en 2 parties égales (PROPORTION_TRAIN = 0.5). Sur l'échantillon d'apprentissage a été élaboré le modèle suivant :

[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
>
```

Le taux d'erreur de référence obtenue sur l'échantillon test initial est **ERR.REF = 0.042857**. Les variables prédictives sont fortement liées entre elles, le déterminant de la matrice de corrélation est égale à **D = 0.000699**.

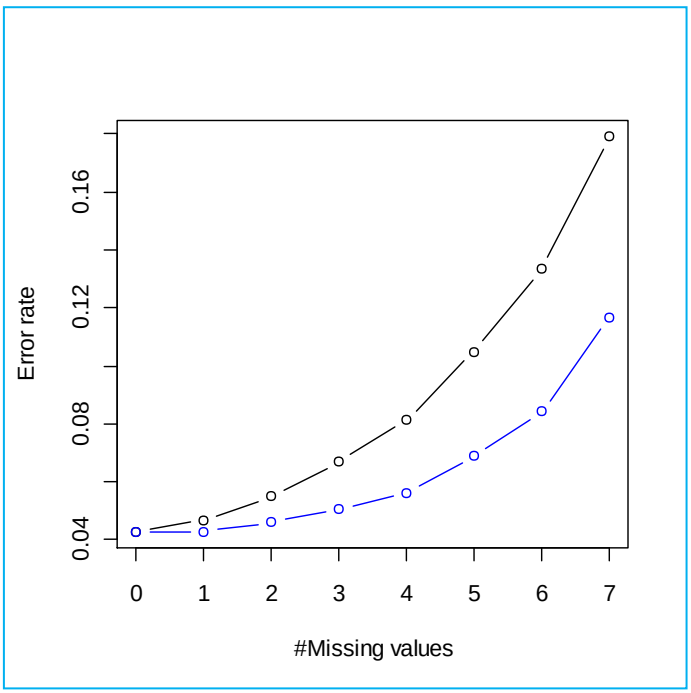

Figure 1 - BREAST : Taux d'erreur vs. le nombre de valeurs manquantes pour chaque individu à classer

Clairement, l'imputation permet de pallier la présence de valeurs manquantes en classement. Mais elle est de moins en moins efficace à mesure que leur nombre « k » augmente (Figure 1). Ce n'est guère étonnant. Classer les individus à partir d'une description incomplète est difficile. La tâche le sera d'autant plus que les informations manquent. Nous remarquons néanmoins que la stratégie **U2** (courbe bleue, basée sur l'imputation par régressions croisées) s'avère nettement meilleure qu'**U1** (courbe noire, basée sur la moyenne) pour les valeurs élevées de « k ». Elle bénéficie de la redondance entre les variables prédictives, l'imputation est de meilleure qualité.

Pima Indian Diabete. Elle contient 8 variables prédictives et 768 observations. Nous avons fixé PROPORTION_TRAIN à 0.5. Le modèle prédictif s'écrit :

```
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
>
```

Le taux d'erreur de référence est **ERR.REF = 0.2421875**. Les variables prédictives sont faiblement redondantes avec **D = 0.257740**.

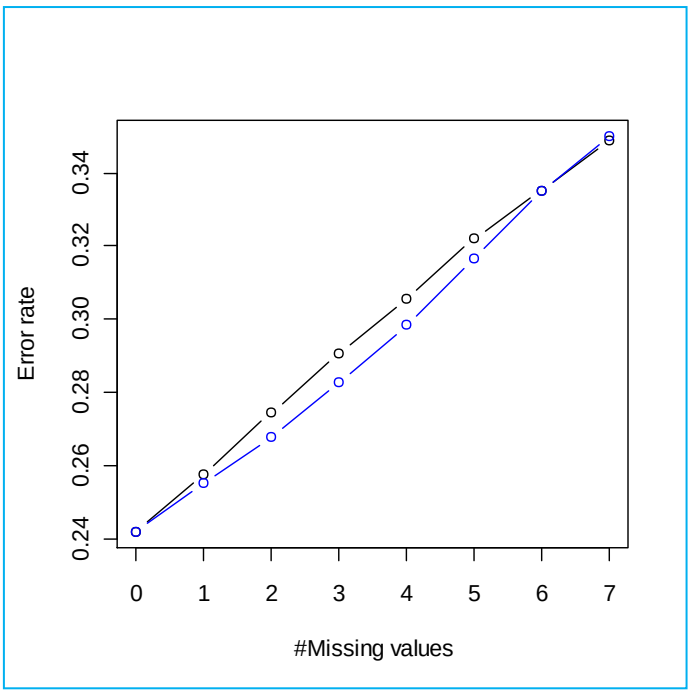

Figure 2 - PIMA : Taux d'erreur vs. le nombre de valeurs manquantes pour chaque individu à classer

Ici également la dégradation est manifeste mais, à la différence de la base BREAST, elle est identique pour les deux stratégies U1 (courbe noire) et U2 (courbe bleue). Il faut y voir l'impact de la faible liaison entre les variables prédictives du modèle. La régression n'est pas en mesure d'exploiter les informations en provenance des autres variables pour imputer efficacement.

Waveform (2 classes). Il s'agit d'une version binaire de la base WAVEFORM, avec 21 variables prédictives et 33367 observations. Nous en utilisons une très faible fraction PROPORTION_TRAIN = 0.01 pour l'apprentissage. La taille de l'échantillon test étant importante, l'estimation de l'erreur est plus précise, nous réduisons le nombre de répétitions (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
>
```

