Statistiques appliquées aux sciences

SCI 1018

Introduction à R – les packages

Marc J. Mazerolle

Institut de recherche sur les forêts, Université du Québec en Abitibi-Témiscamingue

Table des matières

1 Les packages

Toutes les fonctions de `R` sont regroupées en bibliothèques de fonctions appelées **packages**. Ces bibliothèques de fonctions permettent de réaliser des tâches particulière regroupées sous un même thème (p. ex., faire des graphiques, travailler avec des données géoréférencées). Par exemple, plusieurs fonctions graphiques sont incluses dans le package `graphics`, alors que le package `base` comprend des fonctions de base permettant à `R` d'agir comme un langage, et que le package `stats` regroupe plusieurs fonctions pour exécuter des analyses statistiques classiques. Lors de l'installation de `R`, une série de packages sont installés par défaut. Quelques packages sont activés au démarrage de `R`, et chacun contient des fonctions de base pour manipuler des objets, importer des fichiers, exécuter des analyses classiques, et créer plusieurs types de graphiques. Dès que l'on désire effectuer des manipulations, des analyses ou des graphiques plus complexes, ou accéder à un jeu de données dans un format moins conventionnel, on doit activer le package qui contient la fonction que l'on désire utiliser.

1.1 Activer un package

Pour activer un package, il suffit d'utiliser la fonction `library( )`, en spécifiant le nom du package entre parenthèses. Par exemple, si on veut accéder à une fonction du package `MASS`, on procéderait comme suit :

```
> library(MASS) #pour charger le package MASS
```

Pour connaître les packages déja installés sur notre ordinateur, on utilise la commande `library( )`, sans spécifier quoi que ce soit entre parenthèses. Autrement, il est aussi possible d'utiliser `help.start( )` pour naviguer vers la page web des packages à l'aide d'hyperliens. En consultant la liste des packages sur cette page, on peut voir les fonctions incluses dans chacun des packages installés.

1.2 Installer un package

Lorsque la fonction que nous voulons utiliser n'est pas incluse dans un des packages installés par défaut, nous devons télécharger le package contenant la fonction d'intérêt à partir du site de `CRAN` (Comprehensive R Archive Network) ou l'un de ses sites miroirs. Ce site a été décrit dans le premier document sur l'installation de `R` (Installation de R et choix d'un éditeur) et veuillez y référer pour de plus amples informations.

Le nombre de packages disponibles augmente continuellement. Ces packages contiennent des fonctions créées par des utilisateurs et sont offerts aux utilisateurs à travers la planète afin de réaliser des tâches spécifiques dans `R` fig. 1, 2). En date du 18 juillet 2012, on en comptait plus de 3700.

