Scénario : modélisation de la perte en eau par torréfaction

Résumé

Modélisation avec SAS par régression linéaire simple puis multiple puis par analyse de covariance avec sélection de variables et interactions de la perte en eau du café vert après torréfaction. Essais de la procédure non linéaire GAM. Modélisation avec R et comparaison des qualités de prévision de ces différents modèles sur un échantillon test avec ceux d'un arbre de régression et d'un réseau de neurones. Il s'agit d'une introduction au statistical learning.

1 Introduction

1.1 Objectif

Ce scénario retrace les aventures d'El Ringo en Statistique. El Ringo achète du café vert dans le monde entier avant de le torréfier puis le redistribuer. Son problème est de prévoir la perte de poids due à la torréfaction. Cette perte, qui peut atteindre 20%, conditionne directement sa marge bénéficiaire, elle doit être estimée le plus précisément possible au moment de l'achat afin de pouvoir négocier le prix au plus juste. Son nez, légendaire lors de la torréfaction, est inefficace sur du café vert et la mesure directe du taux d'humidité est trop longue et trop coûteuse. El Ringo fait l'acquisition d'un chromatographe qui peut lui fournir rapidement 5 indicateurs numériques à partir d'un échantillon. Pour tester son approche, il réalise 189 expériences sur des échantillons de diverses provenances de café vert (Arabie, Afrique, Amérique... codées de 1 à 7) et construit un tableau contenant, pour chaque échantillon, les mesures chromatographiques sur le café vert (lumin, xa, xb, xy, xgn) et la perte de poids après torréfaction. L'objectif est de construire un bon modèle de prévision de cette perte de poids. La variable `humid`, mesurant un taux d'humidité, n'est pas prise en compte et supprimée.

1.2 Données

1.3 Déroulement

L'histoire est présentée selon une progression "pédagogique" à complexité croissante : régression simple, régression multiple, analyse de covariance avec ou sans interaction, sélection de variables, modèles non linéaires. Ce n'est pas la façon "nomimale" de conduire une analyse statistique qui nécessiterait d'étudier tout de suite le modèle complet (analyse de covariance) correspondant précisément à l'expérimentation réalisée (*experimental design*) aavant d'aborder éventuellement des approches non linéaires.

Répondre aux questions successives à l'aide des sorties des logiciels SAS ou R.

2 Étude préalable

2.1 Lecture des données

Prise en charge des données : charger les données du fichier `cafe.dat` à partir de l'URL : `http://wikistat/data/`

Exécuter le programme de lecture :

```
data sasuser.cafe;
infile "cafe.dat";
input origine humid perte lumin xa xb xy xgn ;
drop humid;
run;
```

2.2 Exploration des données

```
solutions > Analysis > Interactive Data analysis
sasuser
procpin > open
```

En cliquant sur `Int` au dessus de `origine`, déclarer la variable `origine` `Nominal` plutôt que `Interval` puis analyser brièvement les distributions des variables (symmétrie, valeurs atypiques) :

```
Analyse > Distribution(Y) > lumin > Y > OK
```

Tester la normalité de certaines avec la fenêtre `perte` active :

```
Tables > Test for normality
```

Calculer la matrice des corrélations :

```
Analyse > Multivariate > tout sélect. (sauf origine) >
   Y > OK
```

Tracer la matrice des nuages de points (*scatter plot matrix*) :

```
Analyse > Scatter plot > tout sélec. (sauf origine)
   Y tout sélec. > X > OK
```

Colorier les points par origine de café :

```
Edit > Windows > Tools
bouton Arc en ciel >  origine > OK
```

Commenter l'importance des corrélations, la forme des nuages, la répartition des couleurs.

Calculer et représenter l'analyse en composantes principales :

```
Analyse>Multivariate> tout selec. sauf origine >Y
Output>Principal Component Options
Principal component options >
   Biplot (Std Y)>OK>OK>OK
```

Que dire de la forme général du nuage ?

