Table des matières

1 Présentation du langage R

1.1 Bref historique

À l'origine fut le S, un langage pour «programmer avec des données» développé chez Bell Laboratories à partir du milieu des années 1970 par une équipe de chercheurs menée par John M. Chambers. Au fil du temps, le S a connu diverses versions communément identifiées par la couleur du livre dans lequel la nouvelle incarnation était présentée : ...

Dès la fin des années 1980 et pendant près de vingt ans, le S a principalement été popularisé par une mise en œuvre commerciale nommée S-PLUS. En 2008, Lucent Technologies a vendu le langage S à Insightful Corporation, ce qui a effectivement stoppé le développement du langage par ses auteurs originaux. Aujourd'hui, le S est commercialisé de manière relativement confidentielle sous le nom S+ par Tibco Software.

Ce qui a fortement contribué à la perte d'influence de S-PLUS, c'est une nouvelle mise en œuvre du langage développée au milieu des années 1990. Inspirés à la fois par le S et par Scheme (un dérivé du Lisp), Ross Ihaka et Robert Gentleman proposent un langage pour l'analyse de données et les graphiques qu'ils nomment R (Ihaka et Gentleman, 1996). À la suggestion de Martin Maechler de l'ETH de Zurich, les auteurs décident d'intégrer leur nouveau langage au projet GNU (`www.gnu.org`), faisant de R un logiciel libre.

Ainsi disponible gratuitement et ouvert aux contributions de tous, R gagne rapidement en popularité là même où S-PLUS avait acquis ses lettres de noblesse, soit dans les milieux académiques. De pâle copie «*not unlike S*», R devient un concurrent sérieux à S-PLUS, puis le surpasse lorsque les efforts de développement se rangent massivement derrière le projet libre. D'ailleurs John M. Chambers travaille aujourd'hui pour le projet R.

1.2 Description sommaire de R

Le S est

> Ce n'est pas seulement un «autre» environnement statistique (comme SPSS ou SAS, par exemple), mais bien un langage de programmation complet et autonome.
> Inspiré de plusieurs langages, dont l'APL et le Lisp, le S est :
  – interprété (et non compilé) ;
  – sans déclaration obligatoire des variables ;
  – basé sur la notion de vecteur ;
  – particulièrement puissant pour les applications mathématiques et statistiques (et donc actuarielles).

1.3 Les moteurs S

Il existe quelques «moteurs» ou dialectes du langage S.

> Le plus connu est S-Plus, un logiciel commercial de Insightful Corporation (Bell Labs octroie à Insightful la licence exclusive de son système S).
> R, ou GNU S, est une version libre (*Open Source*) .

S-Plus et R constituent tous deux des environnements intégrés de manipulation de données, de calcul et de préparation de graphiques.

1.4 Interfaces pour S-Plus et R

Provenant du monde Unix, tant S-Plus que R sont d'abord et avant tout des applications en ligne de commande (`sqpe.exe` et `rterm.exe` sous Windows).

> S-Plus possède toutefois une interface graphique élaborée permettant d'utiliser le logiciel sans trop connaître le langage de programmation.
> R dispose également d'une interface graphique rudimentaire sous Windows et Mac OS X.
> L'édition sérieuse de code S bénéficie cependant grandement d'un bon éditeur de texte.
> À la question 6.2 de la foire aux questions (FAQ) de R, «Devrais-je utiliser R à l'intérieur de Emacs ?», la réponse est : «Oui, absolument.» Nous partageons cet avis, aussi ce document supposera-t-il que S-Plus ou R sont utilisés à l'intérieur de GNU Emacs avec le mode ESS.
> Autres options : Tinn-R (libre), WinEdt (partagiciel) avec l'ajout R-WinEdt.

1.5 Installation de Emacs avec ESS

Il n'existe pas de procédure d'installation similaire aux autres applications Windows pour la version officielle de GNU Emacs.

> Pour une installation simplifiée de Emacs et ESS, consulter le site Internet

`http://vgoulet.act.ulaval.ca/emacs/`

On y trouve une version modifiée de GNU Emacs pour Windows et des instructions d'installation détaillées.

> Les utilsateurs de Mac OS X devraient installer Aquamacs (`http://aquamacs.org`), une version de GNU Emacs bien intégrée à OS X et contenant déjà ESS.

> L'annexe A présente les plus importantes commandes à connaître pour utiliser efficacement Emacs et le mode ESS.