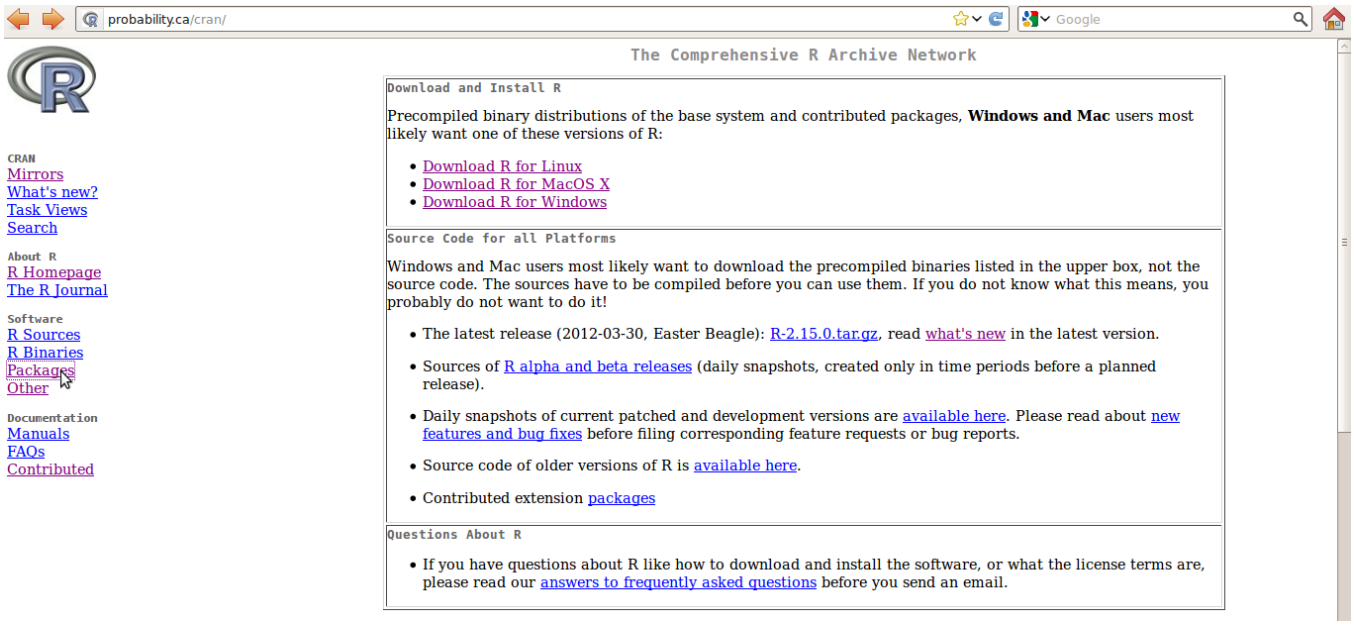

FIGURE 1 – Section du site de `R` et de ses sites miroirs réservée aux packages.

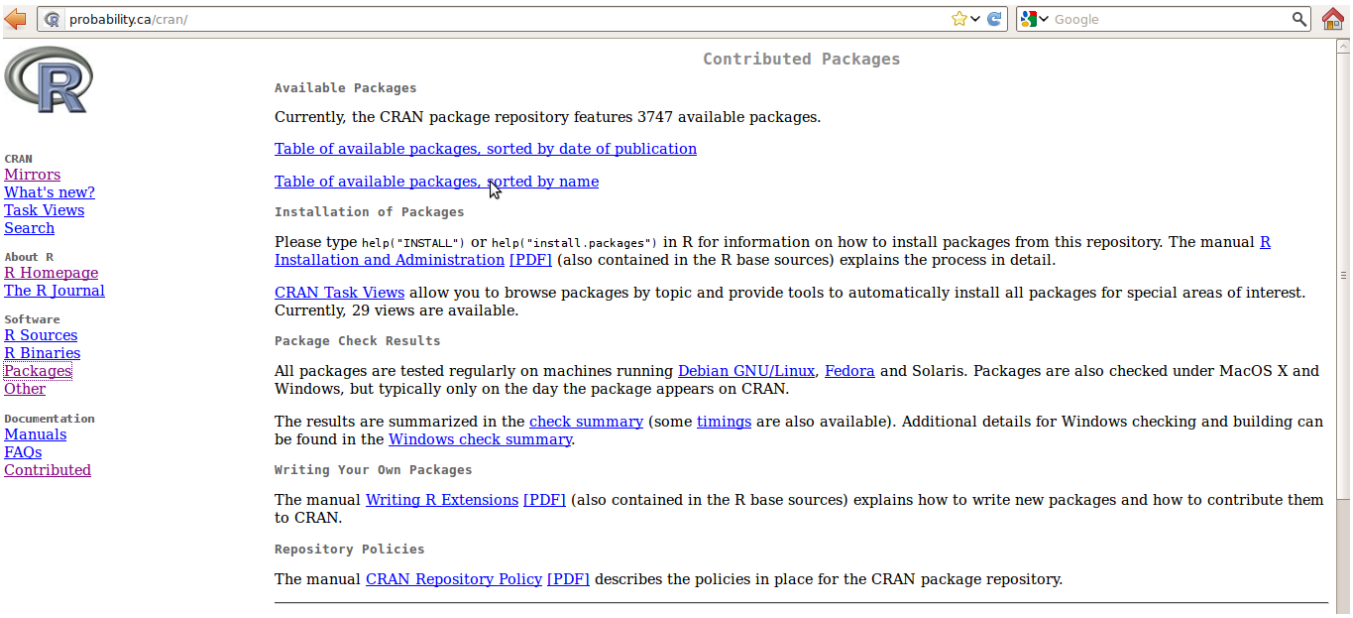

FIGURE 2 – Page des packages contenant des liens vers la liste complète des packages.

