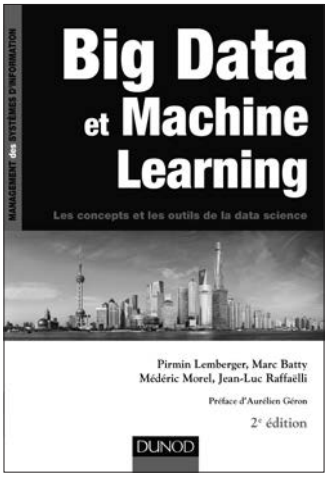
LE LANGAGE R
AU QUOTIDIEN

SAS l'essentiel

O. Decourt (2011)

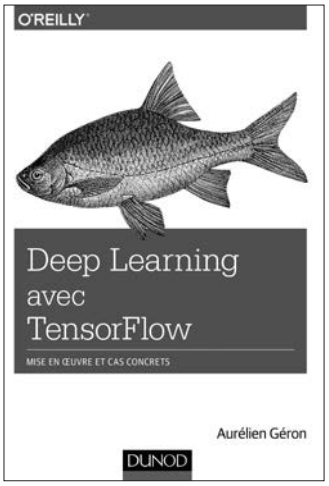


Big data et machine learning

P. Lemberger et al.

Machine learning avec Scikit learn

A. Géron



Deep learning avec TensorFlow

A. Géron

LE LANGAGE R AU QUOTIDIEN

Traitement et analyse de données volumineuses

Olivier Decourt

DUNOD

11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-077076-2

TABLE DES MATIÈRES

AVANT-PROPOS

1. 1 À QUI S'ADRESSE CE LIVRE ?

Le projet de ce livre part d'un constat : il est très facile de trouver de l'information sur R, sur le code, y compris dans de nombreux blogs, forums et articles gratuits sur Internet, mais on trouve peu d'informations synthétiques sur les besoins du quotidien : importer, gérer, organiser, faire parler et exporter des données. De vraies données. De grosses bases de données.

Ce livre est conçu pour résoudre les tracas de tous les jours. Tous les codes présentés ici ont été testés sur différents fichiers, gros et petits. Tous les exemples se rattachent à des problématiques très concrètes en milieu professionnel. Si vous cherchez le code le plus optimal pour une jointure, le calcul d'un âge à partir d'une date de naissance ou la production d'un diagramme en bâtons, vous êtes au bon endroit.

Si vous utilisez R dans vos études, ce livre vous aidera au cours de vos projets, de vos stages et dans votre vie professionnelle. Si vous avez déjà un emploi, j'espère que ce livre restera à portée de main pour vous soulager de petits tracas récurrents ou ponctuels dans votre écriture de code R.

Ce livre s'adresse autant à des professionnels qui découvrent R ou l'utilisent régulièrement, qui doivent migrer vers R des codes venant d'autres logiciels de traitement de données ou de statistiques, qu'à des étudiants qui voudraient un compagnon de papier pour leurs projets et leurs stages sous R.

1. 2 ORGANISATION DE CE LIVRE

La première partie est une introduction à R pour ceux qui ne connaissent pas du tout le logiciel et le langage. Un bref historique est suivi de la présentation des fichiers qu'utilise R et des interfaces sous lesquelles on peut exploiter sa puissance. Le deuxième chapitre traite des notions fondamentales dans la logique de R : les types de données et d'objets.

La deuxième partie, la plus longue, est consacrée à la manipulation de données : imports, requêtes, création de nouvelles colonnes, fusions, tris et transpositions. L'aspect le plus quotidien de R se trouve dans ces pages. Comme le langage R s'enrichit très rapidement, des syntaxes historiques y sont présentées comme des fonctions plus récentes, avec des comparatifs de performance.

La troisième partie propose d'exploiter les données préparées lors de la partie précédente : statistiques, graphiques et reportings sont au programme, ainsi que des exports qui permettront également une diffusion et une exploitation sous d'autres logiciels.

1. 3 REMERCIEMENTS

On ne fait jamais un livre tout seul, et celui-ci encore moins que tous ceux que j'avais pu écrire auparavant. Je tiens avant tout à remercier l'équipe de Dunod : Jean-Luc Blanc en particulier, dont la confiance m'est un trésor précieux.

