Visualiser le code d'une fonction

Annie Bouvier et Fabrice Dessaint
Inra, UR1404 MaIAGE, Jouy-en-Josas et
Inra, UMR1347 Agroécologie, Dijon

Février 2016

Cette note présente l'utilisation des fonctions `methods()`, `getAnywhere()` ainsi que les fonctions spécifiques aux objets de type S3, `.S3methods()`, `getS3method()` et S4, `showMethods()`, `.S4methods()` et `getMethod()`.

Ces fonctions permettent d'afficher la liste et le code de fonctions *R*.

Visualiser le code source d'une fonction

En général, sous *R*, il suffit de saisir le nom d'une fonction, sans les parenthèses, pour afficher son code source. Par exemple, pour la fonction `rank()`, on obtient le code de la fonction avec l'instruction :

```
print(rank)

function (x, na.last = TRUE, ties.method = c("average", "first",
    "random", "max", "min"))
{
    nas <- is.na(x)
    nm <- names(x)
    ties.method <- match.arg(ties.method)
    if (is.factor(x))
        x <- as.integer(x)
    x <- x[!nas]
    y <- switch(ties.method, average = , min = , max = .Internal(rank(x,
        length(x), ties.method)), first = sort.list(sort.list(x)),
        random = sort.list(order(x, stats::runif(sum(!nas)))))
    if (!is.na(na.last) && any(nas)) {
        yy <- NA
        NAkeep <- (na.last == "keep")
        if (NAkeep || na.last) {
            yy[!nas] <- y
            if (!NAkeep)
                yy[nas] <- (length(y) + 1L):length(yy)
        }
        else {
            len <- sum(nas)
            yy[!nas] <- y + len
            yy[nas] <- seq_len(len)
        }
        y <- yy
        names(y) <- nm
    }
    else names(y) <- nm[!nas]
    y
}
<bytecode: 0x7fb28ac2d5c8>
<environment: namespace:base>
```

ou plus simplement

```
rank
```

Note : La ligne `<bytecode: 0x7ff8b4881df0>` indique que la fonction est byte-compiled (voir la package **compiler**)

Cette procédure est très utile lorsque l'on veut créer une version personnalisée d'une fonction de *R* puisqu'il suffit de copier le code dans la nouvelle fonction pour ensuite le modifier.

```
Rangs <- rank
```

POUR certaines fonctions dites « génériques »[1], comme les fonctions `summary()`, `print()`, `plot()`, `anova()`, etc., le code affiché est très restreint.

Par exemple, pour la fonction `summary()`, on a juste l'indication qu'il existe des méthodes.

1. Fonctions possédant plusieurs versions, chacune d'elle étant spécifique d'une « méthode »

```
print(summary)

function (object, ...)
UseMethod("summary")
<bytecode: 0x7fb28aef0710>
<environment: namespace:base>
```

La fonction `methods()`

Cette fonction affiche les différentes *méthodes* — versions de la fonction générique pour différentes classes d'objet — disponibles.

Elle possède 2 arguments : l'argument `generic.function=` permet de spécifier le nom de la fonction dont on cherche à connaître les différentes méthodes ; l'argument `class=` permet de rechercher les fonctions spécifiques à une classe d'objet.

Syntaxe de la fonction `methods()`.

```
methods(generic.function, class)
```

Par exemple, si l'on veut connaître les fonctions disponibles[2] pour la classe d'objet `lm`, on saisit l'instruction :

2. Le nombre de fonctions affichées peut varier selon les **packages** chargés

```
methods(class="lm")

 [1] add1           alias          anova          case.names     coerce
 [6] confint        cooks.distance deviance       dfbeta         dfbetas
[11] drop1          dummy.coef     effects        extractAIC     family
[16] formula        hatvalues      influence      initialize     kappa
[21] labels         logLik         model.frame    model.matrix   nobs
[26] plot           predict        print          proj           qr
[31] residuals      rstandard      rstudent       show           simulate
[36] slotsFromS3    summary        variable.names vcov
see '?methods' for accessing help and source code
```

Pour connaître, les différentes méthodes[3] de la fonction `summary()` :