La liste des packages par ordre alphabétique permet de visualiser rapidement tous les packages disponibles dans `CRAN` (fig. 3).

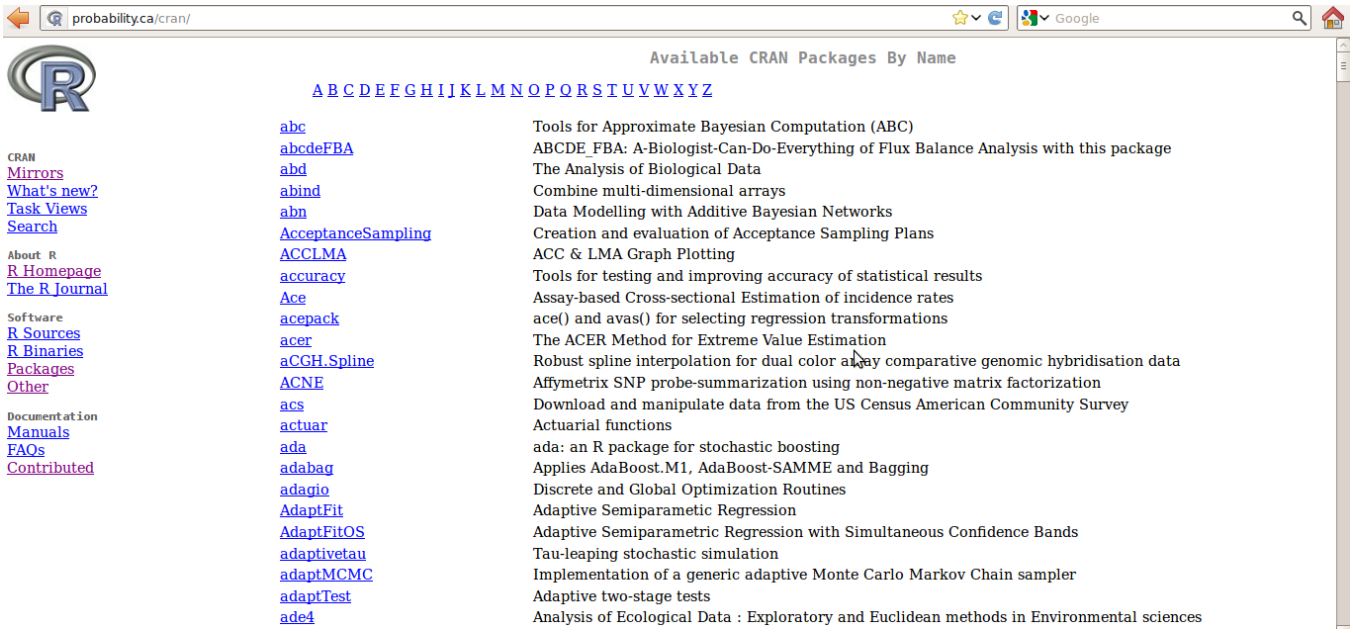

FIGURE 3 – Liste des packages par ordre alphabétique.

Lorsqu'on cherche le package qui contient une fonction particulière dont on connaît le nom, par exemple, une fonction mentionnée dans un article scientifique, dans une page web ou par un collègue, on peut faire une recherche directement à l'aide de `RSiteSearch( )` [1].

Pour trouver une fonction qui réalise une tâche quelconque, on peut également faire appel à `RSiteSearch( )`. Disons que l'on désire importer le fichier `vers.xls` dans `R` sans avoir à convertir ce fichier `Excel` en format texte. Nous devrons chercher s'il existe une fonction pour importer des fichiers `Excel` directement dans `R`. Pour trouver cette fonction, on procède ainsi :

```
> RSiteSearch("read Excel files")
```

1. Rappel : il faut être connecté à l'internet pour utiliser la fonction `RSiteSearch( )` puisque cette fonction va chercher dans les archives de `R` sur le web.

```
Une requête de recherche a été soumise à http://search.r-project.org
La page de résultats devrait s'ouvrir dans votre navigateur
```