Le taux d'erreur de référence est **ERR.REF = 0.08797**. Les variables prédictives sont fortement redondantes avec un déterminant de la matrice de corrélation égale à **D = 0.000038755**.

Les variables prédictives étant fortement liées entre elles dans la base WAVEFORM, nous retrouvons logiquement des résultats similaires à ceux de la base BREAST. La dégradation de l'erreur est toujours aussi patente avec l'augmentation de « k ». En revanche la stratégie U2 (courbe bleue) basée sur les régressions croisées est meilleure qu'U1 (courbe noire). L'écart est plus sensible pour les valeurs élevées de « k » (Figure 3).

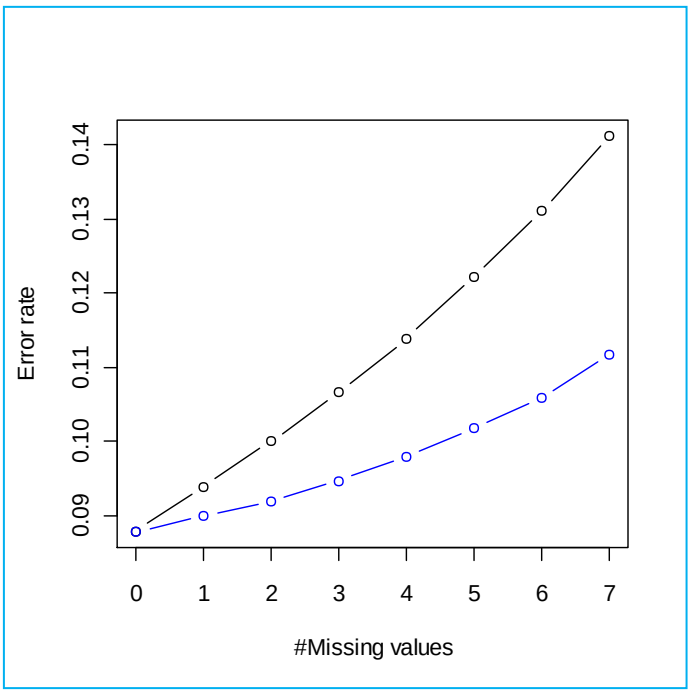

Figure 3 - WAVEFORM : Taux d'erreur vs. le nombre de valeurs manquantes pour chaque individu à classer

Au final, plusieurs éléments retiennent notre attention dans les expérimentations :

- Les techniques d'imputation U1 et U2 permettent de pallier les valeurs manquantes lors du classement d'un nouvel individu à l'aide du modèle prédictif.
- Mais elles sont de moins en moins efficaces lorsque leur nombre augmente.
- La stratégie U2 basée sur les régressions croisées s'avère meilleure qu'U1 lorsque les variables sont fortement liées entre elles. Elle (U2) est nettement moins décisive en revanche lorsque les variables prédictives du modèle sont peu redondantes.

Quoiqu'il en soit, ces deux approches U1 et U2 ont le mérite de proposer une solution concrète à un problème étrangement très peu étudié dans la littérature.

4 Détail du programme sous R

4.1 Spécifications

Dans cette section, nous décrivons le programme utilisé pour évaluer l'efficacité des techniques d'imputation en déploiement. Il est fonctionnel pour différentes bases de données pourvu que :

1. La variable à prédire soit binaire.
2. Elle soit située en dernière position (dernière colonne) dans la base.
3. Elle s'appelle forcément « classe ».
4. Les variables prédictives soient toutes quantitatives.
5. La base ne comporte pas de valeurs manquantes.