3 Modèles de régression

3.1 Régression simple

Se souvenant de quelques cours de Statistique et utilisant son tableur préféré, El RIngo tente d'expliquer la `perte` par la variable `lumin` par un modèle de régression linéaire simple. Pour aller plus vite, les calculs sont faits directement avec SAS.

```
Analyse>fit, perte dans Y, lumin dans X
Output > Residual Normal QQ
Output variables>Studentized residuals>
    Cook's D >OK>OK>OK
```

Vérifier pour chacun d'eux les hypothèses de
- linéarité (graphe des résidus)
- homoscédasticité (graphe des résidus)
- normalité des résidus (droite de Henri, test de la variable)
- contrôler l'existence ou non de points influents (graphe distance de Cook)

Que dire de la linéarité de ce modèle ? Le problème était déjà mis en évidence par la matrice des nuages de points. El Ringo se rend compte qu'il est allé trop vite et demande conseil au biologiste de son équipe. Ce dernier lui suggère de transformer la variable `perte` par la fonction `1/Y` :

```
Edit > Variables > 1/Y > lumin > Y > Name > Tlumin
```

Observer la distribution de cette variable ainsi que le nuage de point `origine x Tlumin`. Ce choix semble-t-il raisonnable.

Etudier comme précédemment la régression de `perte` sur `Tlumin`. Que devient le nuage des résidus ? Etudier leur normalité (distribution, test de `RNperte`. Que dire de la validité de ce modèle ?

3.2 Régression multiple

El Ringo rencontre l'économètre des services financiers à la machine à café. Celui-ci lui conseille d'arrêter le "bricolage" et de se doter d'un vrai logiciel statistique pour prendre en compte toute l'information disponible. El Ringo fait l'acquisition de SAS pour étudier un modèle de régression multiple. Reconsidérant la matrice des nuages de points il ajoute les transformées `Txy` et `Txgn` des variables `xy` et `xgn`.

```
data sasuser.cafe;
set sasuser.cafe;
Tlumin=1/lumin;
Txy=1/xy;
Txgn=1/xgn;
run;
```

Premier modèle

Estimer(`Analyse>fit` le modèle complet expliquant la `perte` par les variables `lumin xa xb xy xgn Tlumin Txy Txgn` sélectionnées en `X`.

Vérifier les hypothèses et diagnostics (linéarité, homoscédasticité, normalité des résidus, points influents) comme pour la régression simple en tenant compte de l'effectif de l'échantillon.

Que dire de la qualité d'ajustement ? Comparer à celle de la régression simple. Que signifient les `VIF` et les diagnostics de colinéarité ? Etudier les p-valeurs des différents tests, que conclure ?

4 Sélection de variable en régression multiple

El Ringo comprend qu'il a atteint son niveau d'incompétence et décide de faire appel à un stagiaire statisticien niveau M1. Celui-ci continue le travail par une sélection de variables afin de prendre en compte les problèmes de multicolinéarité.

La première procédure de sélection est manuelle afin de comprendre le processus.

4.1 Choix de modèle manuel

Après avoir estimé comme précédemment le modèle linéaire complet avec `lumin, xa, xb, xy xgn, Tlumin, Txy, Txgn` comme variables explicatives, itérer la procédure suivante dans SAS/INSIGHT :

1. Choisir, parmi les variables explicatives, celle X^j pour lequel le test de Student ($H_0 : b_j = 0$) est le moins significatif, c'est-à-dire avec la plus grande "prob value".

2. La retirer du modèle et recalculer l'estimation. Il suffit pour cela de sélectionner le nom de la variable dans le tableau (TYPE III) et d'exécuter la commande `delete` du menu `edit` de la même fenêtre. Le modèle est ré-estimé automatiquement.

Arrêter le processus lorsque tous les coefficients sont considérés comme significativement différents de 0. Attention, la "variable" INTERCEPT (terme constant) ne peut pas être considérée au même titre que les autres variables ; la *conserver* toujours dans le modèle.

Noter le modèle finalement obtenu, son coefficient de détermination à comparer à celui du modèle complet.

