Ecole nationale de la Statistique et de l’Analyse économique Pierre Ndiaye

Synthèse des exposés de R

Jean Pierre A. NDIAYE - Stewart NFEGUE - P. TIENDREBEOGO - Crépin MEDEHOUIN

Juillet 2023

## *Introduction*

R est un logiciel de traitement statistique très outillé. Nous avons eu l’opportunité de le découvrir grace aux exposés portant sur les thèmes: le package Janitor, le package gtsummary, RmarkDown, R Quarto, R vers Excel,Textmining, Résolution des systèmes d’équations non linéaires avec R,Cartographie avec R,R avec Python,Calculs parallèles, R shiny. Qui parle de statistien parle d’un spécialiste en synthèse d’information. C’est l’objet de ce document qui résume les 11 exposés.

## 1. *Le package Janitor*

Avez-vous envie de nettoyer vos données et améliorer vos tableaux de contingence(exploration)? Janitor est à votre disposition. Il suit les principes de “l’inverse” et fonctionne bien avec la fonction pipe %>%.

### 1.1 Installation et Importation du package

#install.packages(janitor)  
library(janitor)

### 1.2 Fonctions principales de Janitor

#### 1.2.1 La fonction clean\_names

Elle est incontournable.

# Create a data.frame with dirty names  
  
test\_df <- as.data.frame(matrix(ncol = 6))  
names(test\_df) <- c("firstName", "ábc@!\*", "% successful (2009)",  
 "REPEAT VALUE", "REPEAT VALUE", "")  
test\_df

## firstName ábc@!\* % successful (2009) REPEAT VALUE REPEAT VALUE   
## 1 NA NA NA NA NA NA

#### 1.2.2 La fonction get\_dupes()

Cette fonction permet de rechercher et d’examiner les enregistrements en double lors du nettoyage des données.get\_dupes() renvoie les enregistrements (et insère un nombre de doublons) afin que vous puissiez examiner les cas problématiques :

get\_dupes(mtcars, wt, cyl) # or mtcars %>% get\_dupes(wt, cyl) if you prefer to pipe

## wt cyl dupe\_count mpg disp hp drat qsec vs am gear carb  
## 1 3.44 6 2 19.2 167.6 123 3.92 18.30 1 0 4 4  
## 2 3.44 6 2 17.8 167.6 123 3.92 18.90 1 0 4 4  
## 3 3.57 8 2 14.3 360.0 245 3.21 15.84 0 0 3 4  
## 4 3.57 8 2 15.0 301.0 335 3.54 14.60 0 1 5 8

#### 1.2.3 La fonction remove\_empty()

Elle permet le nettoyage de fichiers Excel qui contiennent des lignes et des colonnes vides après avoir été lus dans R.

q <- data.frame(v1 = c(1, NA, 3),  
 v2 = c(NA, NA, NA),  
 v3 = c("a", NA, "b"))  
q %>%remove\_empty(c("rows", "cols"))

## v1 v3  
## 1 1 a  
## 3 3 b

#### 1.2.4 La fonction tabyl, une meilleure version de la fonction table()

tabyl() est un remplacement de table() et est accompagnée d’une série de fonctions adorn\_\*:

mtcars %>%  
 tabyl(gear, cyl) %>%  
 adorn\_totals("col") %>%  
 adorn\_percentages("row") %>%  
 adorn\_pct\_formatting(digits = 2) %>%  
 adorn\_ns() %>%  
 adorn\_title(placement = "combined")

## gear/cyl 4 6 8 Total  
## 3 6.67% (1) 13.33% (2) 80.00% (12) 100.00% (15)  
## 4 66.67% (8) 33.33% (4) 0.00% (0) 100.00% (12)  
## 5 40.00% (2) 20.00% (1) 40.00% (2) 100.00% (5)

### 1.3 Les limites de Janitor

Le package Janitor ne permet pas d’obtenir un tableau synthétique lorsqu’on veut croiser plus de deux variables. Pour ce faire, nous avons besoin du package gtsummary.

