SYNTHESE DES EXPOSES DE R

Yatoute MINTOAMA - Richard GOZAN - Isabelle Danielle MOSSE

2023-05-07

Table of contents

# 1. Introduction

Nous nous proposons dans ce document de faire la synthèse des exposés que nous avons été amenés à faire dans le cadre de notre de cours de “Projet statistique sur R et Pyhton”, dispensé par M. Aboubacar HEMA, Research Analyst à IFPRI. Les exposés portent sur les thèmes suivants qui sont abordés dans les différents chapitres :

* Package Janitor
* Package GtSummary
* Rmarkdown
* RQuarto
* R vers Excel
* Text Mining avec R
* Système d’équation
* Calcul parallèle
* Python sur R (package Reticulate)
* Cartographie avec R
* RShiny

# 2. JANITOR  « homme à tout faire »

Le package Janitor est un ensemble de fonctions qui permettent d’améliorer la manipulation et le formatage des données dans les data.frames. Certaines des principales fonctions du package Janitor sont:

* **Intérêt** : le principal intérêt du package JANITOR est de permettre d’effectuer des opérations sur des tableaux tels que des tableaux de contingences.
* **Fonctions** : il a pour principales fonctions :
  + **Le nettoyage des données** : pour ce faire elle utilise les fonctions telles que : *clean\_names, compare\_df\_cols, make\_clean\_names, remove\_empty, remove\_constant, get\_dupes, etc.*
  + **L’exploration des données** : cela consiste à parcourir la base de données pour voir ce qui est anormale et le supprimer. Cela nécessite notamment quelques fonctions telles que *: tabyl, adorn\_totals, adorn\_percentages, adorn\_pct\_formatting, adorn\_ns, adorn\_title.*
* **Limites** : JANETOR ne peut pas faire des tableaux et doit faire appel à d’autre packages. De plus, il ne peut pas combiner trois variables.

# 3. GT\_Summary

* **Intérêt** : GT\_Summary a été développer dans l’objectif de faire d’évaluation d’impact. Il s’agit d’un package qui permet de créer et manipuler des tableaux statistiques et de faire des tris à plat.
* **Fonctions** : gtsummary permet notamment de :
  + Générer des tables de statistique récapitulative simples (en utilisant la fonction *tbl\_summary()* ), sortir des tableaux croisés ( tbl\_cross)
  + Combiner plusieurs variables ensemble (à l’aide des fonctions telles que *tbl\_stack(), tbl\_merge()* )
  + Produire des statistiques (telles que la moyenne, médiane l’écart-type) avec par exemple la fonction *tbl\_continuous().*
  + Résumé des résultats de régression (en utilisant la fonction *tbl\_regression()* )
  + Le package gtsummary peut aussi être combiner à Rmarkdown pour générer des rapports sous différents formats tel que html, word, pdf, rtf.
  + Changer les labels dans un tableau, et faire une mise en forme avec la fonction topack.

# 4. R Markdown

R Markdown est un package intégrer à l’application Rstudio permettant de rédiger un document tel que des rapports, des articles, des mémoires, etc. à partir d’un code R et enregistré avec l’extension .Rmd. Le document RMarkdown sera généré sous les formats HTML, PDF ou Word. On utilise notamment les packages tels que : *knitr*, *dplyr*, *kabelExtra.*

Un fichier R Markdown a principalement deux parties :

* L’entête comportant le titre du document écrit, le nom de l’auteur et la date de publication du document.
* Le corps qui est composé de deux parties : une partie où l’on saisit du texte et une autre (chunck) pour saisir du code. On a la possibilité de ne pas afficher de code dans le document final ainsi que les messages d’erreurs et d’avertissement.

Les documents R Markdown : peuvent subir des mises en forme (mettre un texte en gras, italique, souligné, etc.) et on a la possibilité d’insérer différent niveau de titre, d’insérer des formules mathématiques (en utilisant un code latex encadre par des $), insérer des codes dans la partie texte, d’intégrer des tableaux sur la partie texte de RMarckdown, de faire des mises en forme sur les différents tableaux, insérer des images (en pointant l’image par !). Avec R Markdown on a aussi la possibilité d’utiliser des modèles ou templates pour la création d’un document Rmd.