1.6 Démarrer et quitter S-Plus ou R

On suppose ici que S-Plus ou R sont utilisés à l'intérieur de Emacs.

> Pour démarrer R Rà l'intérieur de Emacs :
`M-x R RET`
puis spécifier un dossier de travail (voir la section 1.8). Une console R est ouverte dans une fenêtre (*buffer* dans la terminologie de Emacs) nommée `*R*`.

> Pour démarrer S-Plus sous Windows, la procédure est similaire, sauf que la commande à utiliser est S+
`M-x Sqpe RET`
Consulter l'annexe B pour de plus amples détails.

> Pour quitter, deux options sont disponibles :
>
> 1. Taper `q()` à la ligne de commande.
> 2. Dans Emacs, faire `C-c C-q`. ESS va alors s'occuper de fermer le processus S ainsi que tous les *buffers* associés à ce processus.

1.7 Stratégies de travail

Il existe principalement deux façons de travailler avec S-Plus et R.

1. Le code est virtuel et les objets sont réels. C'est l'approche qu'encouragent les interfaces graphiques, mais c'est aussi la moins pratique à long terme. On entre des expressions directement à la ligne de commande pour les évaluer immédiatement.

   ```
   > 2 + 3

   [1] 5

   > -2 * 7

   [1] -14

   > exp(1)

   [1] 2.718282

   > log(exp(1))

   [1] 1
   ```

   Les objets créés au cours d'une session de travail sont sauvegardés. Par contre, à moins d'avoir été sauvegardé dans un fichier, le code utilisé pour créer ces objets est perdu lorsque l'on quitte S-Plus ou R.

2. Le code est réel et les objets sont virtuels. C'est l'approche que nous favorisons. Le travail se fait essentiellement dans des fichiers de script (de simples fichiers de texte) dans lesquels sont sauvegardées les expressions (parfois complexes !) et le code des fonctions personnelles. Les objets sont créés au besoin en exécutant le code. Emacs permet ici de passer efficacement des fichiers de script à l'exécution du code :

   i) démarrer un processus S-Plus (`M-x Sqpe`) ou R (`M-x R`) et spécifier le dossier de travail ;

   ii) ouvrir un fichier de script avec `C-x C-f`. Pour créer un nouveau fichier, ouvrir un fichier inexistant ;

iii) positionner le curseur sur une expression et faire `C-c C-n` pour l'évaluer ;

iv) le résultat apparaît dans le *buffer* `*S+6*` ou `*R*`.

1.8 Gestion des projets ou environnements de travail

S-Plus et R ont une manière différente mais tout aussi particulière de sauvegarder les objets créés au cours d'une session de travail.

> Tous deux doivent travailler dans un dossier et non avec des fichiers individuels.

> Dans S-Plus, tout objet créé au cours d'une session de travail est sauvegardé de façon permanente sur le disque dur dans le sous-dossier `__Data` du dossier de travail. S+

> Dans R, Rles objets créés sont conservés en mémoire jusqu'à ce que l'on quitte l'application ou que l'on enregistre le travail avec la commande `save.image()`. L'environnement de travail (*workspace*) est alors sauvegardé dans le fichier `.RData` du dossier de travail.

Le dossier de travail est déterminé au lancement de l'application.

> Avec Emacs et ESS, on doit spécifier le dossier de travail chaque fois que l'on démarre un processus S-Plus ou R.

> Les interfaces graphiques permettent également de spécifier le dossier de travail.

- Dans l'interface graphique de S-Plus, choisir `General Settings` dans le menu `Options`, puis l'onglet `Startup`. Cocher la case `Prompt for project folder`. Consulter également le chapitre 13 du guide de l'utilisateur de S-Plus. S+
- Dans Rl'interface graphique de R, le plus simple consiste à changer le dossier de travail à partir du menu `Fichier|Changer le répertoire courant...` Consulter aussi la *R for Windows FAQ*.

1.9 Consulter l'aide en ligne

Les rubriques d'aide des diverses fonctions disponibles dans S-Plus et R contiennent une foule d'informations ainsi que des exemples d'utilisation. Leur consultation est tout à fait essentielle.

> Pour consulter la rubrique d'aide de la fonction `foo`, on peut entrer à la ligne de commande

```
> ?foo
```

> Dans Emacs, `C-c C-v foo RET` ouvrira la rubrique d'aide de la fonction `foo` dans un nouveau *buffer*.
> Plusieurs touches de raccourcis facilitent la consultation des rubriques d'aide ; voir la section A.6.3.