## 2. *Le package gtSummary*

Le package gtsummary est un outil qui nous permet de créer des tableaux analytiques avec des statistiques (moyenne, écart-type, médiane etc) qui résument au mieux l’ensemble des données, faire le resumé des modèles de regression.Pour installer le package gtsummary, on procède comme suit :

### 2.1 Une installation pas si simple

Nous allons voir certaines fonctionnalités du package gtsummary

* Création de tables simples et croisées (utilisé avec dplyr)
* Tableau de résumé simples sur les variables en montrant les paramètres statistiques(tbl\_summary et tbl\_continuous)
* Tables croisées avec la fonction tbl\_cross des résumés graphiques

Si nous sommes dans une situation où il faut comparer 2 ou plusieurs tables à la fois, gtsummary nous facilite la vie avec les fonctions ci-dessous :

* tbl\_stack() : Elle permet de coller deux ou plusieurs tables l’un au-dessus de l’autre.
* tbl\_merge() : Elle permet de placer les tables côte-à-côte en s’assurant qu’une même variable est affichée sur la même ligne.

Enfin, pour personnaliser nos tableaux, nous avons theme\_gtsummary\_journal et theme\_gtsummary\_compact etc.

### 2.2 La fonction tbl\_summary

library(gtsummary)  
## importation des données  
library(readr)  
BaseUtile\_r <- read\_csv("C:\\Users\\LENOVO T470\\Documents\\EXPOSES\\TP\_R\_Gtsummary\\BaseProjet\_R.csv")  
  
##### Types de variables  
binary <- BaseUtile\_r[c("sexe\_r", "Delai\_r")]  
multicoto <- BaseUtile\_r[c("diplome\_r", "Diff2\_r", "StatutM\_r", "StatutP\_r", "Conseil\_r")]  
continu <- BaseUtile\_r[c("Exppro\_r", "Abs1\_r", "Abs2\_r")]  
  
# Summary  
  
binary %>% tbl\_summary() #table de résumé pour les variables binaires

| **Characteristic** | **N = 8,882**1 |
| --- | --- |
| sexe\_r |  |
| Femme | 4,321 (49%) |
| Homme | 4,561 (51%) |
| Delai\_r |  |
| Dificile | 2,245 (25%) |
| Facile | 6,637 (75%) |
| 1n (%) | |

multicoto %>% tbl\_summary() #table de résumé pour les variables multicotomiques

| **Characteristic** | **N = 8,882**1 |
| --- | --- |
| diplome\_r |  |
| >=BAC+3 | 2,852 (32%) |
| Aucun | 648 (7.3%) |
| BAC | 1,979 (22%) |
| BAC+2 | 2,370 (27%) |
| CAP | 1,033 (12%) |
| Diff2\_r |  |
| Plutot dificile | 2,923 (33%) |
| Plutot facile | 4,717 (53%) |
| Tres dificile | 694 (7.8%) |
| Tres facile | 548 (6.2%) |
| StatutM\_r |  |
| agric | 136 (1.7%) |
| artis-commerc | 342 (4.2%) |
| employe | 3,930 (48%) |
| ing-cadre-prof\_lib | 1,711 (21%) |
| NSP | 397 (4.9%) |
| ouvrir | 1,005 (12%) |
| prof\_interm | 587 (7.2%) |
| Unknown | 774 |
| StatutP\_r |  |
| agric | 288 (3.5%) |
| artis-commerc | 879 (11%) |
| employe | 1,530 (19%) |
| ing-cadre-prof\_lib | 2,465 (30%) |
| NSP | 333 (4.1%) |
| ouvrir | 1,778 (22%) |
| prof\_interm | 870 (11%) |
| Unknown | 739 |
| Conseil\_r |  |
| aucun | 7,194 (81%) |
| plusieurs | 597 (6.7%) |
| un | 1,091 (12%) |
| 1n (%) | |

continu %>% tbl\_summary() #table de résumé pour les variables continue

| **Characteristic** | **N = 8,882**1 |
| --- | --- |
| Exppro\_r |  |
| 0 | 2,322 (32%) |
| 1 | 1,017 (14%) |
| 2 | 1,346 (18%) |
| 3 | 1,086 (15%) |
| 4 | 619 (8.4%) |
| 5 | 280 (3.8%) |
| 6 | 674 (9.2%) |
| Unknown | 1,538 |
| Abs1\_r | 1 (0, 5) |
| Unknown | 139 |
| Abs2\_r | 13 (2, 37) |
| Unknown | 1,372 |
| 1n (%); Median (IQR) | |