R Markdown offre une multitude d’autres possibilités à explorer.Rmarkdown

# 5. R vers Excel

Pour manipuler des données entre R et Excel, plusieurs packages et fonctions sont utilisés. Ces packages et fonctions permettent d’importer des données depuis Excel vers R, d’exporter des données depuis R vers Excel, et de manipuler des fichiers Excel en effectuant des opérations telles que l’écriture de données, l’ajout d’en-têtes, de paragraphes, de tableaux et de graphiques dans les feuilles de calcul Excel.

## 5.1 Importation des données depuis Excel vers R :

Package : readxl

Fonction : read\_excel()

Exemple :

data <- read\_excel("chemin/vers/votre/fichier.xlsx", sheet = "nom\_de\_la\_feuille")

## 5.2 Exportation des données depuis R vers Excel :

Package : writexl

Fonction : write\_xlsx()

Exemple :

write\_xlsx(data, path = "chemin/vers/votre/fichier.xlsx")

Ce package fournit aussi la fonction **writeData()** permettant d’écrire des données dans un fichier Excel;

## 5.3 Manipulation avancée de fichiers Excel :

Packages :

**openxlsx, xlsx et r2excel:** ils permettent de lire, d’écrire et de manipuler des fichiers Excel (.xlsx)

Principales onctions : Il existe de nombreuses fonctions dont quelques unes sont :

**createWorkbook()** : Crée un nouveau fichier Excel.

**addWorksheet()** : Ajoute une nouvelle feuille de calcul à un fichier Excel existant.

**add\_sheet()** : Ajoute une nouvelle feuille de calcul à un fichier Excel existant.

**add\_table()** : Ajoute un tableau formaté à une feuille de calcul Excel existante.

**xlsx.addHeader()** : Ajoute un en-tête à une feuille de calcul Excel.

**xlsx.addParagraph()** : Ajoute un paragraphe à une feuille de calcul Excel.

**xlsx.addTable()** : Ajoute un tableau à une feuille de calcul Excel.

**xlsx.addPlot()** : Ajoute un graphique à une feuille de calcul Excel.

**xlsx::saveWorkbook()** : Enregistre le fichier Excel.

**xlsx.openFile()** : Ouvre le fichier Excel

# 6. Textming

Le textmining est une méthode permettant d’analyser automatiquement des textes pour en extraire des informations utiles. Les packages les plus importants sont : **tidytext**, **dplyr**, **tm**, *ggplot2*, **wordcloud**, **ggraph**, **rwhatsapp**, **lubridate**.

Pour se faire, plusieurs étapes sont nécessaires.

## 6.1 Cas d’une base de données

* Prétraitement du texte :

Les packages **tidytext**, **dplyr** et **tm** sont utilisés pour effectuer le prétraitement du texte.

Dans le package **tidytext** les fonctions *unnest\_tokens* et *stop\_words* sont utilisées respectivement pour diviser le texte en tokens et supprimer les mots vides.

Différentes fonctions de manipulation de données du package dplyr sont utilisés (“filter”, “count”, “mutate”, etc.).

Le package *tm* offre une gamme de fonctions **tm\_map** qui permettent d’appliquer des transformations sur des textes. Elle prend en entrée un texte et une ou plusieurs fonctions de transformation, telles que la suppression de la ponctuation(**removePunctuation**), des nombres(**removeNumbers**), des espaces(**stripWhitespace**), etc…

* Fréquence des mots ou des n-grammes : Le package *ggplot2* est utillisé pour afficher la fréquence des mots ou des n-grammes. On peut égalent créer une matrice terme-document en utilisant la fonction *DocumentTermMatrix* du package **tm**.
* Nuage de mots : Le package **wordcloud** est utilisé pour créer un nuage de mots représentant les mots les plus fréquents dans le texte.
* Réseau des mots : Le package **ggraph** est utlisé pour créer un réseau des mots, permettant de visualiser les relations entre les mots les plus fréquents.

## 6.2 Cas des messages WhatsApp, Facebook, Twitter ou un livre

* Chronologie des messages : A l’aide du package **rwhatsapp**, les messages dans des bases de données distinctes. Ensuite les packages **lubridate** et *ggplot2* sont utlisés pour visualiser la chronologie des messages.

