Les variables dans R

Nadia Aubin-Horth

Frédéric Maps

Philippe Massicotte

Matériel accompagnateur

Classes de base

Il existe quatre principales classes de variables dans R que vous devriez connaître sur le bout de vos doigts.

La classe `numeric`

La classe `numeric` sert à représenter les nombres (ex.: `1`, `3.14`, `-3324.56`).

```
12
```

```
## [1] 12
```

```
-1.45
```

```
## [1] -1.45
```

```
3 + pi
```

```
## [1] 6.141593
```

La classe `character`

La classe `character` sert à représenter les caractères ou les chaînes de caractères. Les chaînes de caractères sont soit entourées de simples `'` ou doubles `"` guillemets.

```
"banane"
```

```
## [1] "banane"
```

```
'banane'
```

```
## [1] "banane"
```

```
# Est-ce que les deux chaînes sont identiques?
"banane" == "Banane"
```

```
## [1] FALSE
```

Info! R est un langage de programmation dit *case sensitive*. C'est-à-dire que la casse (minuscule/majuscule) des caractères est importante. Ainsi, la chaîne de caractères `"banane"` est différente de la chaîne de caractères `"Banane"`.

La classe `logic`

La classe `logic` sert à représenter les valeurs booléennes (VRAI/FAUX). Dans R, les valeurs `logic` sont représentées soit à l'aide des valeurs `TRUE`, `FALSE` ou `T`, `F`. Il est cependant recommandé d'utiliser la forme longue (`TRUE`, `FALSE`).

```
TRUE
```

```
## [1] TRUE
```

```
FALSE
```

```
## [1] FALSE
```

La classe `factor`

La classe `factor` sert à représenter les valeurs lorsqu'on connaît d'avance les différentes possibilités (ex. mois de l'année). Les facteurs sont généralement créés avec la fonction `factor()`.

```
factor(c("lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche")
)
```

```
## [1] lundi    mardi    mercredi jeudi    vendredi samedi   dimanche
## Levels: dimanche jeudi lundi mardi mercredi samedi vendredi
```

On peut utiliser la fonction `class()` pour connaître la classe d'une variable.

```
class(1)
```

```
## [1] "numeric"
```

```
class("banane")
```

```
## [1] "character"
```

```
class("Banane")
```

```
## [1] "character"
```

```
class(factor(123.1))
```

```
## [1] "factor"
```

Il est également possible de convertir les classes. Par exemple, on peut convertir une classe `character` en classe `numeric`.

```
class("3.4")
```

```
## [1] "character"
```

```
class(as.numeric("3.4"))
```

```
## [1] "numeric"
```

Noms de variables

Il est souvent fort utile de pouvoir réutiliser les valeurs (numérique, texte ou autre) d'une opération dans d'autres calculs par exemple. Pour ce faire, il faut assigner la valeur à une variable. Dans R, l'assignement d'une valeur à une variable peut se faire à l'aide des opérateurs `<-` et `=`. Cependant, il est préférable d'utiliser l'opérateur `<-` plutôt que `=` pour l'assignation d'une variable. En plus, l'utilisation de `<-` permet d'éviter la confusion avec l'opérateur logique `==`.

```r
x <- 2 # Assigner la valeur numérique 2 à la variable x
y <- 40 # Assigner la valeur numérique 40 à la variable y
x + y # Somme de x et y
```

```
## [1] 42
```

Un nom de variable peut contenir à la fois des lettres et des chiffres, mais doit commencer par une lettre.

```r
## # Bon
## x123 <- TRUE
##
## # Pas bon
## 123x <- "R est vraiment trop cool!"
```

Il est important de choisir un nom de variable qui est descriptif. Dans les exemples suivants, on créer une variable pour contenir une valeur de chlorophylle a.

```r
## x <- 200 # x n'est pas assez descriptif
## chla <- 200 # mieux, mais on ne connaît pas les unités
## chla_mg_m2 <- 200 # idéal, on connaît la variable et son unité
```

Il est important d'être consistent dans la casse des noms de variables. Par exemple, `chla_mg_m2` et `Chla_mg_m2` sont deux variables distinctes.

```r
## chla_mg_m2 <- 200
## Chla_mg_m2 <- 200
```

Info! Il est important que vos noms de variables n'entrent pas en conflit avec des noms de variables ou de fonctions déjà définies dans R. Par exemple, il faut éviter d'utiliser `cos`, `sin`, `min`, `median`, `pi`.

```r
## cos <- 3 # À ne pas faire!
```

