Programmation à partir d'objets statistiques sous R

Ricco Rakotomalala

http://eric.univ-lyon2.fr/~ricco/cours/cours_programmation_R.html

R est un « vrai » langage de programmation.

Le logiciel R propose de très nombreuses techniques statistiques dont les résultats se présentent sous forme d'objets (avec un ensemble de propriétés/attributs) que l'on peut manipuler.

Nous pouvons aller (nettement) plus loin en programmant de nouvelles méthodes / procédures à partir des résultats fournis par les techniques existantes.

Un exemple de technique statistique : la régression avec lm()

LES PROPRIÉTÉS DE L'OBJET RÉGRESSION (LM)

L'objet « lm »

```
#chargement des données
data(mtcars)
#régression mpg vs. autres variables
reg <- lm(mpg ~ ., data = mtcars)
print(reg)
print(class(reg))
#attributs de la régression
print(attributes(reg))
#accéder aux coefficients
print(reg$coefficients)
#quels coefficients sont > 0 ?
print(which(reg$coefficients > 0))
```

attributes() permet de connaître les propriétés d'un objet R, qui se présente comme une liste en définitive !

Ex. reg[[1]] ⇔ reg$coefficients

```
Console ~/
> #chargement des données
> data(mtcars)
>
> #régression mpg vs. autres variables
> reg <- lm(mpg ~ ., data = mtcars)
> print(reg)

Call:
lm(formula = mpg ~ ., data = mtcars)

Coefficients:
(Intercept)          cyl         disp           hp         drat
   12.30337     -0.11144      0.01334     -0.02148      0.78711
         wt         qsec           vs           am         gear
   -3.71530      0.82104      0.31776      2.52023      0.65541
       carb
   -0.19942

> print(class(reg))
[1] "lm"
>
> #attributs de la régression
> print(attributes(reg))
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"

>
> #accéder aux coefficients
> print(reg$coefficients)
(Intercept)         cyl        disp          hp        drat          wt
12.30337416 -0.11144048  0.01333524 -0.02148212  0.78711097 -3.71530393
       qsec          vs          am        gear        carb
 0.82104075  0.31776281  2.52022689  0.65541302 -0.19941925
>
> #quels coefficients sont > 0 ?
> print(which(reg$coefficients > 0))
(Intercept)        disp        drat        qsec          vs          am
          1           3           5           7           8           9
       gear
         10
>
```

L'objet summary.lm

```
#summary d'une régression
sum.reg <- summary(reg)
print(sum.reg)
#ses attributs
print(attributes(sum.reg))
```

```
> print(sum.reg)

Call:
lm(formula = mpg ~ ., data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4506 -1.6044 -0.1196  1.2193  4.6271

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.30337   18.71788   0.657   0.5181
cyl         -0.11144    1.04502  -0.107   0.9161
disp         0.01334    0.01786   0.747   0.4635
hp          -0.02148    0.02177  -0.987   0.3350
drat         0.78711    1.63537   0.481   0.6353
wt          -3.71530    1.89441  -1.961   0.0633 .
qsec         0.82104    0.73084   1.123   0.2739
vs           0.31776    2.10451   0.151   0.8814
am           2.52023    2.05665   1.225   0.2340
gear         0.65541    1.49326   0.439   0.6652
carb        -0.19942    0.82875  -0.241   0.8122
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.65 on 21 degrees of freedom
Multiple R-squared:  0.869, Adjusted R-squared:  0.8066
F-statistic: 13.93 on 10 and 21 DF,  p-value: 3.793e-07

>
> #ses attributs
> print(attributes(sum.reg))
$names
 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"

$class
[1] "summary.lm"
```