Ces spécifications respectées, le programme devrait fonctionner correctement. Il fournit des résultats permettant de comparer les performances respectives des deux méthodes d'imputation analysées dans ce didacticiel.

4.2 Paramètres

Le programme est paramétré par le nom du fichier de la base traitée (NOM_FICHIER), par la proportion des observations dévolues à l'apprentissage (PROPORTION_TRAIN), le nombre maximal de valeurs manquantes pour chaque individu à classer (M), le nombre de répétitions (N) pour chaque valeur « k » de valeurs manquantes.

Par exemple, nous avons utilisé les valeurs suivantes pour la base BREAST.TXT.

```
#nom du fichier à charger
NOM_FICHIER <- "breast.txt"
#proportion des observations en apprentissage
PROPORTION_TRAIN <- 0.50
#nombre max. de valeurs manquantes par ligne
M <- 7
#nombre de répétitions de chaque configuration
N <- 200
```

4.3 Principales étapes du programme

4.3.1 Partition des données, construction du modèle et calcul de l'erreur de référence

Dans un premier temps, nous chargeons la base, nous la scindons en deux parties. Sur la première, nous construisons le modèle prédictif ; sur la seconde, nous appliquons le modèle pour obtenir le taux d'erreur de référence, lorsque les individus à classer sont tous décrits de manière complète.

```
#changement du répertoire courant
setwd("... votre répertoire des données ...")
#chargement des données
donnees.all <- read.table(file=NOM_FICHIER,sep="\t",header=T,dec=".")
#initialiser le générateur de nombres aléatoires
set.seed(100)
#partition en apprentissage-test (-> donnees$train et donnees$test)
donnees <- partition(donnees.all,prc=PROPORTION_TRAIN)
#construction du modele sur les données d'apprentissage (donnees$train)
modele.glm <- glm(classe ~ ., data = donnees$train,family=binomial)
#mesurer le taux d'erreur de reference (données complètes : donnees$test)
err.ref <- pred_and_confusion_matrix(donnees$test,modele.glm)
print(paste("Erreur sur observations completes =",as.character(err.ref)))
```

Deux procédures sont intégrées dans ce processus. La première, « **partition** », se charge de subdiviser les données en deux parties.

```
#partition des données en apprentissage et test
#répartiton par défaut (apprentissage = 70%, test = 30 %)
partition <- function(donnees,prc=0.7){
	#effectifs
	n <- nrow(donnees)
	n.train <- trunc(n*prc)
	n.test <- n - n.train
	#index
	alea <- runif(n)
	index <- rank(alea,ties.method="random")
	#partition
	donnees.train <- subset(donnees,subset=(index <= n.train))
	donnees.test <- subset(donnees,subset=(index > n.train))
	#retour
	return(list(train=donnees.train,test=donnees.test))
}
```

La seconde « **pred_and_confusion_matrix** » se charge d'appliquer le modèle prédictif sur l'échantillon test pour obtenir la prédiction, de construire la matrice de confusion et de fournir le taux d'erreur.

```
#construction matrice de confusion à partir du modèle
#new.dataset représente l'échantillon test
#model.glm est le modèle
pred_and_confusion_matrix <- function(new.dataset,model.glm){
	#probabilité prédite par le modèle
```

```
      data.pred.prob <- predict(model.glm,newdata = new.dataset)
      #affectation
      data.pred.class <- ifelse(data.pred.prob > 0.5,"B","A")
      data.pred.class <- as.factor(data.pred.class)
      #matrice de confusion (classe observée vs. prédite)
      mc <- table(new.dataset$classe,data.pred.class)
      #taux d'erreur
      err.rate <- 1.0 - (mc[1,1]+mc[2,2])/sum(mc)
      return(err.rate)
}
```

Remarque : Attention, la procédure peut planter si le modèle prédit systématiquement une seule valeur. La dimension de la matrice de confusion est incorrecte, le taux d'erreur n'est plus calculable à l'aide de la formule utilisée. Cette configuration peut survenir lorsque nous avons inséré trop de valeurs manquantes dans la description de chaque observation de l'échantillon test, la prédiction n'est plus possible.