3. Le nombre de méthodes affichées peut varier selon les **packages** chargés

```
methods(generic.function="summary")

 [1] summary.aov                    summary.aovlist*
 [3] summary.aspell*                summary.check_packages_in_dir*
```

```
 [5] summary.connection            summary.data.frame
 [7] summary.Date                  summary.default
 [9] summary.ecdf*                 summary.factor
[11] summary.glm                   summary.infl*
[13] summary.lm                    summary.loess*
[15] summary.manova                summary.matrix
[17] summary.mlm*                  summary.nls*
[19] summary.packageStatus*        summary.PDF_Dictionary*
[21] summary.PDF_Stream*           summary.POSIXct
[23] summary.POSIXlt               summary.ppr*
[25] summary.prcomp*               summary.princomp*
[27] summary.proc_time             summary.srcfile
[29] summary.srcref                summary.stepfun
[31] summary.stl*                  summary.table
[33] summary.tukeysmooth*
see '?methods' for accessing help and source code

## ## ou plus simplement
## methods("summary")
```

Dans la liste retournée[4], tous les noms de fonctions commencent par le nom de la fonction générique (ici `summary`) complétés par un mot qui correspond en général à une classe d'objet *R* (`factor`, `prcomp`).

Dans cette liste, on peut différencier 2 types de fonctions : des fonctions *étoilées* dont le nom se termine par un astérisque (`summary.prcomp*`) et d'autres sans astérisque (`summary.factor`).

4. Dans cet exemple, il n'y a que des fonctions de type S3

Les fonctions non marquées d'un astéristique sont directement affichables :

```
summary.factor

function (object, maxsum = 100, ...)
{
    nas <- is.na(object)
    ll <- levels(object)
    if (any(nas))
        maxsum <- maxsum - 1
    tbl <- table(object)
    tt <- c(tbl)
    names(tt) <- dimnames(tbl)[[1L]]
    if (length(ll) > maxsum) {
        drop <- maxsum:length(ll)
        o <- sort.list(tt, decreasing = TRUE)
        tt <- c(tt[o[-drop]], `(Other)` = sum(tt[o[drop]]))
    }
    if (any(nas))
        c(tt, `NA's` = sum(nas))
    else tt
}
<bytecode: 0x7fb28c4dafb8>
<environment: namespace:base>
```

alors que pour les fonctions étoilées, la demande d'affichage se traduit par un message d'erreur :

```
summary.aovlist
```

```
Error in eval(expr, envir, enclos): objet 'summary.aovlist' introuvable
```

```
summary.princomp
```

```
Error in eval(expr, envir, enclos): objet 'summary.princomp' introuvable
```

Pour afficher le contenu de ces fonctions, on doit utiliser une autre fonction, la fonction `getAnywhere()`, que l'on détaille dans le paragraphe suivant.

La fonction `methods()` retourne d'autres informations qui ne sont pas affichées. En particulier, elle indique si la fonction dérivée de la fonction générique est de type S3 ou S4 (`isS4=TRUE`, dans la sortie ci-dessous).

```
library("Matrix")
tmp <- methods("dim")
print(attr(tmp, "info")[, c(1,4)])
```

```
                               visible  isS4
dim,Matrix-method                 TRUE  TRUE
dim,MatrixFactorization-method    TRUE  TRUE
dim.data.frame                    TRUE FALSE
dim.layout                       FALSE FALSE
dim.trellis                      FALSE FALSE
```

Le type S3 ou S4 est celui du mécanisme de programmation orientée objet utilisé dans la méthode. Les méthodes de type S3 apparaissent sous la forme `nom-fonction.nom-classe` (`dim.data.frame`) tandis que celles de type S4 apparaissent sous la forme `nom-fonction,nom-classe-method` (`dim,Matrix-method`).