Les vecteurs

Les vecteurs sont des tableaux d'éléments successifs. Les éléments d'un vecteur peuvent être, par exemple: `numeric`, `character`, `logic`. Dans R, les vecteurs sont créés à l'aide de la fonction `c()` (qui veut dire *combine*) et les valeurs sont séparées à l'aide de la virgule (`,`).

```r
# Création d'un vecteur avec 6 valeurs numériques
v1 <- c(100, 101, 102, 103, 104, 105
)
v1
```

```
## [1] 100 101 102 103 104 105
```

Pour accéder aux valeurs d'un vecteur, il suffit de spécifier l'index (position) à laquel nous voulons accéder.

```r
v1[3] # 3ième élément du vecteur v
```

```
## [1] 102
```

```r
v1[3:5] # Positions 3,4,5 du vecteur v
```

```
## [1] 102 103 104
```

Info! Dans R il est possible de créer rapidement une séquence de nombre en utilisant l'opérateur `:`. Par exemple, la commande `3:5` retourne un vecteur numérique avec les valeurs 3, 4 et 5.

Il est également possible de supprimer un élément d'un vecteur en utilisant l'opérateur `-` suivi de l'index à supprimer.

```r
v1[-3] # Supprimer le 3ième élément du vecteur v
```

```
## [1] 100 101 103 104 105
```

Des opérations arithmétiques sont également possibles avec les vecteurs.

```r
v1 <- c(100, 101, 102, 103, 104, 105
)
v2 <- c(10, 20, 30, 40, 50, 60
)
v1 + v2
```

```
## [1] 110 121 132 143 154 165
```

Autres types de classes

Il existe un très grand nombre d'objets dans R. Ainsi, d'autres classes d'objets vous seront utiles:

- `data.frame` : Format habituel de tableau de données que vous importez.
- `list` : Liste d'objets de classes différentes (texte, numérique, etc.).
- `lm` : Résultat d'une analyse de modèle linéaire.
- `date` : Pour représenter les dates.

Dans l'exemple suivant, un modèle linéaire `model1` est créé avec la fonction `lm()`. Différentes fonctions génériques peuvent être utilisées avec la variable `model1`.

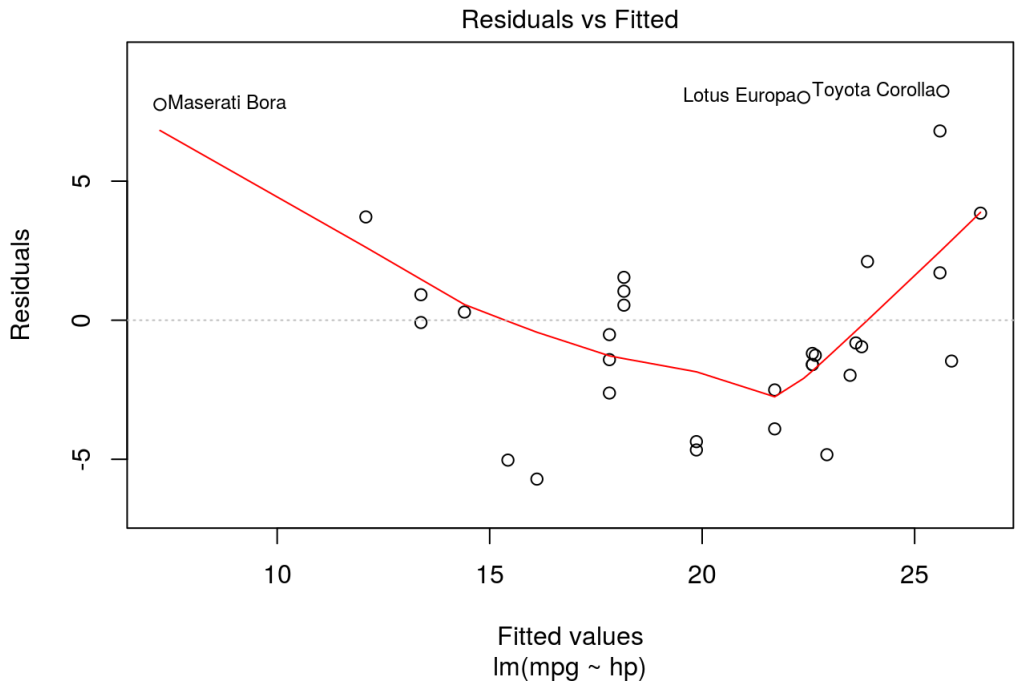
```
model1 <- lm(mpg ~ hp, data = mtcars) # Créer un modèle linéaire (lm)

class(model1) # Voir la classe de model1
```