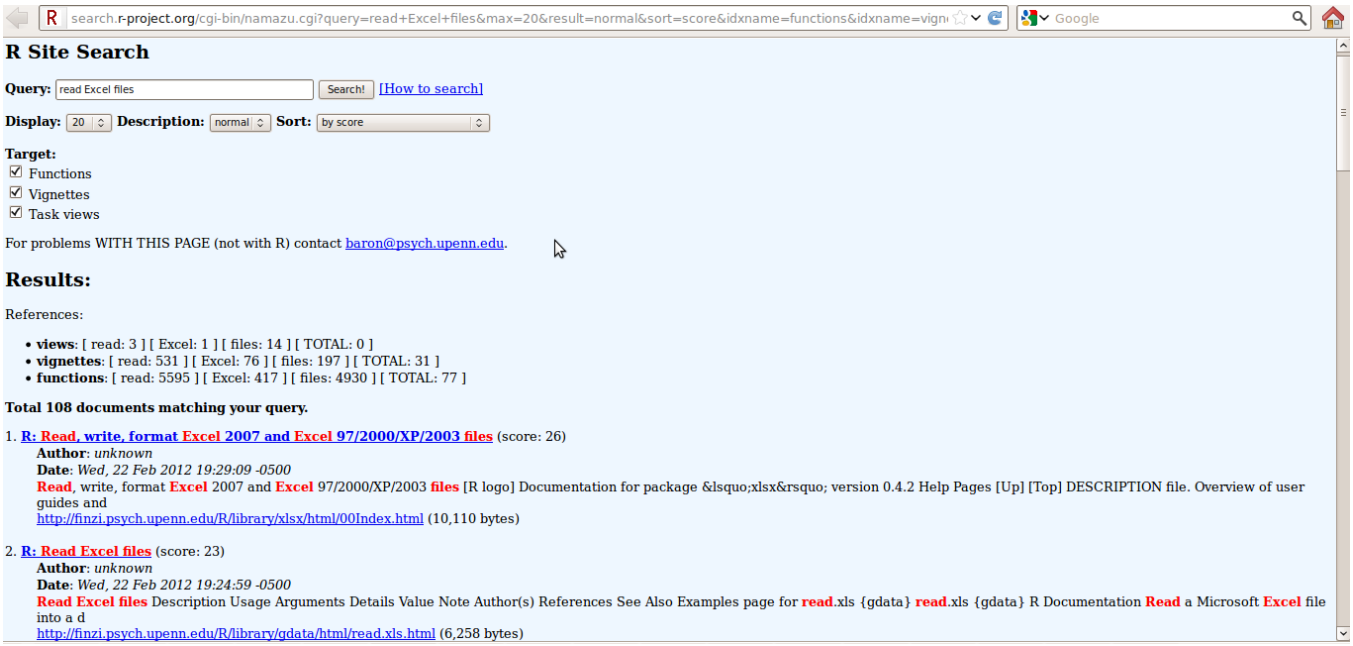

FIGURE 4 – Résultat de `RSiteSearch("read Excel files")`. On remarque que les deux premiers résultats sont des fonctions qui permettent d'importer des fichiers `Excel`. Ces fonctions proviennent des packages `xlsx` et `gdata`, tels qu'indiqués par les adresses `http`.

Les deux premiers résultats de `RSiteSearch( )` sont des fonctions qui permettent de lire des fichiers de données en format `Excel` (fig. 4). On peut par la suite aller voir directement dans la liste des packages de `R` pour le package `gdata` (fig. 5). Cette page contient plusieurs fichiers, dont le manuel (*Reference manual*) en format `pdf`. Ce document inclut les pages d'aide de toutes les fonctions de ce package. C'est en inspectant ce document que l'on peut comprendre comment utiliser les fonctions d'un nouveau package. Les sections d'exemples (*Examples*) illustrent l'utilisation des factions et s'avèrent particulièrement utiles.