La fonction `getAnywhere()`

Syntaxe de la fonction `getAnywhere()`.

```
getAnywhere(x)
```

Cette fonction affiche le code d'une fonction étoilée passée en argument.

```
getAnywhere(x="summary.aovlist")
```

```
A single object matching 'summary.aovlist' was found
It was found in the following places
  registered S3 method for summary from namespace stats
  namespace:stats
with value

function (object, ...)
{
    if (!is.null(attr(object, "weights")))
        cat("Note: The results below are on the weighted scale\n")
    dots <- list(...)
    strata <- names(object)
    if (strata[1L] == "(Intercept)") {
        strata <- strata[-1L]
        object <- object[-1L]
    }
    x <- setNames(vector(length = length(strata), mode = "list"),
        paste("Error:", strata))
```

```
    for (i in seq_along(strata)) x[[i]] <- do.call("summary",
        c(list(object = object[[i]]), dots))
    class(x) <- "summary.aovlist"
    x
}
<bytecode: 0x7fb290029a38>
<environment: namespace:stats>
```

Pour afficher le code d'une méthode de type S4 (`isS4=TRUE`), il faut utiliser la fonction `getMethod()`.

Les autres fonctions

Les fonctions `.S3methods()` et `getS3method()`

La fonction `.S3methods()` liste les méthodes de type S3 disponibles pour une fonction générique ou pour une classe d'objet donnée. Sa synthaxe est la suivante :

```
.S3methods(generic.function, class, envir = parent.frame())
```

Par exemple, pour connaître les méthodes de type S3 spécifiques à la fonction générique `dim()`, on saisit l'instruction :

```
.S3methods("dim")

[1] dim.data.frame dim.layout*    dim.trellis*
see '?methods' for accessing help and source code
```

Pour connaître les méthodes de type S3 disponibles pour la classe d'objet `Matrix`, l'instruction est :

```
.S3methods(class="Matrix")

[1] as.array  as.matrix as.vector
see '?methods' for accessing help and source code
```

Remarque : la fonction `.S3methods()` est quasi-similaire à la fonction `methods()`, mais elle ne fait apparaître que les méthodes de type S3.

La fonction `getS3method()` affiche le code source d'une méthode S3.

```
getS3method(f, class, optional = FALSE)
```

Par exemple, l'instruction suivante affiche le code source de la méthode de type S3, `dim()`, définie pour la classe d'objet `layout`.

```
 getS3method("dim", class="layout")

function (x)
{
    c(x$nrow, x$ncol)
}
<bytecode: 0x7fb28e4002e8>
<environment: namespace:grid>
```

Remarque : la sortie est quasi-similaire à celle de l'instruction :

```
getAnywhere("dim.layout")
```

Les fonctions `showMethods()`, `.S4methods()`, `getMethod()`

La fonction `showMethods()` liste les méthodes de type S4 disponibles pour une fonction générique et/ou d'une classe d'objet.

```
showMethods(f = character(), where = topenv(parent.frame()),
    classes = NULL, includeDefs = FALSE, inherited = !includeDefs,
    showEmpty, printTo = stdout(), fdef = getGeneric(f, where = where))
```

Par exemple, si on veut connaître les méthodes de type S4 de la fonction générique `dim()`, on saisit l'instruction :

```
showMethods("dim")

Function: dim (package base)
x="Matrix"
x="MatrixFactorization"
```

Le nom du package où la fonction générique est définie apparaît d'abord. Il est suivi par la liste des *signatures* — c'est-à-dire des classes d'objet — pour lesquelles une méthode de style S4 est disponible.

Remarque : Pour limiter la liste aux méthodes disponibles pour une classe d'objet, on rajoute l'argument `class=`.

Par exemple, pour connaître la liste des méthodes S4 de la fonction générique `dim()` qui sont disponibles pour la classe d'objet `MatrixFactorization`, on saisit l'instruction :

```
showMethods("dim", class="MatrixFactorization")

Function: dim (package base)
x="MatrixFactorization"
```

Pour connaître la liste de toutes les méthodes S4 disponibles pour une classe d'objet, on n'indique que l'argument `class=`.

```
showMethods(class="MatrixFactorization")

Function: dim (package base)
x="MatrixFactorization"

Function: expand (package Matrix)
x="MatrixFactorization"

Function: show (package methods)
object="MatrixFactorization"

Function: solve (package base)
a="MatrixFactorization", b="ANY"
a="MatrixFactorization", b="missing"
a="MatrixFactorization", b="numeric"
```

La fonction `.S4methods()` est une forme simplifiée de la fonction `showMethods()`.

```
.S4methods(generic.function, class)
```