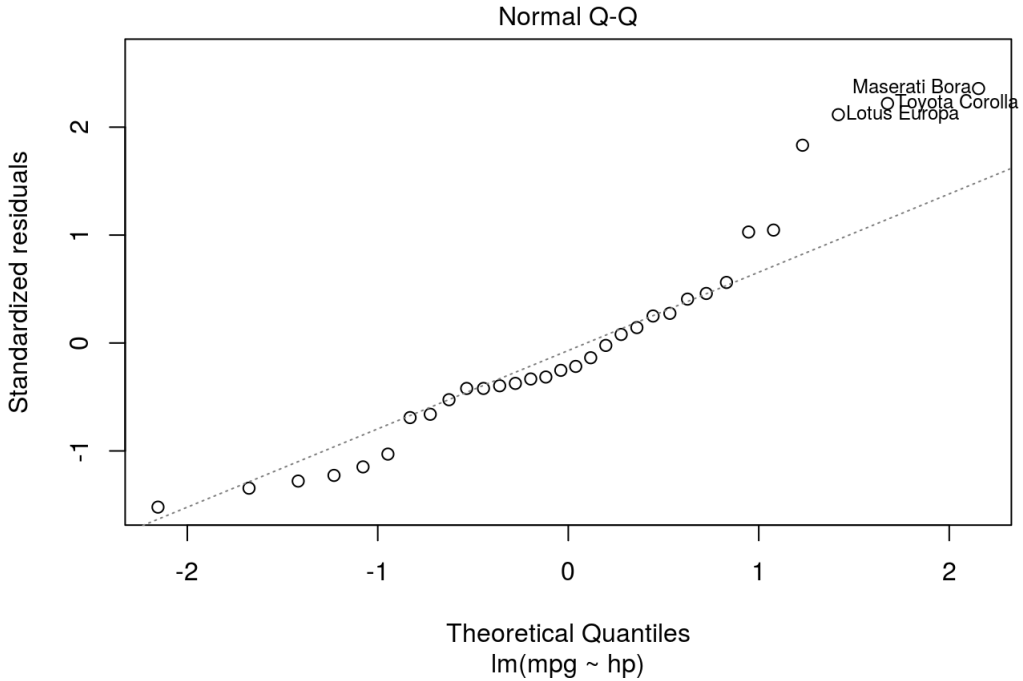
```
## [1] "lm"
```

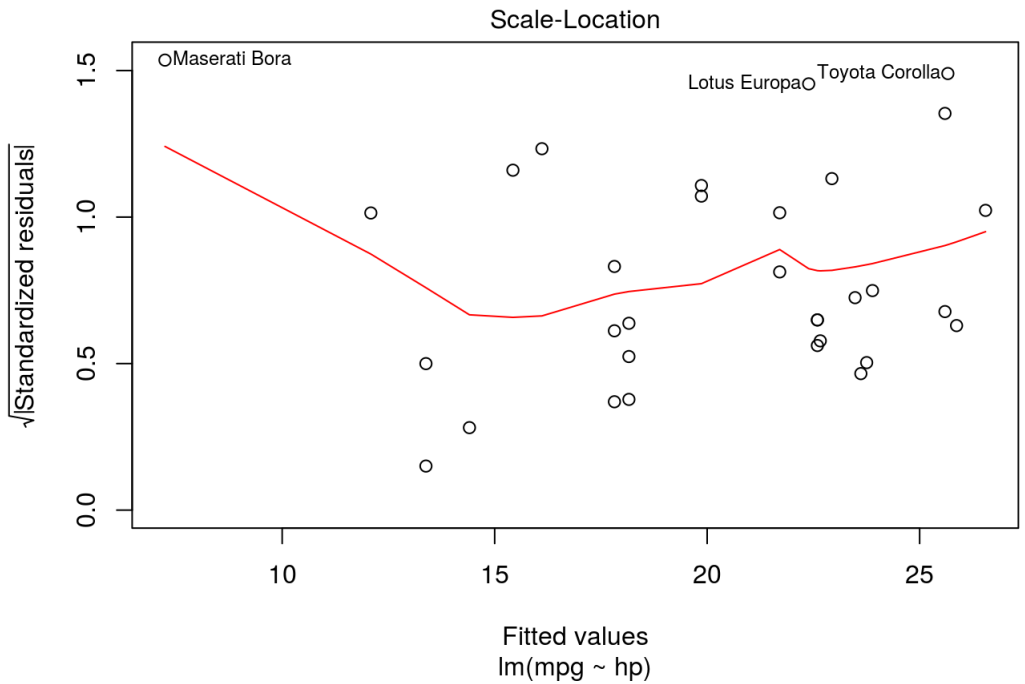
```
summary(model1) # Diagnostiques du modèle
```