### 2.3 Perspectives

D’accord! Nous savons grace aux packages Janitor et gtsummary comment avoir des tableaux statistiques de qualité. Mais comment mettre ces tableaux dans un document Word ou Pdf? R markdown est une partie de R concut pour répondre aux limites de gtsummary en se qui concerne la production des documents sur et des rapports sur R.

## 3. *R Markdown*

### 3.1 Création d’un document R Markdown en RSudio

En RStudio, nous pouvons créer un nouveau document R Markdown (extension .Rmd) par le menu :

« File > New File > R Markdown… »

### 3.2 Entête d’un fichier R Markdown

Un fichier R Markdown débute pratiquement toujours par un entête encadré de deux lignes contenant uniquement trois tirets (—).

Les métadonnées de base sont :

* title : le titre du document,
* author : le nom de l’auteur,
* date : la date de publication du document.

### 3.3 Formats du document final :

* HTML;
* PDF (à la condition d’avoir un compilateur LaTeX[miktex pour les utilisateurs Windows]);
* Word.

### 3.4 Les parties de Rmarkdown

Le fichier R markdown est composé de deux parties:

#### 3.4.1 *partie chunk* reservée aux codes.

Les options de blocs de code R les plus usuelles sont les suivantes :

* eval (TRUE par défaut, ou FALSE) : détermine si le code R doit être évalué ou non;
* echo (TRUE par défaut, ou FALSE) : détermine si le code R doit être affiché ou non;
* warning (TRUE par défaut, ou FALSE) : détermine si les messages d’avertissement doivent être affichés.

#### 3.4.2 *partie texte* (non éxécutable sauf les écritures latex)

Dans la partie texte, la syntaxe(##) en début permet de créer un sous titre et nous y appliquons principalement les techniques suivantes.

* Pour forcer le retour à la ligne, il faut mettre “deux espaces vides à la fin d’une ligne;
* Pour écrire en gras il faut mettre deux étoiles entre le mot souhaité;

### 3.5 Perspectives

Pour quelqun qui ne sait pas coder, il lui faut une alternative pour produire son document. Ne vous inquiétez pas. Quarto est là.

## 4. *R Quarto*

Tout comme R markdown, Quarto permet de produire des documents présentables. Quarto nous facilite la vie dans le sens qu’on n’a pas besoin des codes pour faire des jolies présentations car il offre un interface visuel. Voici comme ca marche.

### 4.1 Insertion des titres

Pour insérer un titre, on clique sur l’onglet Normal et on choisit le niveau du titre désirée (de Header 1 à Header 6).

### 4.2 Style du texte

Comme dans MS Word, il est possible de mettre notre texte en gras, italique, majuscule, miniscule etc. Il suffit juste d’aller sur l’onglet Format.

### 4.3 Mise en couleur

Pour faire la mise en couleur, il faut avoir des notions sur le langage CSS. Pour mettre tout le texte en couleur, on va sur l’onglet Format -> Div puis dans CSS Stle.

### 4.4 Listes

Possibilité de faire des lstes ordonées ou non. L’onglet Format -> Bullets & Numbering permet de le faire.

### 4.5 Insérer une image

On part dans le menu insérer et on choisit Image/Figure. Comme dans word, on peut effectuer des modification sur la taille, la largeur etc. Il y a également la possibilité de mettre 2 images l’une à côté de l’autre. Pour ça, on insère un div et on tape layout-ncol= 2.

### 4.6 Insertion des tableaux

On clique sur le menu Table puis sur Insert table. Il y a également la possibilité de mettre plusieurs tableaux selon nos choix comme avec les images.