Par exemple, l'instruction suivante liste les méthodes de type S4 de la fonction générique `dim()`

```
.S4methods("dim")

[1] dim,Matrix-method
[2] dim,MatrixFactorization-method
see '?methods' for accessing help and source code
```

La fonction `getMethod()` permet d'afficher le code source d'une méthode de type S4.

```
getMethod(f, signature = character(), where = topenv(parent.frame()),
    optional = FALSE, mlist, fdef)
```

Par exemple, l'instruction suivante affiche le code source de la méthode S4 `dim()` définie pour la classe d'objet `MatrixFactorization`.

```
getMethod("dim","MatrixFactorization")

Method Definition:

function (x)
x@Dim
<bytecode: 0x7fb28f3b4ff0>
<environment: namespace:Matrix>

Signatures:
        x
target  "MatrixFactorization"
defined "MatrixFactorization"
```

Compléments

Visualiser le code d'une fonction sous Rstudio

Sous Rstudio, on peut visualiser le code d'une fonction en positionnant le curseur sur la fonction et en appuyant sur le touche F2

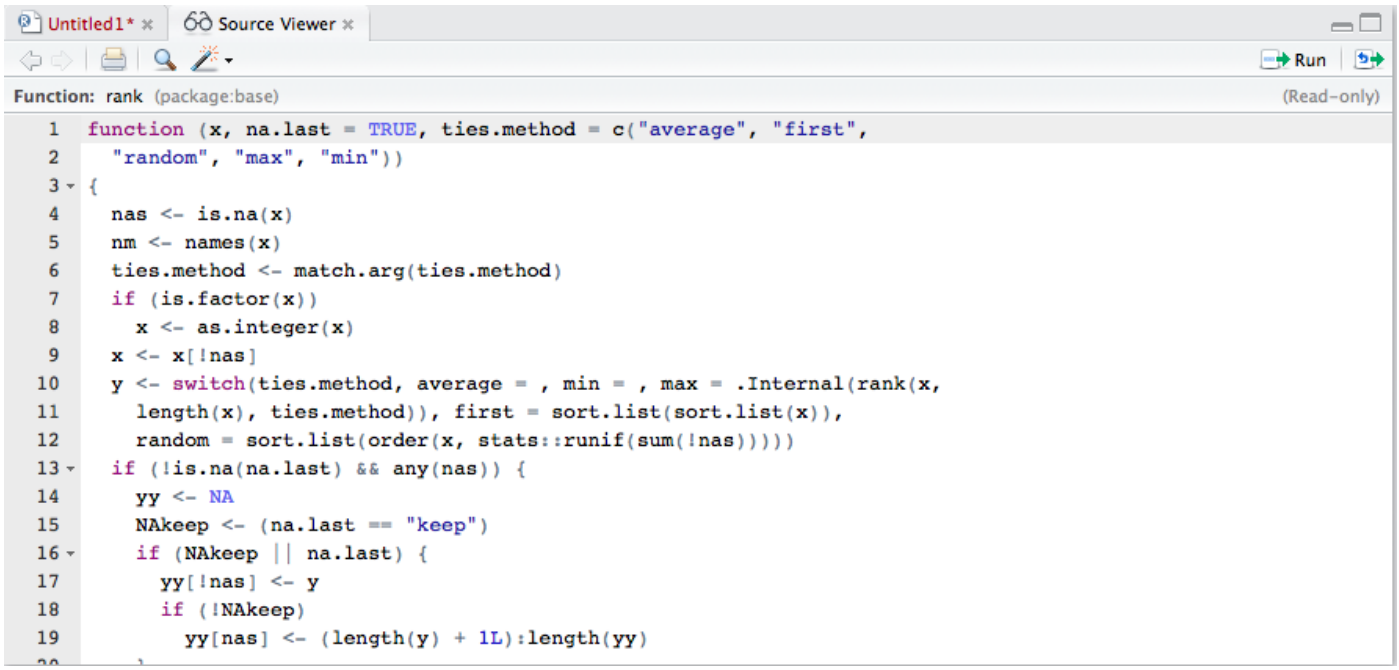

FIGURE 1: Source de la fonction `rank()` dans Rstudio