L'objet summary.lm (suite)

```
#coefficient devient une matrice
print(sum.reg$coefficient)
#ex. colonne des p-value
print(sum.reg$coefficient[,4])
#accéder au R2
sum.reg$r.squared
```

```
> #coefficient est une matrice maintenant
> print(sum.reg$coefficient)
               Estimate  Std. Error    t value   Pr(>|t|)
(Intercept) 12.30337416 18.71788443  0.6573058 0.51812440
cyl         -0.11144048  1.04502336 -0.1066392 0.91608738
disp         0.01333524  0.01785750  0.7467585 0.46348865
hp          -0.02148212  0.02176858 -0.9868407 0.33495531
drat         0.78711097  1.63537307  0.4813036 0.63527790
wt          -3.71530393  1.89441430 -1.9611887 0.06325215
qsec         0.82104075  0.73084480  1.1234133 0.27394127
vs           0.31776281  2.10450861  0.1509915 0.88142347
am           2.52022689  2.05665055  1.2254035 0.23398971
gear         0.65541302  1.49325996  0.4389142 0.66520643
carb        -0.19941925  0.82875250 -0.2406258 0.81217871
>
> #ex. colonne des p-value
> print(sum.reg$coefficient[,4])
(Intercept)         cyl        disp          hp        drat          wt
 0.51812440  0.91608738  0.46348865  0.33495531  0.63527790  0.06325215
       qsec          vs          am        gear        carb
 0.27394127  0.88142347  0.23398971  0.66520643  0.81217871
>
> #accéder le R2
> sum.reg$r.squared
[1] 0.8690158
```

La 4ème colonne d'une matrice est un vecteur…

Accès au R^2

Recherche de la meilleure régression simple à partir d'un ensemble de variables explicatives candidates

EXEMPLE : SÉLECTION DE MODÈLES

- y : un vecteur représentant la variable cible.
- X : data frame contenant un ensemble de variables prédictives candidates.
- On cherche parmi ces variables celle qui prédit le mieux y dans une régression simple.

```
best.regression.simple <- function(y,X){
 #préparation vecteur recueil des R2
 r2.all <- numeric(ncol(X))
 #pour chaque variable candidate
 #on pourrait passer par un sapply() !
 for (j in 1:ncol(X)){
  z <- X[,j]
  reg.1 <- lm(y ~ z)
  sum.reg.1 <- summary(reg.1)
  r2.all[j] <- sum.reg.1$r.squared
 }
 #nommer les R2 avec le nom des variables
 names(r2.all) <- colnames(X)
 #affichage
 print(r2.all)
 #détection de l'indice de la meilleure variable
 best.index <- which.max(r2.all)
 #récupération du nom de la variable
 best.name <- names(X)[best.index]
 #formule pour la meilleure régression simple
 formule <- paste("y",best.name,sep=" ~ ")
 #régression avec la meilleure variable détectée
 best.reg <- lm(formule,data=cbind(y,X))
 return(summary(best.reg))
}
```

R^2 des régressions simples. Prise individuellement, « wt » se révèle être la meilleure variable explicative.

Résultat détaillé de la régression de « mpg » (y) sur « wt ».

```
> #ex. prédiction de mpg
> #pour les données mtcars
> y <- mtcars[,1]
> X <- mtcars[,-1]
> #appel de la fonction
> print(best.regression.simple(y,X))
      cyl      disp        hp      drat        wt      qsec        vs
0.7261800 0.7183433 0.6024373 0.4639952 0.7528328 0.1752963 0.4409477
       am      gear      carb
0.3597989 0.2306734 0.3035184

Call:
lm(formula = formule, data = cbind(y, X))

Residuals:
    Min      1Q  Median      3Q     Max
-4.5432 -2.3647 -0.1252  1.4096  6.8727

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
wt           -5.3445     0.5591  -9.559 1.29e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared:  0.7528,	Adjusted R-squared:  0.7446
F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

Conclusion - R est magique

De la documentation à profusion (**n'achetez jamais des livres sur R**)

Site du cours

http://eric.univ-lyon2.fr/~ricco/cours/cours_programmation_R.html

Programmation R

http://www.duclert.org/

Quick-R

http://www.statmethods.net/

POLLS (Kdnuggets)

Data Mining / Analytics Tools Used

(R, 2nd ou 1er depuis 2010)

What languages you used for data mining / data analysis?

http://www.kdnuggets.com/polls/2013/languages-analytics-data-mining-data-science.html

(Août 2013, langage R en 1ère position)

Article New York Times (Janvier 2009)

"**Data Analysts Captivated by R's Power**" - http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=1