4.2 Procédures automatiques

La procédure précédente est innefficace pour des modèles complexes c'est-à-dire avec un nombre important de varaibles ; tester les procédures automatiques. Fermer SAS/INSIGHT puis exécuter successivement les programmes ci-dessous afin de comparer les modèles obtenus par différentes procédures de sélection : descendante, ascendante, pas à pas.

```
proc reg data=sasuser.cafe;
model perte =  lumin--Txgn/ selection=backward;
run;
proc reg data=sasuser.cafe;
model perte =  lumin--Txgn / selection=forward;
run;
proc reg data=sasuser.cafe;
model perte =  lumin--Txgn /  selection=stepwise;
run;
```

4.3 Optimisation globale

les algorithmes précédents peuvent, ou pas, passer par le "meilleur" modèle. Parmi les trois types d'algorithmes disponibles dans SAS et les différents critères de choix, une des façons les plus efficaces consistent à choisir les options du programme ci-dessous. Tous les modèles (parmi les plus intéressants selon l'algorithme de Furnival et Wilson) sont considérés. Seul le meilleur pour chaque niveau, c'est-à-dire pour chaque valeur q du nombre de variables explicatives sont donnés. Il est alors facile de choisir celui minimisant l'un des critères globaux (C_p, BIC...) estimant un risque pénalisé. Cette procédure de recherche d'un modèle optimal global n'est exécutable que pour un nombre raisonnables de variables, disons moins d'une vingtaine.

```
proc reg data=sasuser.cafe;
model perte = lumin -- Txgn
   / selection=rsquare cp adjrsq bic best=1;
run;
```

Sélectionner le modèle de C_p minimum et celui de R^2 ajusté maximum. Comparer les sélections obtenues.

4.4 Comparaisons

Parmi les sélections précédentes, déterminer quelles sont celles différentes : choix descendant, descendant, stepwise, C_p minimum, R^2 ajusté maximum. Les paramètres des algorithmes (seuils de significativité des tests) sont ceux par défaut mais peuvent être modifiés. Recalculer chacun de ces modèles :

```
proc reg data=sasuser.cafe;
model perte = lumin -- Txgn
     / collin r p ;
run;
proc reg data=sasuser.cafe;
model perte = lumin xa xb xy Txgn
     / collin r p ;
run;
proc reg data=sasuser.cafe;
model perte =   xa xgn Txgn
     / collin r p ;
run;
```

Comparer dans un tableau les valeurs obtenues des coefficients de détermination et de PRESS. Quel modèle choisir pour la prévision ?

5 Analyse de covariance

5.1 Modèle élémentaire

En reprenant le problème, le stagiaire se rend compte que la variable d'origine du café n'a pas été prise en compte. Il s'interroge donc sur la pertinence de l'introduire dans le modèle. Il refait le nuage des points croisant `perte x Tlumin`. Que penser de l'influence de l'origine du café ?

Vérifier cette hypothèse en testant dans SAS/INSIGHT le modèle :

```
fit > perte en Y, Tlumin et origine en X > OK
```

5.2 Modèle avec interactions

Le stagiaire s'interroge également sur l'influence possible des interactions mais une approche manuelle de sélection de modèle avec SAS/INSIGHT à partir du modèle de toutes interactions d'ordre 2 devient ingérable. La procédure `reg` qui fait de la sélection de modèle n'est pas adaptée à l'analyse de covariance mais celle `glmselect`, plus récente, l'est. Elle intègre plusieurs procédures de sélection classique (`backward, forward, stepwise`) et aussi moins classiques : `lasso, lars`

```
proc glmselect data=sasuser.cafe;
class origine;
/* modèle de toutes interactions d'ordre 2*/
model perte =   origine | lumin | xa | xb | xy | xgn
     | Tlumin | Txy | Txgn @2
     /  details=all stats=all choose=press
                    selection=backward;
run;
proc glmselect data=sasuser.cafe;
class origine;
model perte =   origine | lumin | xa | xb | xy | xgn
     | Tlumin | Txy | Txgn @2
     /  details=all stats=all choose=press
                    selection=stepwise;
run;
```

Comparer les valeurs de PRESS obtenus avec ces modèles et ceux précédents. Que conclure ?