Dans notre cas, on remarquera la présence de la fonction `read.xls( )` qui permet d'importer directement un jeu de données stocké dans un fichier `Excel`. À noter que le jeu de

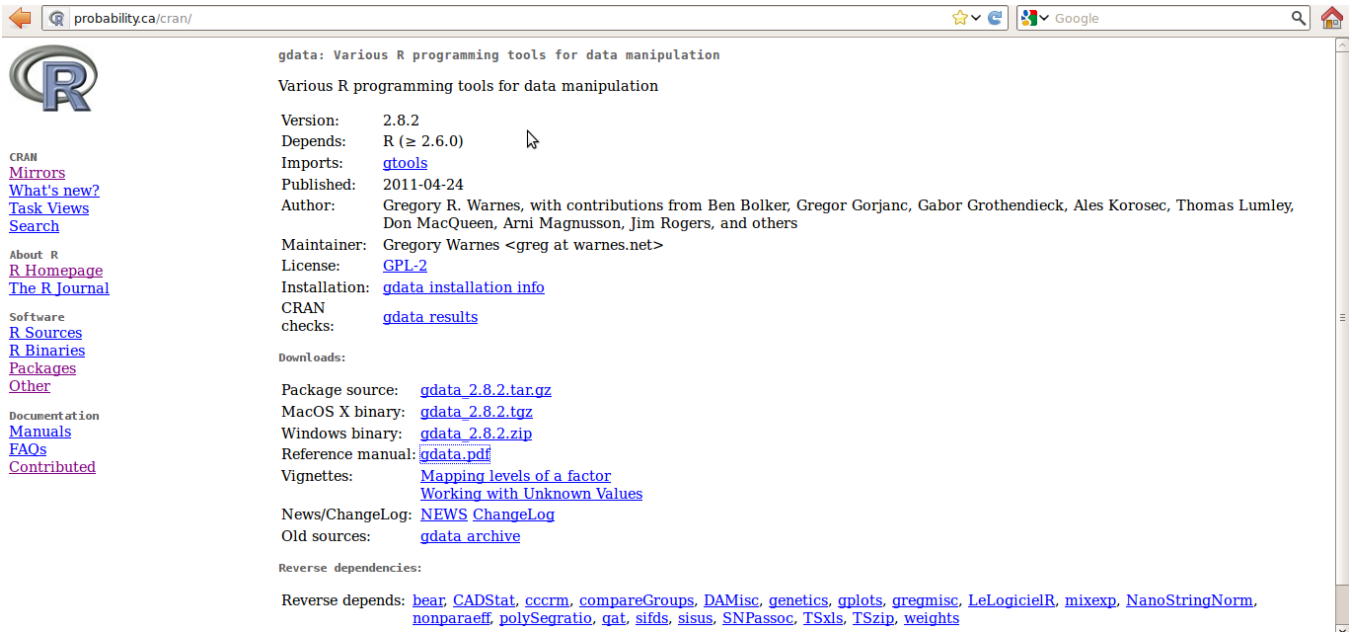

FIGURE 5 – Page du package `gdata` avec l'archive des fichiers comportant toutes les fonctions codées en `R` (fichier `tar.gz`), ainsi que le manuel en `pdf`.

données doit tout de même respecter les conventions pour l'importation des jeux de données, telles que le nom des variables, la présence de données manquantes, le caractère spécifiant la décimale et la structure du fichier (format long). Ainsi, il faudra tout de même choisir la feuille du fichier `Excel` sur laquelle se trouvent les données.

L'installation d'un package s'effectue à l'aide de la fonction `install.packages( )` en spécifiant le nom du package d'intérêt entre guillemets à l'aide de l'argument `pkgs`. Il faut donc être branché au réseau internet lorsqu'on veut télécharger et installer un package sur notre système. On peut aussi spécifier l'adresse du site miroir de `R` avec l'argument `repos`, sans quoi, une fenêtre s'ouvrira nous demandant de sélectionner un site afin de télécharger les fichiers du package que l'on veut installer.

```
> install.packages(pkgs = "gdata", repos = "http://probability.ca/cran")
```

Cette commande lance le téléchargement et l'installation du package `gdata` à partir du site de miroir de Toronto. Dans certains cas, des fonctions d'un package dépendent de fonctions provenant d'autres packages, c'est ce qu'on appelle la dépendance entre deux packages. La

gestion des dépendances est faite automatiquement par R. En d'autres mots, si le package que nous désirons installer dépend d'un autre package, les deux seront installés automatiquement. Une fois installé, le package peut être utilisé à n'importe quel moment. Pour utiliser une fonction d'un package nouvellement installé, il faut tout d'abord activer le package à l'aide de la fonction `library( )`.

```
> library(gdata)
```