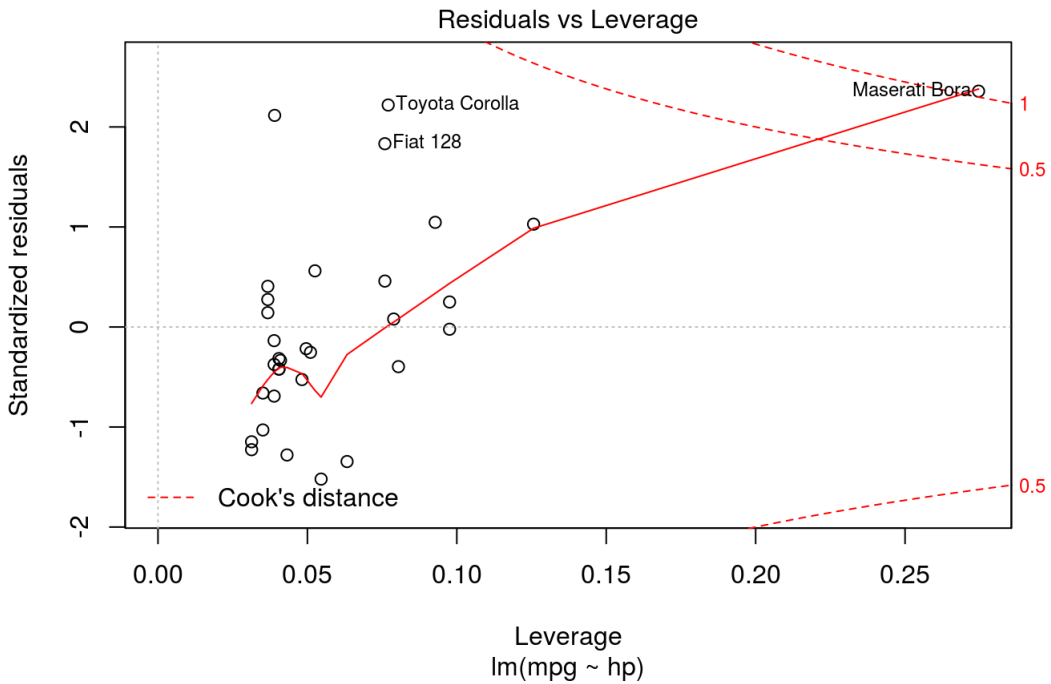
```
##
## Call:
## lm(formula = mpg ~ hp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7121 -2.1122 -0.8854  1.5819  8.2360
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.09886    1.63392  18.421  < 2e-16 ***
## hp          -0.06823    0.01012  -6.742 1.79e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.863 on 30 degrees of freedom
## Multiple R-squared:  0.6024, Adjusted R-squared:  0.5892
## F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07
```

```
plot(model1) # Graphiques de diagnostic du modèle
```


Residuals vs Fitted
Maserati Bora
Lotus Europa
Toyota Corolla
Residuals
5
0
-5
10
15
20
25
Fitted values
lm(mpg ~ hp)


Normal Q-Q
Maserati Bora
Toyota Corolla
Lotus Europa
Standardized residuals
-1
0
1
2
-2
-1
0
1
2
Theoretical Quantiles
lm(mpg ~ hp)


Scale-Location
Maserati Bora
Lotus Europa
Toyota Corolla
√|Standardized residuals|
0.0
0.5
1.0
1.5
10
15
20
25
Fitted values
lm(mpg ~ hp)



Téléchargement

Le matériel pédagogique utilisé dans cette capsule est disponible pour le téléchargement sous deux formats différents:

1. Format PDF standard que vous pouvez consulter et commenter avec Adobe Reader par exemple.
2. Format HTML dynamique qui se comporte comme une page web et doit être lu avec votre navigateur préféré (Chrome, Firefox, Edge, Safari, etc.).

Pour télécharger le fichier localement sur votre ordinateur, tablette ou téléphone portable, il suffit de cliquer sur le lien désiré avec le bouton droit de votre souris et choisir “sauvegarder sous...”.

- Télécharger la documentation sous format PDF
- Télécharger la documentation sous format HTML