Les fonctions de manipulation de données du package *dplyr* sont utilisés pour réaliser les différentes étapes suivantes :

* Classement des ID selon le nombre de messages envoyés
* Mots les plus fréquents utilisés par chaque ID
* Comparaison des mots les plus fréquents entre les groupes

# 7. Calcul Parallèle

## 7.1 Introduction

La plupart des traitements que nous effectuons suivent le principe du calcul séquentiel, où une tâche doit être exécutée avant que les autres ne soient déclenchées. Cependant, lorsqu’il s’agit de grandes bases de données, nous remarquons facilement la lenteur de l’ordinateur à exécuter les tâches demandées. Pour remédier à ces problèmes et accélérer les calculs, il est essentiel d’utiliser le calcul en parallèle.

## 7.2 Principes généraux

1. Découper le calcul informatique en blocs de calcul indépendants.
2. Exécuter simultanément (en parallèle) les blocs de calcul sur plusieurs unités de calcul.
3. Rassembler les résultats et les renvoyer.

## 7.3 Étapes du calcul en parallèle

1. Démarrer m processus “travailleurs” (c’est-à-dire les cœurs de calcul) et les initialiser.
2. Envoyer les fonctions et les données nécessaires à chaque tâche aux travailleurs.
3. Diviser les tâches en m opérations de taille similaire et les envoyer aux travailleurs.
4. Attendre que tous les travailleurs aient terminé leurs calculs et obtenu leurs résultats.
5. Rassembler les résultats des différents travailleurs.
6. Arrêter les processus des travailleurs.

## 7.4 Packages utilisés

* Package Parallel : inclus dans la distribution de base de R, il utilise des fonctions de la famille des apply.
* Package doParallel et Foreach.
* Package rmr2 (MapReduce).
* Et d’autres.

## 7.5 Quelques fonctions utiles :

* Detectcores() : permet de détecter le nombre de cœurs de la machine.
* Makecluster() : utilisé pour créer un cluster de travailleurs.
* Stopcluster() : utilisé pour arrêter et libérer les différents travailleurs.
* Les fonctions de la famille des apply, adaptées au calcul parallèle sous R, permettent d’exécuter simultanément des opérations sur différents blocs.
* parApplay() : effectue des calculs en parallèle sur une matrice ou un tableau en utilisant un cluster de travailleurs.
* parLapply() : applique une fonction à chaque élément d’une liste

# 8. Résolution de système d’équations

Il existe différentes approches pour résoudre des systèmes d’équations non linéaires avec R

## 8.1 Méthodes directes

Il y a principalement 3 packages

### 8.1.1 Le package RootSolve

Ce package fournit la fonction pour résoudre des équations non linéaires multidimensionnelles, utilisant des méthodes itératives basées sur la méthode de Newton pour trouver les solutions.

**Fonctionnalité** : Elle prend en entrée un vecteur de fonction d’équations non linéaires à résoudre, un vecteur initial et retourne un objet(liste) contenant les éléments **root**(la solution trouvée), **f.root**(valeurs des fonctions évaluées aux points de la solution), **iter**(nombre d’itérations effectuées) et **estim.precis**(estimation de la précision atteinte).

Notons l’utilisation du symbole “$” pour accéder à ces difféents élements

### 8.1.2 Le package nleqslv

Ce package permet la résolution numérique de systèmes d’équations non linéaires multivariés. Il fournit principalement la fonction **nleqslv()** utilisant des méthodes itératives basées sur la méthode de Newton et de broyden(méthode par défaut).

**Fonctionnalité:** Elle prend en entrée un vecteur de fonction d’équations non linéaires à résoudre, un vecteur initial, la méthode et retourne une liste contenant les éléments : **x**(solution trouvée), **fvec**(valeurs des fonctions évaluées aux points de la solution), **iter**(nombre d’itérations effectuées) et **d’autres élements**.

Notons l’utilisation du symbole “$” pour accéder à ces difféents élements

### 8.1.3 Le package pracma

Il propose aussi des méthodes numériques pour résoudre des équations non linéaires. Par exemple, les fonctions **fsolve()** et **broyden()** en utilisant la méthode de Newton ou la méthode de Broyden,

## 8.2 Méthodes indirectes

On peut résoudre un système d’équations non linéaires en le transformant en un problème d’optimisation. Les fonctions **optim()** dans le **package stats** et **optimx()** dans le **package optimx** sont utilisées pour résoudre le problème d’optimisation.