### 4.7 Insérer un saut de page

Ceci n’est possible qu’avec les documents word et PDF. On procède par Insert-shotcode et page break;

### 4.8 Insérer une équation

Il faudrait une bonne maitrise de LaTex pour insérer une équation. On part sur l’onglet Insert-LaTex Math.

### 4.9 Insérer les emojis ou symboles

Pour rendre nos présentations plus attractive, on peut mettre des emojis en allant sur Insert-special character-insert emoji.

### 4.10 Insérer des liens hypertextes

Quarto nous donne les moyens d’accéder à une figure, equation etc à l’aide des liens hypertextes. On va dans Insert-Cross reference.

### 4.11 Insérer des notes de bas de pages

Pour cela on clique sur Insert-Footnote.

### 4.12 Insérer des références bibliothèque

Quarto permet d’insérer la source d’une citation dans un document dans notre référence bibligraphique. Une fois que nous avons écrit notre citation, on insère le DOI qui est l’identifiant unique d’un article. Pour le faire, on tape Insert-citation-DOI et on met le DOI du fichier. Non seulement Quarto nous permet de faire de bonnes présentations, il permet également de faire des livres. Le livre permet d’avoir une vue d’ensemble sur la présentation. On doit noter qu’on ne peut pas ouvrir 2 livres en même temps et ça constitue une limite de Quarto.

### 4.13 Perspectives

Nous avons trouvé Quarto, une alternative à R markdown. Toutefois pouvons nous quitter de R vers Excel?

## 5. *R vers Excel*

L’objectif est d’utiliser R pour créer des tableaux et des graphiques, puis exporter les résultats au format Excel, offrant ainsi une alternative à l’utilisation directe d’Excel. Pour ce faire, cela necessite installation des packages a savoir:

### 5.1 Une installation pas si simple

### 5.2 Un exemple simplifié :

En somme, Les packages xlsx, openxlsx, et writexl sont des outils très utiles en R pour travailler avec des fichiers Excel.

### 5.3 Perspectives

Au delà du transfert d’informations de R vers excel, une analyse de la fréquence des mots permettrait de mieux tirer les informations. D’où le Textmining.

## 6. *TEXT MINING*

Il sert à traiter des questions ouvertes des questionnaires d’enquête par exemple pour faire ressortir des connaissances significatives. Afin de faire du text mining sur une base de données, il serait nécessaire de:

* faire un prétraitement de texte,
* faire ressortir la fréquence des mots ou des bigrammes,
* faire la matrice des mots,
* faire un nuage de points,
* enfin faire un reseau des mots.

### 6.1 *Prétraitement du texte*

Pour le prétraitement, on aura besoin des packages ci-dessus :

### *6.2 Fréquence des mots et nuage des mots*

Après le prétraitement, on peut soit s’intéresser à la fréquences des mots et des bigrammes ou au nuage des mots.L’avantage des bigrammes est qu’ils apportent plus d’informations approfondies. Pour avoir le nuage des mots, on utilise le package suivant :

* Le nuage des mots peut-être ajusté aléatoirement avec la fonction Set.seed(). Par exemple set.seed(2314) ne peut pas fixer le nuage de mots de la même manière que set.seed(9675).
* Le nuage de mots change mais la fréquence des mots n’est aucunument impactée. Outre le nuage des mots, on peut aussi faire le nuage des bigrammes. Plus un mot(ou bigramme) sur le nuage de mots(ou bigramme) est gros et gras pus est grande la fréquence.

## 7. *Cartographie avec R*

### 7.1 Concepts de base de la cartographie}}

#### 7.1.1 Système de coordonnées de référence (CRS)}}

Le système de coordonnées de référence (CRS) permet de situer les objets géographiques sur la Terre.Sur R, pour créer un CRS on utilise la fonction *CRS()* du package *sp*. Par exemple :

library(sp)  
crs <- CRS("+proj=utm +datum=WGS84 +ellps=WGS84")

* Le paramètre *+proj* spécifie la projection à utiliser.
* *+datum* définit l’origine, l’orientation et l’échelle du système de coordonnées par rapport à la Terre.
* *+ellps*, quant à lui, spécifie uniquement l’ellipsoïde de référence utilisé pour représenter la forme de la Terre.