5.3 Vers le non linéaire

En consultant la documentation de SAS, la stagiaire découvre l'existence de quelques autres procédures de modélisation. Compte tenu des possibles relations non linéaires entre les variables, il teste un modèle additif généralisé (GAM). Ce modèle recherche une combinaison linéaires de transformations

non paramétriques “optimales” des variables , transformations définies par des fonctions spline plutôt qu’imposées sous une forma analytique une comme 1/Y. Le modèle est plus complexe (transformation des variables) mais plus simple que ceux avec interactions par le nombre des termes. Remarque : il est également possible de prendre en compte des “interactions” dans un tel modèle par l’introduction de transformations / surfaces spline de deux variables.

```
proc gam data=sasuser.cafe;
class origine;
model perte =  param(origine) spline(lumin)
   spline(xa) spline(xb) spline(xy) spline(xgn);
run;
```

Le modèle ainsi ontenu est-il meilleur que les précédents ? Difficile à dire car le critère PRESS utilisé jusqu’à présent n’est pas fourni par cette procédure. Poursuivant ses investigations, le stagiaire M1 a entendu parlé d’autres méthodes non linéaires comme les réseaux de neurones ou les arbres binaires (CART) mais celles-ci ne sont disponibles que dans le module *Enterpise Miner*.

Considérant les résultats obtenus à l’issue du stage et le prix d’*Enterprise Miner*, El Ringo est bien embarrassé. Il va donc utiliser un modèle d’analyse de covariance avec interactions, même s’il n’est sans doute pas optimal, pour sa prochaine campagne et rechercher par ailleurs un stagiaire niveau M2 pour continuer le travail.

6 Recherche d’un modèle “optimal”

6.1 Principe

Le nouveau stagiaire M2 prend la suite et décide d’utiliser R afin de tester d’autres méthodes non linéaires et mettre en place une procédure de comparaison efficace des qualités prédictives des différents modèles. Le stagiaire a appris que l’erreur d’estimation ou d’ajustement d’un modèle est nécessairement une estimation biaisée car optimiste de l’erreur de prévision. Cette dernière doit donc être estimée sans biais par exemple par validation croisée (PRESS) ou sur un échantillon test. Le principe adopté est finalement celui largement répandu en *m*achine ou *s*tatistical learning ; il faut dissocier les étapes :

- estimation du modèle pour une méthode donnée,
- optimisation de la complexité du modèle : nombre de variables, pénalisation...,
- estimation de l’erreur de prévision.

L’estimation d’un modèle est réalisée sur la part “apprentissage” de l’échantillon, son optimisation par validation croisée sur ce même échantillon tandis que l’estimation de l’erreur de prévision est réalisée sur la part “test” de l’échantillon.

La plupart des autres scénarios proposés dans wikistat pour l’apprentissage de l’apprentissage statistique utilisent la librairie `caret` (Kunh, 2008 [**?**]) pour atteindre cet objectif. Cette librairie est un méta-package qui permet d’estimer toutes les methodes de modélisation par une seule fonction `train` et une syntaxe uniformisée. Malheureusement, cette fonction n’est pas “complètement” généralisée et elle pose des problèmes, pour certaines méthodes, avec l’introduction de variables explicatives qualitatives et plus encore avec la prise en compte des interactions dans les modèles linéaires. Elle est ici seulement utilisée pour diviser l’échantillons en deux partie : apprentissage et test car elle permet de le faire de façon équilibrée vis à vis de la variable `origine`.

6.2 Prise en compte des données

Lecture des données

```
cafe=read.table("cafe.dat") # lire les données
origine=as.factor(cafe[,1]) # définir un facteur
cafe=cafe[,-c(1,2)]         # oterer deux colonnes
# nommer les variables
dimnames(cafe)[[2]]=c("perte","lumin","xa","xb",
   "xy","xgn")
# construire la table finale
cafe=data.frame(cafe,origine)
```

Exploration

Il n’est pas utile de reproduire toute l’analyse réalisée avec SAS/INSIGHT mais évidemment, les mêmes fonctionnalités sont disponibles dans R.

```
hist(cafe$perte)
```

```
hist(cafe$lumin)
plot(cafe$perte,cafe$lumin,col=origine)
qqnorm(cafe$perte)
shapiro.test(cafe$perte)
# matrice des nuages de points
pairs(cafe[,1:6],col=origine)
# ACP
pca=princomp(cafe[,-7],cor=TRUE)
plot(pca,main="")
biplot(pca)
```