Ce livre ne serait absolument pas le même – et franchement, il serait beaucoup moins bien – s'il n'était pas passé entre les mains et sous les yeux de mes relecteurs et relectrices. Je suis extrêmement redevable à Sébastien Durier, Antony Briant, Stéphane Tufféry et Fabienne Gire du temps et de l'énergie qu'ils ont déployé par pure amitié sur ce projet. Merci Sébastien pour ta pédagogie. Merci Antony pour ton énergie communicative. Merci Stéphane pour tes échanges toujours enrichissants. Merci Fabienne pour ta guérilla minutieuse contre mes points-virgules incessants (peut-être un reliquat de mes années de SAS ?).

Enfin, ma famille à qui j'ai encore dû voler du temps. Merci à Vanessa et à Clémentine de ne m'en avoir pas tenu rigueur et de m'avoir encouragé tout au long de ces mois d'écriture et de recherches.

DONNÉES UTILISÉES COMME EXEMPLES DANS CE LIVRE

1. 1 DONNÉES AIRBNB

Tout au long des chapitres qui vont suivre, nous utiliserons le même jeu de données. Il sert de fil rouge et montre les capacités de R pour l'exploration et la mise en valeur des données, y compris de bases volumineuses : un des fichiers manipulés contient presque 10 millions de lignes.

Ce sont des données de location de logements du site AirBnB. Ces données ont été agrégées et mises à disposition par le site insideAirBnB.com qui propose des jeux de données pour de nombreuses villes de par le monde. Nous avons travaillé avec le jeu de données parisien accessible à partir de la page `http://insideairbnb.com/paris/` en cliquant sur le bouton GET THE DATA en haut de l'écran.

Seule une partie des fichiers disponibles en ligne a été exploitée : le fichier listings.csv.gz des logements et le fichier calendar.csv.gz des croisements entre logements et dates indiquant les prix et les disponibilités de chacun.

Les fichiers mis en ligne par insideAirBnB.com ayant toujours la même structure, on pourra répéter les opérations des exemples de ce livre sur d'autres villes si on le souhaite !

Afin que le code des exemples reste applicable, dans la mesure du possible, à des données concernant d'autres villes, ou à d'autres extractions sur des périodes différentes, nous avons conservé les noms des colonnes et leurs valeurs telles quelles. Aucun nom, aucune valeur n'a été traduite en français.

Les données étant celles saisies sur AirBnB, elles peuvent contenir des anomalies et des inexactitudes dont ni l'auteur, ni l'éditeur, ne sauraient être tenus pour responsables !

Afin de pouvoir tester les imports du chapitre 3, les fichiers sont récupérés directement sur le site Internet de l'auteur (`http://www.od-datamining.com`). À partir du chapitre 4, on suppose que les données sont disponibles dans R : pour cela, il faudra exécuter le code de l'exemple 3.06.

Ces données au format R comprennent trois éléments distincts : calendar, flats et houses. Ces trois objets reprennent les informations des fichiers CSV indiqués ci-dessus : flats et houses sont des sous-ensembles de listings.csv.gz après expansion de l'archive et import dans R ; calendar reprend le fichier calendar.csv.gz après expansion de l'archive et import dans R.

Les informations contenues dans flats et houses sont identiques, seul le type de logement diffère. L'objet flats contient uniquement des appartements (valeurs « Apartment », « Condominium » [appartement partagé type colocation] et « Loft » de la colonne property_type) tandis que houses ne présente que des maisons (valeurs « House », « Townhouse » [maison de ville] et « Villa »). Il existe encore d'autres types de logements (bateaux, tipi, chalet, et même igloo – à Paris ![1]) qui sont regroupés dans un fichier Excel other.xlsx également disponible sur le site de l'auteur. Ce classeur Excel sera également importé au chapitre 3.

1. Après quelques recherches, il s'agissait apparemment d'une installation éphémère dans une galerie d'art.

L'objet flats compte 51 494 lignes, houses, 639 et other 590, soit 52 7223 logements. Ces trois objets contiennent 21 informations décrites dans le tableau 1.2. Les noms des colonnes et les valeurs sont en anglais, directement repris du formatage fait par InsideAirBnB.com.

Seule une partie des informations disponibles sur InsideAirBnB.com a été conservée pour les objets que nous manipulerons.