Pour attribuer un CRS à un objet spatial existant, on peut utiliser la fonction st\_crs() du package *sf* :

library(sf)  
data <- st\_read("data.shp")  
st\_crs(data) <- crs

#### 7.1.2 Fichiers de données spaciales

Les données spatiales suivent deux modes: le mode raster et le mode vecteur. Un raster est une image(png,jpg,asc,db) alors qu’un vecteur représente les entités géographiques(points, lignes, polygones) et comportent les formats shapefile et json.

### 7.2 Lecture et manipulations des données spaciales sur R

#### 7.2.1 Passage des données non spaciale à des données de la classe “Spaciale”

Commençons par importer des données classiques, c’est-à-dire dans un format non spatial, par exemple auformat texte .csv. Pour importer ces données sous R :

seisme\_df <- read.csv2("C:\\Users\\LENOVO T470\\Documents\\EXPOSES\\Exposé R -Cartographie\\Donnees\\earthquake\\earthquakes.csv")  
head(seisme\_df, 2)

## Year Month YYMM Day Time.hhmmss.mm.UTC Latitude Longitude Magnitude Depth  
## 1 1973 1 197301 1 34609.8 -9.21 150.63 5.3 41  
## 2 1973 1 197301 1 52229.8 -15.01 -173.96 5.0 33

Dans la suite, nous présenterons les deux solutions (*sp* et *sf*) simultatément

##### 7.2.2 La classe sp

On utilise la fonction *coordinates()* et on précise avec le symbole *~* quelle sont les variables de géolocalisation.

library(sp)  
seisme\_sp <- seisme\_df  
coordinates(seisme\_sp) <- ~Longitude + Latitude  
class(seisme\_sp)

## [1] "SpatialPointsDataFrame"  
## attr(,"package")  
## [1] "sp"

##### 7.2.3 La classe sf

On utilise la fonction *st\_as\_sf()* de la manière suivante :

library(sf)  
seisme\_sf <- st\_as\_sf(seisme\_df, coords = c("Longitude", "Latitude"))  
class(seisme\_sf)

## [1] "sf" "data.frame"

### 7.3 Importation des données spatiales

Nous allons voir les deux façons d’importer ces données selon qu’on choisit la classe sp ou bien la classe sf

* *Classe sp*

Nous utilisons la fonction “readOGR()” issu du package “rgdal” qui permet d’importer les types de données spatiales:

library(rgdal)  
world\_sp <- readOGR(dsn = "C:\\Users\\LENOVO T470\\Documents\\EXPOSES\\Exposé R -Cartographie\\Donnees\\World WGS84",  
layer = "Pays\_WGS84")

## OGR data source with driver: ESRI Shapefile   
## Source: "C:\Users\LENOVO T470\Documents\EXPOSES\Exposé R -Cartographie\Donnees\World WGS84", layer: "Pays\_WGS84"  
## with 251 features  
## It has 1 fields

* *Classe sf* Dans ce cas, nous utilisons la fonction st\_read()

world\_sf <- st\_read("C:\\Users\\LENOVO T470\\Documents\\EXPOSES\\Exposé R -Cartographie\\Donnees\\World WGS84\\Pays\_WGS84.shp")

## Reading layer `Pays\_WGS84' from data source   
## `C:\Users\LENOVO T470\Documents\EXPOSES\Exposé R -Cartographie\Donnees\World WGS84\Pays\_WGS84.shp'   
## using driver `ESRI Shapefile'  
## Simple feature collection with 251 features and 1 field  
## Geometry type: MULTIPOLYGON  
## Dimension: XY  
## Bounding box: xmin: -180 ymin: -89.9 xmax: 180 ymax: 83.6236  
## Geodetic CRS: WGS 84