4.3.2 Calcul de la matrice de corrélation et du critère D

Nous restreignons la base aux prédictives, forcément toutes numériques dans notre configuration, pour calculer la matrice de corrélation et en déduire le déterminant.

```
#travailler sur les variables prédictives
donnees.train.numeric <- subset(donnees$train,select = -classe)
#calculer le déterminant de la matrice de corrélation
#==> indique le degré de colinéarité entre les variables explicatives
d <- det(cor(donnees.train.numeric))
print(paste("Determinant matrice de correlation =",as.character(d)))
```

4.3.3 Construction des modèles d'imputation

Nous devons maintenant calculer les modèles d'imputation à partir de l'échantillon d'apprentissage. Pour U1, il s'agit simplement de la moyenne par variable. Pour U2, il s'agit d'implémenter la régression de chaque variable sur toutes les autres.

```
#calculer les moyennes par colonne - échantillon d'apprentissage
donnees.numeric.mean <- sapply(donnees.train.numeric,mean)
#construire les modèles de remplacement par la régression
#modeles.replace.by.reg <- modele.replace.by.regression(donnees.train.numeric)
modeles.replace.by.reg                                                    <-
lapply(donnees.train.numeric,modele.replace.by.reg.for.lapply,donnees.train
.numeric)
```

Pour U1, un simple **sapply(.)** suffit amplement. Nous obtenons le vecteur des moyennes.

Pour U2, j'ai testé deux approches différentes. La première est basée sur une boucle ultra-classique. A chaque itération, une des variables devient l'endogène. Voici le code :

```
#modele de prédiction par la régression
#pour remplacer les valeurs d'une variable à partir des variables
#utilisation très classique d'une boucle
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)
}
```

Mais cette solution ne me satisfaisait pas. Elle ne fait pas très R dans sa philosophie (« *horreur, une boucle, il a fait une boucle… » diraient mes étudiants…*). Et surtout, je me suis rendu compte que j'allais rééditer ce genre de code – assez maladroit – pour les procédures d'imputation à venir.

Il faut plutôt passer par un **lapply(.)**. L'ennui est que nous n'avons pas accès au nom de la variable traitée par la fonction intégrée dans l'appel **lapply(.)**. On se retrouve alors à effectuer une régression d'une variable sur les autres, y compris elle-même. Ce qui ne sert absolument à rien.

En cherchant un peu sur le web, je me suis rendu compte qu'il est en revanche possible de récupérer le numéro d'une variable au sein d'un data.frame dans les appels lapply(.)[6]. Ni une ni deux, j'ai intégré l'astuce dans la fonction permettant d'implémenter la régression d'une variable sur tous les autres.

```
#modèle de prédiction par la régression
#pour remplacer les valeurs d'une variable à partir des variables
#fonction pour appel de lapply
modele.replace.by.reg.for.lapply <- function(x,donnees){
      #récupérer le numéro de x dans le data.frame
      #cf. référence web
      numero <- as.numeric(gsub("\\D","", deparse(substitute(x)), perl=T))
      #régression de x contre toutes les autres
      modele <- lm(x ~ ., data = subset(donnees,select = -numero))
      return(modele)
}
```

La solution est autrement plus élégante et, surtout, nous l'exploiterons intensivement par la suite, lors de la programmation des procédures d'imputation.

4.3.4 Boucle de gestion de l'expérimentation

Nous sommes parés, nous pouvons lancer l'expérimentation en faisant varier le nombre de valeurs manquantes pour chaque individu à classer. Elle s'écrit :

```
#*********************************************************************
#lancer l'expérimentation
#pour un nombre de valeurs manquantes allant de 1 à M pour chaque ligne
#de l'échantillon test
#>> répeter N fois l'opération
#*********************************************************************

#pour collecter les résultats
```

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

```
resultats <- c()
#expérimentation (s'il y a >> j << valeur manquante dans chaque ligne)
for (k in 1:M){
      erreurs <- replicate(N,experiments(donnees$test,modele.glm,k))
      moyennes <- apply(erreurs,1,mean)
      resultats <- rbind(resultats,moyennes)
}
#rajouter les résultats pour données complètes
resultats <- rbind(c(err.ref,err.ref),resultats)
rownames(resultats) <- 0:M
print(resultats)
#graphique
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")
```

Pour chaque valeur de « k », nous réitérons N fois la session d'expérimentation. Les résultats sont empilées au fur et à mesure dans la matrice à 2 colonnes « résultats ».