Tableau 1.2 Description du contenu des objets flats et houses et du classeur Excel other.xlsx

Nom de la colonne	Description
`id`	Identifiant du logement, unique
`listing_url`	Adresse Internet de l'annonce sur le site AirBnB.com
`last_scraped`	Date de dernière extraction des informations sur ce logement
`host_id`	Identifiant de l'hôte ; permet de compter le nombre de logements proposés par la même personne (record dans l'objet flats : 153 !)
`host_name`	Nom sous lequel l'hôte se présente sur le site AirBnB.com
`host_since`	Date d'inscription de l'hôte
`host_location`	Lieu de résidence de l'hôte
`neighbourhood`	Quartier dans lequel le logement est localisé (plus de 60 valeurs dans flats) tel que décrit dans l'annonce
`neighbourhood_cleansed`	Version normalisée du quartier (seulement 20 valeurs dans flats)
`zipcode`	Code postal de rattachement du logement, certaines valeurs étant assez surprenantes. Normalement toutes devraient être sur 5 chiffres commençant par 750 suivi d'un nombre entre 01 et 20.
`property_type`	Type de logement (maison, appartement, igloo, etc.)
`room_type`	Type d'hébergement proposé : logement entier (« Entire home/apt »), chambre privatisée (« Private room ») ou chambre en partage (« Shared room »)
`accommodates`	Nombre de personnes maximum pouvant être accueillies dans ce logement
`bathrooms`	Nombre de salles de bains. Les valeurs ne sont pas forcément entières, il y a des demi-salles de bains !
`bedrooms`	Nombre de chambres. Ici les valeurs sont entières ou manquantes
`beds`	Nombre de lits. Ici les valeurs sont entières ou manquantes

Nom de la colonne	Description
price	Prix de location par nuit en euros, tel que visible sur la page de présentation du logement. Il s'agit en général du prix minimum et des tarifs plus élevés peuvent être demandés pour certaines dates, tandis que des réductions pour des séjours de longue durée peuvent également être consentis par l'hôte
number_of_reviews	Nombre d'avis sur ce logement visibles sur le site AirBnB.com
review_scores_rating	Note moyenne de ces avis, entre 0 et 100. Les logements sans avis ont des valeurs manquantes
cancellation_policy	Conditions d'annulation de la location (remboursement du prix déjà payé, partiel ou intégral) : quatre valeurs « super strict », « strict », « moderate » et « flexible ». Pour plus de détails se reporter au site AirBnB.com
amenities	Equipements disponibles dans ce logement, sous forme d'une liste où les éléments sont séparés par le caractère | La notion d'équipement intègre aussi bien des biens matériels comme une télévision («TV ») ou un lave-linge («Washer »), des services comme l'accès Internet ou le wifi («Wireless Internet »), une politique d'accueil (fumeurs acceptés [« Smoking Allowed »]) ou des mises en garde comme la présence de chats ou de chiens dans le logement [« Pets live on this property »]

L'objet calendar contient quant à lui beaucoup moins de colonnes… et beaucoup plus de lignes (près de 10 millions !). Il contient les disponibilités annoncées entre le 3 juillet 2016 et le 3 juillet 2017, à la date 3 juillet 2016.

- ✓ `listing_id` est l'identifiant du logement. Ses valeurs sont identiques à celles de la colonne id des objets flats et houses. Cette colonne sert de clé de jointure pour rapprocher ces objets du calendrier des prix
- ✓ `date` est la date de location considérée. Les combinaisons de date et d'identifiant logement sont uniques
- ✓ `price` est le prix en euros demandé pour une nuitée dans ce logement à cette date. Les valeurs manquantes ont été éliminées : si une date est absente pour un logement, c'est qu'il n'est pas proposé à la location. En comptant les lignes par identifiant logement, on peut donc savoir combien de jours ce logement était disponible (365 au maximum).

Chaque ligne de calendar correspond au croisement d'une date et d'un logement. En théorie on aurait 52 723 logements * 366 jours = 19 296 618 lignes. Les dates où le logement n'était pas disponible ont été supprimées, ce qui laisse au final 9 461 700 lignes. En moyenne, un logement était donc disponible 9 461 700 / 52 723 = 179,46 jours dans l'année, soit 6 mois sur 12 , sur la période 2016-2017. Ce n'est qu'une moyenne, mais elle montre que pour certains logements, ce n'est pas du tout une activité de location *ponctuelle* qui est proposée.