### 7.4 Sélection d’observations

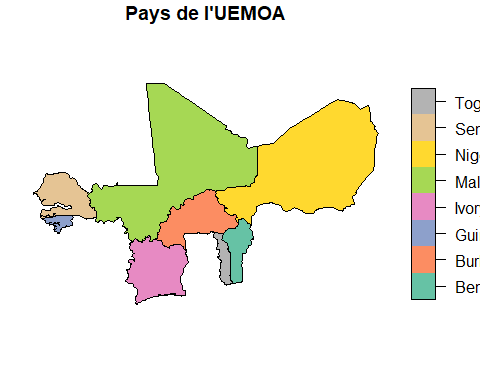
Pour la classe sf, on peut utiliser la même syntaxe que pour sp, mais en plus, on peut utiliser la syntaxe à la mode dplyr.

library(tidyverse)  
uemoa\_sf <- world\_sf %>%  
 filter(NOM %in% c("Benin", "Burkina Faso", "Guinea-Bissau",  
 "Ivory Coast", "Mali",  
 "Niger", "Senegal","Togo"))

### 7.5 Représentations de cartes de base

On utilise la fonction plot() qui appliquée à un objet “Spatial” va seulement représenter la géométrie de l’objet. On peut ensuite utiliser les fonctions graphiques de base (title(), legend(), etc.) pour ornementer le graphique

plot(uemoa\_sf, main = "Pays de l'UEMOA")



### 7.6 Aggrégation des données

Créons un nouveau jeu de données avec les mêmes observations.

uemoa2.df <- data.frame(NOM = c("Benin", "Burkina Faso",  
 "Guinea-Bissau", "Ivory Coast",   
 "Mali", "Niger", "Senegal"),  
 pib = c(14954,16686,1458,59221,16183,12609,23354),  
 pop = c(12996,22100,2060,27478,21904,25252,16876),  
 region = rep("O", 7))  
row.names(uemoa2.df) <- uemoa2.df$NOM

Pour aggéger les données spatiales de type sf, on peut aussi utiliser la syntaxe dplyr. Ici, on aggrége d’abord les entités spatiales et on applique ensuite la fonction merge() :

uemoa\_sf\_new <- uemoa\_sf %>% group\_by(  
 NOM = c("Benin", "Burkina Faso", "Guinea-Bissau",  
 "Ivory Coast", "Mali","Niger", "Senegal", "Benin")) %>%  
 summarise() %>%  
 merge(uemoa2.df, by = "NOM")  
class(uemoa\_sf\_new)

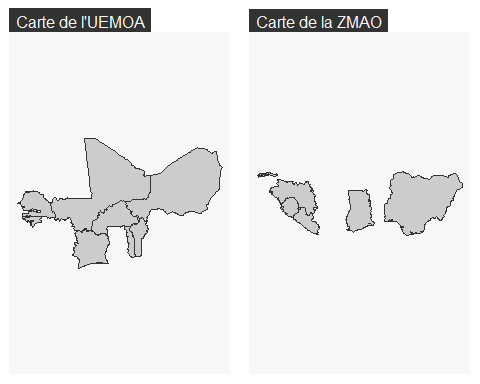
## [1] "sf" "data.frame"

### 7.7 Afficher plusieurs cartes sur la même figure

Il faut ici utiliser l’argument mfrow de la fonction *par()*.

zmoa\_df <- data.frame(NOM = c("Gambia, The", "Ghana",  
 "Guinea", "Liberia", "Nigeria", "Sierra Leone"),  
 pib = c(1719,66882,13606,2965,452971,3505),  
 pop = c(2639,31394,13531,5193,213401,8141),   
 region = rep("O", 6))

library(mapsf)  
par(mfrow = c(1, 2))  
# first map  
mf\_map(uemoa\_sf)  
mf\_title("Carte de l'UEMOA")  
# second map  
zmoa\_sf <- world\_sf %>% filter(  
 NOM %in% c("Gambia, The", "Ghana", "Guinea",  
 "Liberia", "Nigeria", "Sierra Leone"))  
mf\_map(zmoa\_sf)  
mf\_title("Carte de la ZMAO")



## 8. *Résolution de système d’équations non linéaire avec R*

Pour résoudre un système d’équations non linéaires, R dispose d’un ensemble de package et fonctions.