Division de l'échantillon

```
library(caret)
# indices de l'échantillon d'apprentissage
# en équilibrant la répartition des origines
set.seed(xxx) # remplacer xxx par un entier
inTrain = createDataPartition(cafe[,"origine"],
   p = .6, list = FALSE)
# Extraction des échantillons
# data frames apprentissage et test
cafeAppt=cafe[inTrain,]
cafeTest=cafe[-inTrain,]
```

Il est recommandé de centrer et réduire les variables explicatives quantitatives dans plusieurs méthodes. C'est fait systématiquement et simplement en utilisant évidemment les mêmes transformations (`xtrans`) sur l'échantillon test que celles calculées sur l'échantillon d'apprentissage.

```
# Normalisation
xTrans=preProcess(cafeAppt[,2:6])
cafeAppt[,2:6]=predict(xTrans,cafeAppt[,2:6])
cafeTest[,2:6]=predict(xTrans,cafeTest[,2:6])
```

6.3 Différentes modélisations

Le travail reprend les méthodes vues avec SAS avant d'en introduire d'autres. Celles proposées dans la littérature sont fort nombreuses ; voir à ce propos par exemple l'aide en ligne `?train` de la fonction de `caret` et le nombre d'options du paramètre `method`. Toutes ne seront pas testées d'autant que les différences observées en terme de qualité de prévision seront loin d'être toujours significatives. Seule disons une méthode par famille est estimée et testée.

Analyse de covariance

Le stagiaire utilise les même algorithmes de sélection de variables mais, comme le critère de sélection utilisé par R est différent : AIC plutôt que SBC, les modèles obtenus peuvent différer.

```
# Sélection descendante à partir du modèle
# de toutes interactions d'ordre 2
fit1.acova=glm(perte~(lumin + xa + xb + xy + xgn +
    origine)**2,data=cafeAppt)
fit1.step=step(fit1.acova,direction="back")
anova(fit1.step,test="Chisq")
```

Remarquer que, contrairement) la procédure `glmselect` de SAS, la procédure step de R respecte la consigne de conserver un effet simple s'il est par ailleurs présent dans une interaction afin de simplifier l'interprétation.

```
# Sélection stepwise
# à partir du modèle avec terme constant
fit2.acova=glm(perte~1,data=cafeAppt)
fit2.step=step(fit2.acova,direction="both",
   scope=list(lower=~1, upper=~(lumin + xa + xb
   + xy + xgn + origine)**2))
anova(fit2.step,test="Chisq")
```

La fonction `cv.glm` de la librairie `boot` estime l'erreur de prévision d'un modèle linéaire par *k-fold* validation croisée ; k étant le nombre de groupes. Par défaut k est égal au nombre d'observations ce qui correspond à la *leave-one-out cross validation* (loo CV) c'est-à-dire au PRESS. L'usage est souvent d'utiliser $k = 10$ jugée comme un bon compromis entre le biais de l'estimation lorsque k est petit et sa variance avec k grand. Comme la k-fold validation croisée est aléatoire, les résultats peuvent changer à chaque exécution à moins de contrôler l'initialisation du générateur de nombres aléatoirest (`set.seed`)

.

```
# librairie
library(boot)
set.seed(2)
# 10-fold CV du premier modèle
cv.glm(cafeAppt,fit1.step,K=10)$delta[1]
# 10-fold CV du deuxième modèle
set.seed(2)
cv.glm(cafeAppt,fit2.step,K=10)$delta[1]
```

Comme précédemment, il reste important de tracer le graphe des résidus. Une fonction permet de le faire en imposant l'échelle pour faciliter les comparaisons.

```
# fonction de plot des résidus
plot.res=function(x,y,titre=""){
plot(x,y,col="blue",xlim=c(10,25),ylim=c(-2,2),
ylab="Résidus",xlab="Valeurs predites",main=titre)
# points(x2,y,col="red")
abline(h=0,col="green")}
```

Retenir le meilleur modèle et l'utiliser pour prévoir l'échantillon test, calculer l'erreur de prévision et représenter les résidus.

```
# Calcul des prévisions
pred.glm=predict(fit2.step,newdata=cafeTest)
# residus de la prévision
res.glm=pred.glm-cafeTest[,"perte"]
# Erreur quadratique moyenne de prévision
mean(res.glm^2)
# graphe des résidus
plot.res(pred.glm,res.glm)
```

