Construire un package R

Un package, paquet ou bibliothèque de programmes externes, est simplement un ensemble de programmes R qui complète et permet d'augmenter les fonctionnalités de R. Un package est généralement dévolu à des méthodes particulières ou à un domaine d'applications. Il est possible de construire un package, pour usage personnel ou pour être diffusé (via le CRAN par exemple).

1 Générer les fichiers du package

Le plus simple pour générer les fichiers du package est d'ouvrir une session R vide. Il faut ensuite construire les fonctions et générer les jeux de données que l'on souhaite mettre dans le package. Une fois que toutes les fonctions et tous les jeux de données sont présents dans la session, il suffit de faire:

```
> package.skeleton(name="MonPackage")
```

On peut éventuellement préciser la liste des objets que l'on veut mettre dans le package grâce à l'argument `list`.

Un dossier `MonPackage` est créé là où la session R est ouverte. Ce dossier contient tout les fichiers nécessaires pour construire un package:

- un fichier `DESCRIPTION`,
- un répertoire `data` qui contient tous les fichiers qui correspondent à un jeu de données,
- un répertoire `R` qui contient tous les fichiers correspondant aux fonctions disponibles,
- un répertoire `man` qui contient des fichiers qui décrivent les fonctions ou les jeux de données.

1.1 Le fichier `DESCRIPTION`

Dans ce fichier, il faut compléter les arguments :

- `Package`: MonPackage
- `Type`: Package

- `Title`: Ce que fait le package (titre court)
- `Version`: 1.0 pour commencer...
- `Date`: Date de compilation (automatique)
- `Author`: Auteur(s)
- `Maintainer`: Adresse de l'auteur à qui écrire en cas de question(s)
- `Description`: Ce que fait le package, en plus détaillé que le titre (quelques lignes)
- `License`: La licence
- `LazyLoad`: yes

1.2 Le répertoire `man`

Tous les fichiers du répertoire `man` ont la bonne structure mais les différentes rubriques doivent être complétés:

- `name`: précise le nom de la fonction (même nom que celui dans le fichier définissant la fonction et dans le répertoire `R`)
- `alias`
- `title`: titre de la fonction permettant de décrire très succinctement la fonction
- `description`: décrit la fonction en une à cinq lignes
- `usage`: précise comment est utilisée la fonction (donne les arguments de la fonction, dans le même ordre que pour la fonction présente dans un fichier du répertoire `R`)
- `arguments`: décrit chacun des arguments de la fonction
- `details`: (facultatif) description plus longue de la fonction
- `value`: (facultatif) donne les objets retournés par la fonction
- `references`: (facultatif) permet de préciser des références
- `author`: l'auteur de cette fonction
- `note`: d'autres notes
- `seealso`: permet de préciser d'autres fonctions qui sont proches ou utilisées par la présente fonction
- `examples`: donne un exemple d'utilisation de la fonction
- `keyword`: donne un mot-clé. Si on veut plusieurs mots clés, il faut plusieurs `keyword`; les mots clés possibles sont listés dans le fichier `/doc/KEYWORDS`.

1.3 Fonctions programmées en `C` ou `fortran`

Il est également possible d'utiliser des fonctions `C` ou des fonctions `fortran` dans un package. Pour plus de détails sur la construction des packages, voir le manuel suivant: `http://cran.at.r-project.org/doc/manuals/R-exts.pdf`.

2 Étapes préliminaires à la compilation

2.1 Sous Linux

Il est nécessaire d'avoir ActivePerl (disponible à l'adresse suivante:
`http://activestate.com/Products/activeperl/`)
pour tester et compiler le package.

2.2 Sous Windows

2.2.1 Installer les logiciels indispensables

Duncan Murdoch propose gratuitement `Rtools`, un executable qui installe automatiquement les logiciels nécessaires à la compilation de packages sous Windows, en particulier `MinGW` et `Perl`. `Rtools` est disponible à cette adresse :
`http://www.murdoch-sutherland.com/Rtools/installer.html`.

Il faut de plus installer `LaTeX` et le `Microsoft HTML compiler` pour créer les documentations du package en `pdf` et `html`.

2.2.2 Modification des variables d'environnement

Aller dans les "propriétés système" de l'ordinateur (clic droit sur l'icône "Poste de travail" du bureau, et choisir "Propriétés"). Choisir l'onglet "Avancé" puis le bouton "Variables d'environnement". Sélectionner "Path" dans les variables système et cliquer sur le bouton "Modifier".

Modifier la variable de telle façon à avoir au début, et dans l'ordre :

```
C:\Rtools\bin;C:\Rtools\perl\bin;

C:\Program Files\MiKTeX2.7\miktex\bin;

C:\Rtools\MinGW\bin;

C:\Program Files\R\R-2.8.0\bin\;

C:\Program Files\HTML Help Workshop\;...(garder le reste tel quel)
```

3 Tester et compiler le package

Il faut maintenant tester le package, c'est à dire vérifier que les aides des fonctions sont bien documentées et que les exemples fonctionnent, puis le compiler.

3.1 Sous Linux

```
> R CMD check MonPackage
```

S'il n'y a pas d'erreur, on peut alors compiler le package:

```
> R CMD build MonPackage
```

Mettre ensuite le package dans le répertoire `R/version.de.R/src/library`. Pour tester le package, il faut lancer à partir d'une fenêtre DOS:

```
cd C:\R\rw1081\src\gnuwin32 make pkgcheck-pkgname
```

Il faut éliminer les éventuelles erreurs et recommencer ce test jusqu'à ce qu'il n'y ait plus d'erreur.

Ensuite, pour construire le package:

```
make pkg-MonPackage
```

Le package MonPackage est construit et installé dans `R/version.de.R/library`. Il est donc prêt à être utilisé par `R` (pour l'appeler, il suffit de faire **library**`(MonPackage)`.

Il est également possible de zipper ce package pour le rendre disponible à d'autres utilisateurs.

3.2 Sous Windows

Nous conseillons de créer un répertoire `MesPackages` à la racine du disque `C:`, et d'y mettre l'ensemble des dossiers et fichiers générés par la fonction **package.skeleton**.

Ouvrir l'invite de commandes DOS et se mettre dans le répertoire "MesPackages" :

```
C:\>cd MesPackages
```

Pour tester le package :

```
C:\>cd MesPackages> R CMD check MonPackage
```

Et pour le compiler :

```
C:\>cd MesPackages> R CMD build --binary MonPackage
```

Une version zippée du package est alors crée dans le repertoire "MesPackages".