1. 2 LOGICIELS

Les exemples de ce livre ont été testés sur Windows 7 64bits, avec R version 3.4.1 64 bits (packages à jour du 3 juillet 2017) et Microsoft R Open version 3.4.0 64 bits (packages à jour du 1er mai 2017), l'un et l'autre via RStudio version 1.0.143, sur une machine disposant de 32 Go de RAM.

Les scripts des exemples ont aussi été exécutés avec succès sur une machine moins puissante, utilisant R version 3.3.3 i386, c'est-à-dire 32 bits, avec 4 Go de RAM, toujours sous Windows 7.

Pour pouvoir exécuter vous aussi les exemples, vous devez avoir installé R sur votre ordinateur (on le télécharge depuis https://cran.r-project.org/) et, pour votre confort, également RStudio (les téléchargements se font depuis https://www.rstudio.com/products/rstudio/download/). Le chapitre 3 contient les scripts pour récupérer les données décrites ci-dessus directement depuis le site Internet de l'auteur.

PREMIÈRE PARTIE

Découvrir R

Cette première partie présente le langage R et les logiciels qui l'utilisent.

Un peu d'histoire, une présentation des interfaces, des principaux concepts : voici ce que l'on trouvera dans cette partie.

INTRODUCTION À R

Objectif

Ce tout premier chapitre plante le décor. Il présente rapidement l'historique et le développement de R, les fichiers associés et les environnements dans lesquels vous pourrez l'utiliser.

1. 1 ORIGINES DE R

R est un logiciel et un langage. Le logiciel R est gratuit et *open source*. Tout le monde peut l'installer – il est en libre téléchargement sur Internet, avec des versions pour Windows 32 bits, Windows 64 bits, Unix/Linux et Mac. Le site de téléchargement est http://www.cran.r-project.org/.

1.1.1 R et S-Plus

Le langage de programmation R est une version libre et gratuite d'un langage de traitement statistique appelé S puis S-Plus. Ce langage et son logiciel compilateur, développés au milieu des années 1970 chez Bell Laboratories autour de John Chambers, ont été commercialisés à partir de 1980. Les premières versions de R ont été développées par des chercheurs de l'université d'Auckland en Nouvelle-Zélande, Robert Gentleman et Ross Ihaka, en 1993. Leur idée était de reprendre la logique et la syntaxe du code S, avec parfois quelques aménagements, et d'en faire un projet collaboratif où chacun pourrait ajouter ses propres fonctionnalités et les mettre à disposition des autres utilisateurs.

En 1997, un groupe de développeurs est créé pour assurer la maintenance et l'évolution des fonctionnalités basiques de R et du logiciel associé, regroupant une interface de programmation et un compilateur. Ce groupe d'une vingtaine de personnes aujourd'hui est appelé « *R Core Team* ». On y retrouve les développeurs initiaux (Gentleman et Ihaka) et également Chambers.

La première version « stable » de R, la 1.0.0 est publiée fin février 2000.

1.1.2 CRAN et projet R

La Fondation R, créée en 2003, est une organisation à but non lucratif qui détient les droits sur le logiciel R et sa documentation. Son but est d'être un interlocuteur unique pour tous

ceux qui veulent contribuer au développement de R à titre gratuit et soumettre leurs développements pour une mise à disposition « officielle ».

Le site Internet CRAN (*Comprehensive R Archive Network*, c'est-à-dire « réseau d'archives exhaustif sur R ») met à disposition de manière centralisée les éléments de R : l'exécutable qui sert de compilateur, intégrant les fonctionnalités de base, et une liste de « packages », des contributions d'utilisateurs pouvant être ajoutées à la demande aux fonctionnalités natives de R via de nouvelles fonctions. Le site de CRAN est dupliqué sur de nombreux serveurs de par le monde, appelés *miroirs*. Les mises à jour sont synchronisées entre tous les miroirs.

On pourrait penser que l'intégralité des ressources nécessaires au fonctionnement de R se trouve sur les serveurs de CRAN ; mais malheureusement non. On trouve aussi des packages sur les sites personnels de leurs auteurs et pas sur CRAN – soit parce qu'ils n'ont pas été acceptés par le R Core Team, soit parce qu'ils sont encore en cours de développement, soit parce qu'ils sont obsolètes. Il est déconseillé d'avoir recours à de tels packages.