1.10 Où trouver de la documentation

S-Plus est livré avec quatre livres (disponibles en format PDF depuis le menu `Help` de l'interface graphique), mais aucun ne s'avère vraiment utile pour apprendre le langage S.

Plusieurs livres — en versions papier ou électronique, gratuits ou non — ont été publiés sur S-Plus et R. On trouvera des listes exhaustives dans les sites de Insightful et du projet R :

> `http://www.insightful.com/support/splusbooks.asp`
> `http://www.r-project.org` (dans la section `Documentation`).

De plus, les ouvrages de **?**Venables et Ripley (2002) constituent des références sur le langage S devenues au cours des dernières années des standards *de facto*.

1.11 Exemples

```
### Générer deux vecteurs de nombres pseudo-aléatoires issus
### d'une loi normale centrée réduite.
x <- rnorm(50)
y <- rnorm(x)

### Graphique des couples (x, y).
plot(x, y)

### Graphique d'une approximation de la densité du vecteur x.
plot(density(x))

### Générer la suite 1, 2, ..., 10.
1:10

### La fonction 'seq' sert à générer des suites plus générales.
seq(from = -5, to = 10, by = 3)
seq(from = -5, length = 10)

### La fonction 'rep' sert à répéter des valeurs.
rep(1, 5)        # répéter 1 cinq fois
rep(1:5, 5)      # répéter le vecteur 1,...,5 cinq fois
```

```
rep(1:5, each = 5) # répéter chaque élément du vecteur cinq fois

### Arithmétique vectorielle.
v <- 1:12        # initialisation d'un vecteur
v + 2            # additionner 2 à chaque élément de v
v * -12:-1       # produit élément par élément
v + 1:3          # le vecteur le plus court est recyclé

### Vecteur de nombres uniformes sur l'intervalle [1, 10].
v <- runif(12, min = 1, max = 10)
v

### Pour afficher le résultat d'une affectation, placer la
### commande entre parenthèses.
( v <- runif(12, min = 1, max = 10) )

### Arrondi des valeurs de v à l'entier près.
( v <- round(v) )

### Créer une matrice 3 x 4 à partir des valeurs de
### v. Remarquer que la matrice est remplie par colonne.
( m <- matrix(v, nrow = 3, ncol = 4) )

### Les opérateurs arithmétiques de base s'appliquent aux
### matrices comme aux vecteurs.
m + 2
m * 3
m ^ 2

### Éliminer la quatrième colonne afin d'obtenir une matrice
### carrée.
( m <- m[,-4] )

### Transposée et inverse de la matrice m.
t(m)
solve(m)

### Produit matriciel.
m %*% m               # produit de m avec elle-même
m %*% solve(m)        # produit de m avec son inverse
round(m %*% solve(m)) # l'arrondi donne la matrice identité

### Consulter la rubrique d'aide de la fonction 'solve'.
?solve
```

```
### Liste des objets dans l'espace de travail.
ls()

### Nettoyage.
rm(x, y, v, m)
```

1.12 Exercices

1.1 Démarrer un processus S-Plus ou R à l'intérieur de Emacs.

1.2 Exécuter un à un les exemples de la section précédente. Une version électronique du code de cette section est disponible dans le site mentionné dans la préface.

1.3 Consulter les rubriques d'aide d'une ou plusieurs des fonctions rencontrées lors de l'exercice précédent. Observer d'abord comment les rubriques d'aide sont structurées — elles sont toutes identiques — puis exécuter quelques lignes d'exemples.

1.4 Lire le chapitre 1 de Venables et Ripley (2002) et exécuter les commandes de l'exemple de session de travail de la section 1.3. Bien que davantage orienté vers les applications statistiques que vers la programmation, cet exemple démontre quelques-unes des possibilités du langage S.

Bibliographie

Ihaka, R. et R. Gentleman. 1996, «R: A language for data analysis and graphics», *Journal of Computational and Graphical Statistics*, vol. 5, n^{o} 3, p. 299–314.

Venables, W. N. et B. D. Ripley. 2002, *Modern Applied Statistics with S*, 4^{e} éd., Springer, New York, ISBN 0-3879545-7-0.

Index

Les numéros de page en caractères gras indiquent les pages où les concepts sont introduits, définis ou expliqués.