Modèle additif généralisé

Une librairie de R propose la même fonction que SAS pour l'estimation d'un modèle additif généralisé. Les paramètres de lissage de chaque fonction / transformation spline pourraient être optimisés mais cela semble bien compliqué pour une amélioration sans doute non significative. Comme dans SAS, la fonction `s` du modèle note une transformation non paramétrique des variables tandis que le facteur `origine` reste à l'identique.

```
library(gam)
fit.gam=gam(perte~s(lumin)+s(xa)+s(xb)+s(xy)+
    s(xgn)+origine,data=cafeAppt)
# prévision de l'échantillon test
pred.gam=predict(fit.gam,newdata=cafeTest)
# residus de la prévision
res.gam=pred.gam-cafeTest[,"perte"]
# Erreur quadratique moyenne de prévision
mean(res.gam^2)
# graphe des résidus
plot.res(pred.gam,res.gam)
```

Comparer avec les résultats de l'analyse de covariance en prenant en compte la complexité des procédures de choix de modèle mises en œuvre.

Arbre binaire

Cette méthode de contruction récursive d'un arbre binaire décision est fondamentalement non linéaire. Elle est testée à titre d'exemple d'utilisation. Le paramètre `cp` est un coefficient de pénalisation qui contrôle la complexité du modèle, à savoir le nombre de feuilles de l'arbre.

```
library(rpart)  # charger le package
# estimation d'un arbre détaillé
fit.tree=rpart(perte~.,data=cafeAppt,cp=0.001)
# graphe rudimentaire
plot(fit.tree)
text(fit.tree)
# choix  rudimentaire de la pénalisation
# ce choix est mal documenté
plotcp(fit.tree)
summary(fit.tree)
# choisir la valeur de cp correspondant au bon
# nsplit: première valeur ``sous le pointillé''
# Elagage
fit1.tree=prune(fit.tree,cp=0.005719204)
```

```
# Une stratégie plus fiable consiste à
# estimer l'erreur par validation croisée
xmat=xpred.rpart(fit.tree)
xerr=(xmat-cafeAppt[,"perte"])^2
apply(xerr,2,sum) #
fit2.tree=prune(fit.tree,cp=0.001998942)
# une fonction permet de construire un
# fichier postscript du graphe de l'arbre
# nettement mieux dessiné
post(fit2.tree,title="",width = 7.0,height = 7.0,
   horizontal = FALSE, onefile = FALSE,
   paper = "special")
# graphe observations vs. prévision
plot(predict(fit2.tree),cafeAppt[,"perte"])
# prévision de l'échantillon test
pred.tree=predict(fit2.tree,newdata=cafeTest)
# residus de la prévision
res.tree=pred.tree-cafeTest[,"perte"]
# Erreur quadratique moyenne de prévision
mean(res.tree^2)
# graphe des résidus
plot.res(pred.tree,res.tree)
```

Réseau de neurones

Les réseaux de neurones sont une autre méthode typiquement non linéaire mais pouvant présenter des difficultés d'estimation. L'algorithme est initialisé par une tirage aléatoire des poids ou paramètres du modèle. Comme la convergence de l'algorrithme n'est que locale, la solution dépend l'initialisation et donc de la valeur initialisant le générateur de nombres aléatoires. Les paramètres à estimer et contrôlant la complexité du modèle sont le nombre de neurones (`size`), une pénalisation (`decay`) comme en régression *ridge* et éventuellement le nombre d'itérations.

```
library(nnet)
fit.nn=nnet(perte~.,data=cafeAppt,size=2,
   linout=TRUE,decay=1,maxit=1000)
plot(predict(fit.nn),cafeAppt[,"perte"])
```

Une fonction de la librairie `e1071` permet de chercher des valeurs optimales des paramètres en minimisant l'erreur estimée par validation croisée.

```
library(e1071)
plot(tune.nnet(perte~.,data=cafeAppt,size=c(2,3,4),
   decay=c(0,.5,1),maxit=200,linout=TRUE))
set.seed(2)
fit.nn=nnet(perte~.,data=cafeAppt,size=3,
   linout=TRUE,decay=0,maxit=1000)
# prévision de l'échantillon test
pred.nn=predict(fit.nn,newdata=cafeTest)
# residus de la prévision
res.nn=pred.nn-cafeTest[,"perte"]
# Erreur quadratique moyenne de prévision
mean(res.nn^2)
# graphe des résidus
plot.res(pred.nn,res.nn)
```