Toutes les fonctions du package activé seront alors accessibles. Nous pourrons ensuite importer le fichier `vers.xls` en spécifiant la feuille sur laquelle se trouvent les données à l'aide de l'argument `sheet` :

```
> ##importation du fichier
> ##attention au bon chemin menant vers
> ##le fichier
> vers <- read.xls(xls = "vers.xls", sheet = 2)
> head(vers)
             Site Superficie Pente Vegetation pH.sol Humide
1     Nashs.Field        3.6    11    Prairie    4.1  FALSE
2  Silwood.Bottom        5.1     2     Arable    5.2  FALSE
3   Nursery.Field        2.8     3    Prairie    4.3  FALSE
4     Rush.Meadow        2.4     5        Pre    4.9   TRUE
5 Gunness.Thicket        3.8     0  Chaparral    4.2  FALSE
6        Oak.Mead        3.1     2    Prairie    3.9  FALSE
  Densite.vers
1            4
2            7
3            2
4            5
```

```
5           6
6           2
```

1.3 Mettre à jour un package

Les packages sont mis à jour de temps à autre par leurs auteurs et il est fortement conseillé de vérifier les mises à jour à l'aide de la fonction `update.packages( )`. Évidemment, il faut être branché à l'internet afin de réaliser cette opération. En utilisant cette commande sans rien spécifier entre parenthèses, `R` vérifiera la version des packages installés sur notre système et les comparera aux versions les plus récentes des packages dans les sites miroirs. Si les versions des packages sur notre système sont moins récentes que celles sur le site de `R`, chaque package sera identifié et il faudra indiquer si on désire mettre à jour chacun des packages pours lesquels il existe une mise à jour.

Les mises à jour peuvent amener des correctifs à des fonctions déjà présentes ou ajouter de nouvelles fonctions. Par conséquent, il est fortement recommandé de faire la vérification de temps à autre. Une vérification deux à trois fois par année pourrait suffire si vous utilisez des packages établis depuis plusieurs années. Si vous employez des packages développés récemment (version < 1), il est préférable de vérifier les mises à jour un peu plus souvent.

1.4 Désactiver un package

Lorsque vous avez terminé d'utiliser un package que vous avez activé pendant votre session de travail, il est possible de le désactiver à l'aide de la fonction `detach( )`. Par exemple, pour désactiver le package `gdata` qui a été activé plus haut, on exécute la commande :

```
> detach(package:gdata)
> ##une fois le package désactivé, read.xls n'est plus accessible
> vers <- read.xls(xls = "vers.xls", sheet = 2) #donne un message d'erreur
```

Avec près de 4000 packages de R disponibles sur CRAN, il est fort possible de trouver des fonctions ayant des noms identiques dans des packages différents. Cela peut entraîner des conflits entre certains packages lorsqu'ils sont activés lors de la même session. Dans de tels cas, la fonction du package activé en dernier masquera la fonction du même nom du package activé plus tôt pendant la session de travail. Pour éviter des problèmes, il est préférable d'activer uniquement les packages qui sont utilisés pendant une session de R donnée, plutôt que de systématiquement activer une multitude de packages à chaque fois qu'on travaille dans R.

1.5 Désinstaller un package

En de très rares occasions, nous aurons besoin de désinstaller un package. La fonction `remove.packages( )` permet de désinstaller un package spécifié avec l'argument `pkgs`.

```
> remove.packages(pkgs = "gdata")
```

Une raison possible de désinstaller un package est lorsqu'une mise à jour d'un package n'a pu être réalisée avec succès. Ceci serait indiqué par un message d'erreur lors de la mise à jour. Un changement de dépendances d'un package entre deux versions peut causer une telle erreur. Dans une pareille situation, il suffit de désinstaller l'ancienne version du package à l'aide de `remove.packages( )` et d'installer la nouvelle version du package avec `install.packages( )`.

Index