On trouve aussi deux sites Internet offrant de nombreux packages « officieux » à télécharger : Bioconductor qui est spécialisé dans les fonctionnalités de bio-informatique (analyse génomique) et GitHub qui propose des versions en cours de développement, à divers stades d'achèvement et de fiabilité. Il existe des doublons entre les packages proposés par CRAN et par Bioconductor et il est conseillé de privilégier les ressources de CRAN. Quant à installer un package en cours de développement depuis GitHub sans vouloir soi-même contribuer à son amélioration, c'est une entreprise extrêmement périlleuse, surtout dans un contexte professionnel.

1.1.3 Logiciels utilisant le langage R

L'interpréteur de langage R fourni par CRAN et la Fondation R n'est pas le seul logiciel capable de comprendre et d'exécuter du code R. D'autres développeurs proposent en parallèle leurs outils, surtout en se basant sur l'optimisation des opérations matricielles.

Certains compilateurs sont payants, avec un gain de temps de traitement mis en avant par les entreprises qui les commercialisent : ainsi le logiciel Revolution R, qui a été racheté par Microsoft en 2015 et se vend désormais sous le nom de Microsoft R pour la version payante. « Microsoft R Open » en est une version gratuite, avec comme site compagnon MRAN, équivalent de CRAN chez Microsoft. Oracle commercialise aussi un compilateur R sachant se brancher aisément sur ses bases de données. Ces compilateurs payants sont totalement compatibles avec l'ensemble des packages CRAN (d'après les entreprises qui les vendent) et proposent des packages « maison » introuvables ailleurs.

D'autres compilateurs sont gratuits, comme pqR (pour *pretty quick R*, soit R plutôt rapide) ou Renjin qui propose un compilateur R intégré à une machine virtuelle Java (JVM).

À l'heure actuelle, la version CRAN de R reste la norme. Quelques entreprises désireuses de se connecter à de grosses bases de données, à un environnement de type « cloud » sous Hadoop, ou en quête de performance, utilisent la version Microsoft R.

1.2 FICHIERS GÉRÉS PAR R

En travaillant avec R, vous allez générer quelques fichiers. Ils sont principalement de deux types : les **scripts** (fichiers .R) et les **environnements** (fichiers .Rdata). Les fichiers .Rhistory, qui contiennent des listes de code déjà exécutés, viennent compléter cette palette, bien qu'étant très rarement utiles. Les objets, qui rassemblent tout ce qui est utile à R pendant qu'il travaille, sont stockés en mémoire vive.

1.2.1 Scripts

Un script est un ensemble de code R et de commentaires. Il s'agit de texte brut, qu'on peut ouvrir avec un logiciel type Bloc-notes ou dans un traitement de texte. Il n'y a pas de longueur maximale ou minimale pour un script.

Les commentaires qui y figurent commencent par le caractère dièse (#) et se terminent en fin de ligne. Si on veut commenter sur plusieurs lignes, chaque ligne commence par #.

Il est possible (mais pas conseillé) de rédiger tout ce qu'on souhaite faire dans un unique script, et de l'exécuter ensuite par morceaux. Cependant, il est plus agréable à l'usage de fractionner son travail en plusieurs scripts portant des noms explicites, numérotés s'ils doivent absolument être exécutés dans un certain ordre : par exemple « 01_recuperation_donnees.R », « 02_nettoyage_base.R », « 03_analyse_statistique.R », etc. Les noms de scripts sont soumis aux règles en vigueur sur votre système d'exploitation. Si vous mutualisez votre travail avec des collègues travaillant sur d'autres systèmes d'exploitation, donnez des noms à vos scripts avec le minimum de caractères spéciaux, comme dans l'exemple ci-dessus : les caractères non accentués, les chiffres et le caractère *underscore* (_) seront toujours acceptés, et les noms de moins de 25 caractères aussi.

1.2.2 Objets

On appelle **objet** tout élément que R conserve en mémoire de manière individuelle. Il peut s'agir d'un ensemble de données, ou une seule valeur associée à un nom, ou encore une fonction. En général, un objet porte un nom. La section 2.2.2 reviendra en détail sur les normes à respecter pour nommer un objet.

Les objets ne sont pas stockés physiquement sur le disque dur par défaut. Ils sont conservés en mémoire vive, ce qui suppose parfois que votre ordinateur ait suffisamment de mémoire (RAM) disponible. Certains packages comme `ff` proposent de les stocker sur le disque dur de manière transparente.