Pour chaque session, nous réalisons les opérations suivantes : insérer des valeurs manquantes dans chaque ligne de l'échantillon test ; imputer les valeurs manquantes par la moyenne, calculer le taux d'erreur (**e1**) du modèle en prédiction sur ces données imputées ; imputer les valeurs manquantes par la régression, calculer le taux d'erreur (**e2**).

```
#monter une expérimentation
experiments <- function(donnees.test, modele.glm, k){
      #récupérer les colonnes sauf la classe
      donnees.numeric <- subset(donnees.test,select= -classe)

      #insérer les données manquantes
      donnees.na <- insert.na.data.frame(donnees.numeric,k)

      #remplacement des NA par la moyenne
      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)

      #remplacement des NA par la régression
      donnees.test.reg                                                     <-
as.data.frame(lapply(donnees.na,replace.by.reg.one,modeles.replace.by.reg,d
onnees.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)

      #renvoyer le vecteur de 2 valeurs
      return(c(e1,e2))
}
```

Voyons le détail des trois grandes opérations qui jalonnent l'expérimentation.

4.3.5 Insertion des valeurs manquantes dans les lignes de l'échantillon test

Le traitement devant s'effectuer ligne par ligne, nous transformons le data.frame en matrice puis nous faisons appel à **apply(.)** sur la première dimension.

```
#insère k NA par ligne dans un data.frame
#on considère que toutes les colonnes sont numériques
insert.na.data.frame <- function(donnees,k){
	#transformer les données en matrice
	#attention, il faut que toutes les colonnes soient numériques
	#dans le cas contraire, nous aurons une matrice de chaînes de carac.
	matrice <- as.matrix(donnees)
	#appliquer la modification - traitement par ligne
	new.matrice <- apply(matrice,1,insert.na.row,k)
	#transposer la matrice
	new.matrice <- t(new.matrice)
	#transformer en data.frame
	new.donnees <- as.data.frame(new.matrice)
	return(new.donnees)
}
```

Nous utilisons la fonction **insert.na.row(.)** pour insérer k valeurs manquantes dans chaque ligne de la matrice. Attention, l'incorporation de NA est réalisée aléatoirement. Différents colonnes seront affectées d'une ligne à l'autre de notre ensemble de données.

```
#insère k NA au hasard dans une ligne
insert.na.row <- function(ligne,k){
	#nombre de colonnes dans la ligne
	J <- length(ligne)
	#index des colonnes à modifier
	index <- rank(runif(J))[1:k]
	#inserer le code NA
	y <- ligne
	y[index] <- NA
	return(y)
}
```

4.3.6 Imputation par la moyenne

L'imputation par la moyenne repère le numéro de colonne de la variable à traiter et récupère la moyenne dans le vecteur passé en paramètre de la fonction.

```
#remplacement des valeurs manquantes de x par la moyenne
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 Imputation par la régression

Pour l'imputation par régression, nous avons besoin de la liste des modèles et des données imputées par la moyenne. Elles sont nécessaires si plusieurs valeurs sont manquantes pour la ligne à traiter (voir section 2.4).

```
#remplacement des valeurs manquantes par la régression
#on s'appuie sur les valeurs des colonnes où les NA ont été remplacées par
#les moyennes
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)
}
```

Ici également, nous avons besoin de récupérer le numéro de la variable traitée par la fonction via l'appel à **lapply(.)** pour accéder au bon modèle de régression.

5 Conclusion

Dans ce tutoriel, nous avons étudié le comportement de deux techniques d'imputation lors du déploiement d'un modèle prédictif sur des individus décrits partiellement. Leur premier mérite est de proposer une solution directement opérationnelle. Nous constatons que la stratégie multivariée, basée sur la régression croisée, s'avère plus performante lorsque les variables sont corrélées. Dans le cas contraire, elle ne fait pas mieux que l'imputation univariée, basée sur la moyenne.

Que faire lorsque les variables prédictives du modèle comportent des variables qualitatives ? Une extension simple serait d'utiliser le mode (resp. la moyenne) pour les descripteurs qualitatifs (resp. quantitatifs) pour la stratégie univariée. Concernant l'imputation multivariée, nous pourrons nous tourner vers des techniques dont les conditions d'application sont plus larges telles que les arbres de décision et de régression.

Néanmoins, les expérimentations le montrent, ces palliatifs ne sont viables que si, pour chaque individu à traiter, le nombre de valeurs manquantes reste relativement faible par rapport au nombre de variables du modèle.