### 8.1 Le package rootSolve

Le package “rootSolve” fournit des outils pour la résolution numérique de systèmes d’équations non linéaires.

#### 8.1.1 rootSolve::multiroot()

Cette fonction est utilisée pour résoudre des systèmes d’équations non linéaires multivariées et l’algorithme est implémenté avec la méthode de Newton-Raphson.

# install.packages("rootSolve")  
library(rootSolve)  
# Définition du système d'équations  
model <- function(x){ c(F1 = x[1]^2+ x[2]^2 -1,  
F2 = x[1]^2- x[2]^2 +0.5)  
}  
solution <- multiroot(f = model, start = c(1, 1))  
# Affichage des solutions  
print(solution$root)

## [1] 0.5000000 0.8660254

### 8.2 Le package nleqslv

Le package “nleqslv” offre des fonctionnalités avancées en utilisant des méthodes itératives basées sur la méthode de Newton et de broyden(méthode par défaut).

#### 8.2.1 nleqslv::nleqslv()

Cette fonction prend en entrée une fonction d’équation multivariée, un vecteur initial de valeurs approchées pour les variables, et retourne une approximation des valeurs des variables qui satisfont le système d’équations.

### 8.3 Le package pracma

Le package “pracma” aussi propose les fonctions fsolve() et broyden() pour la résolution.

### 8.4 Les méthodes indirectes

Elles consistent à résoudre un système d’équations non linéaires à l’aide d’un problème d’optimisation. Sous R,les fonctions comme optim () dans le package stats , optimx() dans le package optimx font l’affaire.

## 9. *Utilisation de Python dans R :Package Reticulate*

Cet exposé portait sur l’utilisation de Python dans R, c’est-à-dire comment exécuter du code Python dans R. Pour cela, il est nécessaire d’installer un package appelé Reticulate. Reticulate est un package R qui permet une intégration étroite entre R et Python. Alors comment installer Reticulate? Comme les autres packages, nous l’installons à travers la syntaxe suivante :

### 9.1 Installation et importation

Avant d’utiliser le package reticulate de R, il est important de s’assurer d’avoir Python installé sur votre machine, avec une version égale ou supérieure à 2.7. Possible de vérifier la disponibilité de Python à partir de R en utilisant la fonction py\_available() du package reticulate.

La deuxième étape est de créer un environnement de travail

Nous pouvons mainteneant travailler avec des codes Python

### 9.2 Installation des packages Python

reticulate::py\_install("pandas", "virtual\_py")

## Using virtual environment "virtual\_py" ...

## + "C:/Users/LENOVO T470/Documents/.virtualenvs/virtual\_py/Scripts/python.exe" -m pip install --upgrade --no-user "pandas"

### 9.3 Importer des données

Ensuite nous pouvons effectuer toutes manipulations avec des codes Python.

### 9.4 Recommendations

* Demande une maîtrise de la syntaxe de Python et de la syntaxe R;
* Dépendance aux packages Python: vous devez toujours installer les packages avant de les importer

## 10. *Calculs parallèles*

Le package parallel de R est une bibliothèque standard intégrée à R qui offre des fonctionnalités pour l’exécution parallèle de code. Cette dernière consiste à diviser une tâche en sous-tâches qui peuvent être exécutées simultanément sur plusieurs processeurs ou cœurs de calcul.

### 10.1 Les fonctions importantes de parallel sont :

* *detectCores()*: Cette fonction est utilisée pour détecter le nombre de cœurs de calcul disponibles sur votre système.
* *makeCluster()* utilisée pour créer un cluster de calcul parallèle. Un cluster est un groupe de nœuds ou de processus R qui peuvent exécuter des tâches en parallèle.
* *stopCluster()* utilisée pour arrêter et libérer les ressources d’un cluster de calcul parallèle créé avec makeCluster().
* *parApply()* qui divise le tableau en morceaux et distribue ces morceaux sur les cœurs disponibles pour être traités en parallèle.
* *parLapply()*: qui divise la liste ou le vecteur en sous-ensembles et les distribue sur les cœurs disponibles pour être traités en parallèle.
* *parSapply()* similaire à parLapply(), mais qui retourne les résultats sous la forme d’un vecteur simplifié plutôt que d’une liste.