On peut sauvegarder, par une demande explicite, des objets individuellement sous forme de fichiers .RDS, comme on le verra à la section 9.1.1.

1.2.3 Environnements

Un environnement est un ensemble d'objets. L'environnement global est celui dans lequel les nouveaux objets sont créés par défaut. L'exécution d'une fonction s'accompagne de la création d'un environnement temporaire dans lequel des objets nécessaires aux calculs de la fonction sont hébergés. Le contenu de l'environnement spécifique à la fonction est supprimé quand celle-ci n'est plus utilisée.

On peut sauvegarder l'intégralité de l'environnement global avec une fonction appelée `save.image` mais il nous semble préférable de privilégier une sauvegarde plus sélective de quelques objets (les fonctions le permettant seront présentées à la section 9.1.1). Les fichiers produits sont d'extension .Rdata, ou .RDS pour des objets individuels plutôt que des groupes d'objets.

1.2.4 Historique de commandes

Un script ne reflète que la version finale d'un travail. Les essais et erreurs sont généralement supprimés de ce que l'on sauvegarde, ou mis en commentaires. Il est possible de sauvegarder l'intégralité ou les *n* dernières instructions qu'on a exécutées sous forme de fichiers d'historique (extension .Rhistory).

Nous vous conseillons plutôt de garder des traces de vos essais sous forme de commentaires dans les scripts, ce qui assure là encore une sauvegarde sélective de votre travail, et non pas une masse non structurée dans laquelle il faudra ensuite fouiller pour extraire quelque chose de pertinent.

1.2.5 Gestion de la mémoire

Comme indiqué à la section 1.3.2, les objets R sont tous stockés par défaut en mémoire vive. Il peut arriver, au fil de votre emploi de R, que la mémoire disponible vienne à manquer. Des messages d'erreur « cannot allocate … » apparaîtront alors. Une des solutions les plus simples, à défaut de fonctionner systématiquement, est alors de supprimer des objets avec la fonction `rm`.

Il est préférable de commencer par créer les objets les plus volumineux, par exemple quand on restaure des sauvegardes. R demande en effet des zones de mémoire contiguës, et au fil de la création de petits objets, l'espace disponible se fractionne. Dès lors les « gros » blocs de mémoire vive disponible deviennent plus rares et la création de nouveaux gros objets n'est pas toujours possible.

1. 3 INTERFACES POUR LA PROGRAMMATION R

On l'a vu dans la section 1.2.3, le langage R peut être exécuté par plusieurs compilateurs. Mais on trouve également plusieurs logiciels pour servir d'interface avec ce compilateur, et particulièrement pour rédiger, exécuter et faire vivre son code. Les pages qui suivent en décrivent deux : l'interface R « basique » correspondant au logiciel **RGui** distribué par CRAN et l'interface **RStudio**, proposée par la société du même nom. RStudio est également gratuit et s'avère largement plus convivial que l'interface de base.

1.3.1 Interface R basique

Le logiciel RGui est le compagnon de la version de R téléchargée de CRAN.

Il s'agit d'une interface extrêmement basique. On y trouve une fenêtre principale, la **Console**, où s'affichent le rappel du code exécuté et les réactions de R : messages d'erreurs mais aussi résultats. Seuls les graphiques s'affichent dans une fenêtre séparée, comme on le voit sur la figure 1.2. On peut coder directement dans la fenêtre Console au niveau de l'invite >, comme dans une fenêtre système DOS ou shell. Mais il est large-

ment préférable de rédiger des scripts dans des fenêtres séparées. L'éditeur des scripts est un simple bloc-notes sans couleurs, sans indentations automatiques, sans souffleur de syntaxe… On verra que la comparaison avec l'éditeur de code de RStudio est cruelle pour RGui.

L'autre handicap de RGui concerne la visualisation des objets en mémoire dans l'environnement courant. Il n'y a pas de fenêtre prévue à ce sujet et on doit les afficher dans la fenêtre Console avec la fonction `ls()`. De même il faut coder pour savoir quels packages sont installés, lesquels sont chargés, etc.

Enfin, l'aide de R s'ouvrira dans une fenêtre séparée, dans un navigateur Internet (il s'agit cependant de la version locale de la documentation, il n'est pas nécessaire d'être connecté à Internet pour la consulter).

Figure 1.1 – Démarrage de RGui dans sa version 3.4.1 64 bits sous Windows 7