Random forest

Toujours à titre illustratif, un algorithme d'agrégation de modèles, ici les forêts aléatoires est mis en œuvre. Le paramètre à optimiser est un nombre (`mtry`) de variables tirées aléatoirement à la construction de chaque nœud de chaque arbre. La même fonction `tune` est utilisée comme pour les réseaux de neurones.

```
library(randomForest)
fit.rf=randomForest(perte~.,data=cafeAppt)
plot(predict(fit.rf),cafeAppt[,"perte"])
plot(tune.randomForest(perte~.,data=cafeAppt,
   mtry=c(2,3,4,5)))
# estimation avec la valeur optimale
fit.rf=randomForest(perte~.,data=cafeAppt,mtry=3)
# prévision de l'échantillon test
pred.rf=predict(fit.rf,newdata=cafeTest)
# residus de la prévision
```

```
res.rf=pred.rf-cafeTest[,"perte"]
# Erreur quadratique moyenne de prévision
mean(res.rf^2)
# graphe des résidus
plot.res(pred.rf,res.rf)
```

Kernel regression least square

Une dernière méthode est testée : une régression non paramétrique pénalisée introduisant une transformation non linéaire dont la flexibilité est contrôlée, "régularisée" par un noyau. Elle appartient à la famille des machines à noyaux comme les SVM ou "supports vecteurs machines". Les valeurs des paramètres ne sont pas optimisées, ce sont celles par défaut.

```
library(KRLS)
fit.krls=krls(cafeAppt[,-c(1,7)],cafeAppt[,1])
plot(predict(fit.krls,cafeAppt[,-c(1,7)])$
   fit,cafeAppt[,"perte"])
# prévision de l'échantillon test
pred.krls=predict(fit.krls,newdata=
   cafeTest[,-c(1,7)])$fit
# residus de la prévision
res.krls=pred.krls-cafeTest[,"perte"]
# Erreur quadratique moyenne de prévision
mean(res.krls^2)
# graphe des résidus
plot.res(pred.krls,res.krls)
```

En conclusion, il s'agit donc de comparer entre elles les erreurs de prévision sur l'échantillon test afin de sélectionner le modèle (méthode et complexité) "optimal".

Attention à une remarque importante : la taille de l'échantillon test est faible, en conséquence l'estimation de l'erreur de prévision est peu fiable car soumise à une variance importante. Il s'agirait donc, comme cela est fait systématiquement dans les autres scénarios d'apprentissage de wikistat, d'itérer automatiquement la démarche précédente sur plusieurs, quelques dizaines, d'échantillons tests. Il est alors possible de prendre en compte la dispersion de ces erreurs et également d'en calculer la moyenne pour réduire la variance.

En conclusion il est important de rappeler qu'il n'y a pas de méthode uniformément meilleure, cela dépend des données, de leur structure. Pour cet exemple, l'exploration a mis évidence certaines *non linéarités* dans les relations entre les variables. Ce sont des caractéristiques importantes à intégrer. Un modèle linéaire n'est donc pas optimal il s'agit de choisir une meilleure ou moins mauvaise stratégie pour la prise en compte de ces non-linéarités :

1. transformer certaines variables par une fonction paramétrique non linéaire : $1/x, \log$ ... ou toute fonction puissance (transformée de Cox) tout en conservant un modèle linéaire,
2. introduire les interactions et donc un modèle polynomial, ici quadratique,
3. utiliser des méthodes estimant des combinaisons linéaires de transformations non-paramétriques des variables : `gam, krls`,
4. utiliser des méthodes typiquement non linéaires : arbres, réseaux de neurones...

Pour cet exemple, le bon choix semble être entre les stratégies 2 et 3. Bien sûr, l'analyse de covariance semble plus simple et interprétable mais la sélection de variables inhérente est elle nettement plus complexe à exécuter que l'estimation par `gam` ou `krls`.