On a également les packages **doParallel** et **foreach** en R, qui sont souvent utilisés en conjonction pour faciliter l’exécution parallèle de code.

### 10.2 Les fonctions importantes de doParallel et foreach

* *foreach()*: est une fonction du package foreach qui permet d’itérer sur des objets de manière parallèle ou séquentielle.
* *%dopar%* : est utilisé pour différencier les boucles parallèles des boucles séquentielles.
* *registerDoParallel()*: Cette fonction du package doParallel permet d’enregistrer un backend parallèle responsable de la gestion des cœurs ou des nœuds utilisés pour l’exécution parallèle.

Alors pour commencer il est important de nettoyer la machine :

rm(list=ls())

* *gc()* : permet de nettoyer cette mémoire occupée par les objets inutilisés afin d’améliorer les performances du programme

gc()

## used (Mb) gc trigger (Mb) max used (Mb)  
## Ncells 3137583 167.6 5552288 296.6 5240025 279.9  
## Vcells 6575312 50.2 12255594 93.6 9794195 74.8

### 10.3 Importation d’une base de données très volumineuse en utilisant le package parallel

library(parallel)  
  
# Définir le nombre de cœurs  
num\_cores <- detectCores()  
  
# Créer un cluster parallèle  
cl <- makeCluster(num\_cores)  
  
# Liste des noms de fichiers à importer  
file\_names <- c("data\_part1.csv", "data\_part2.csv", "data\_part3.csv")  
  
# Importer les fichiers en parallèle  
import\_data <- parLapply(cl, file\_names, function(file) {  
 read.csv(file) # Importer chaque fichier CSV  
})  
  
# Arrêter le cluster parallèle  
stopCluster(cl)  
  
# Combiner les résultats  
combined\_data <- do.call(rbind, import\_data) # Fusionner les morceaux de données  
  
# Utiliser les données importées

### 10.4 Calcul de moyenne d’un vecteur

En somme , avec parallel, nous pouvons diviser des tâches en sous-tâches et les exécuter en parallèle sur plusieurs cœurs ou nœuds de calcul, ce qui permet de réduire le temps d’exécution.

## 11. *R shiny*

### 11.1 Présentation shiny

Shiny est un package R développé par RStudio qui permet de créer des applications web interactives. Shiny utilise un modèle ui-server : le serveur R exécute le code R et génère les sorties dynamiques, tandis que l’ui (navigateur web) affiche l’interface utilisateur.

### 11.2 Interface utilisateur

L’interface utilisateur contient tout ce qui est visible par l’utilisateur et on utilise plusieurs fonctions:

* **fluidPage** : Crée une page Shiny fluide qui s’adapte à la taille de la fenêtre du navigateur.
* **titlePanel** : Affiche un titre en haut de la page Shiny.
* **selectInput**: Crée un menu déroulant permettant à l’utilisateur de sélectionner une option parmi une liste.
* **sliderInput** : Crée un curseur coulissant permettant à l’utilisateur de sélectionner une valeur numérique dans une plage définie.
* **verbatimTextOutput** : Affiche du texte généré par le serveur dans l’interface utilisateur sans modification.
* **textInput**: Permet de créer un widget de saisie de texte dans une application Shiny.

### 11.3 Server

Dans la partie server, on programme comment les éléments de l’ui doivent s’afficher et les interactions entre eux.

Il y a plusieurs fonctions qu’on utilise dans le serveur.

* **renderPlot**: permet de renvoyer une représentation graphique
* **renderTable**: permet de renvoyer un tableau dans l’interface
* **renderText**: permet de renvoyer un texte dans l’interface

## *Conclusion*

Les 11 exposés confirment que le logiciel R est très riche et disposent de tous les outils de la préparation des données à la création de plateformes interactives en passant par les analyses statisques. Il incombe à tout statisticien ou analyste de données de le maitriser car sans cela, il est “un baobab sans racines”.