**Fonctionnalité de la fonction optim():** Elle prend un entrée Un vecteur de valeurs initiales, la fonction objective à optimiser, la méthode d’optimisation à utiliser et renvoie une liste contenant les éléments : par(la solution optimale), value(la valeur optimale de la fonction objective obtenue), iterations(le nombre d’itérations) et d’autres éléments. Notons l’utilisation du symbole “$” pour accéder à ces difféents élements.

# 9. Python dans R: le package Reticulate

## 9.1 Introduction

R et Python sont deux langages de programmation populaires, chacun étant conçu pour répondre à des besoins spécifiques des utilisateurs. R est réputé pour ses capacités avancées en analyse statistique, manipulation et visualisation des données, tandis que Python est davantage orienté vers la data science, le développement d’applications web et l’automatisation/scripting. Face à cette diversité de fonctionnalités, il est souvent difficile de choisir entre les deux langages. Une approche idéale consiste à combiner les deux langages afin de bénéficier de leurs points forts respectifs.C’est dans cette optique que le package Reticulate pour R a été développé.

## 9.2 Etapes de configuration d’un environnement python sur R

* Phase 1 : Installer le package Reticulate sur R.
* Phase 2 : Installer python sur votre machine si cela n’est pas encore fait.
* Phase 3 : Configurer l’environnement python : On peut le faire de trois manières :
  1. Utiliser l’environnement python de votre système :

reticulate::use\_python("emplacement/de/python/dans/votre/machine")

* 1. Utiliser un environnement:

virtualenv\_create("mon/virtual/env")

* 1. Utiliser un environnement Conda

reticulate::use\_condaenv("nom/de/environnement/conda")

## 9.3 Comment utiliser python sur R ?

Après avoir configurer notre environnement python sur R, on peut utiliser python sur R de différentes manières :

* reticulate::py\_run\_file () : exécuter un scripte python sur R
* reticulate::py\_install() & reticulate::import() : Installer et importer des modules python
* reticulate::source\_python() : référérencement d’un script python afin d’utiliser les fonctions qui y sont
* Ecrire exécuter un code python sur R-markdown dans un chunk python
* reticulate :: py\_run\_string() : exécuter du code python dans un script R avec la fonction
* reticulate::r\_to\_py() & reticulate::py\_to\_r() : convertir un objet R en python & vice versa

# 10. Rshiny

## 10.1 Introduction

Lorsque nous travaillons avec des variables dans une base de données, nous sommes souvent confrontés à la réalisation de statistiques. Souvent, nous devons écrire du code pour effectuer différentes actions sur nos variables, telles que le tri à plat, les croisements de variables ou la création de modèles en fonction de variables spécifiques. Cette tâche peut devenir fastidieuse lorsque nous devons répéter les mêmes actions pour plusieurs variables. De plus, il peut arriver que nous souhaitions simplement suivre la distribution d’une variable par rapport à une autre sans que cela apparaisse dans notre rapport ou document final. Pour remédier à cela, R Shiny nous offre la possibilité de créer des applications web interactives qui facilitent les analyses. Nous pouvons ainsi programmer nos variables et nos différents modèles, puis les choisir à la volée au lieu d’écrire du code à chaque fois.

## 10.2 Shiny : qu’est ce qu’un ça fait ?

Shiny est un package R développé par RStudio qui permet de créer des applications web interactives. Avec Shiny, il est possible de construire des interfaces utilisateur dynamiques sans avoir besoin de connaissances approfondies en HTML, CSS ou JavaScript. Shiny suit un modèle ui-serveur, où le serveur exécute le code R pour générer les sorties dynamiques, tandis que l’interface utilisateur (navigateur web) affiche l’interface et permet aux utilisateurs d’interagir avec l’application. Shiny facilite la création d’applications web pour l’analyse de données, l’affichage de graphiques, l’exécution de modèles statistiques, etc., le tout directement depuis R.

## 10.3 Les différentes parties d’une application Shiny

Une application Shiny se compose essentiellement de deux parties :

* **L’ui (interface utilisateur)** : cette partie comprend tout ce que nous voyons sur l’application. Elle gère l’aspect visuel et ce qui nous captive lorsque nous utilisons l’application. Si nous devions comparer une application Shiny à un être humain, cette partie serait son apparence physique.
* **Le serveur** : c’est le cerveau qui gère tous les calculs en arrière-plan et nous renvoie les résultats que nous voyons sur l’application.