LES 12 TRAVAUX D'ASTERIX - EXAM R

Allakere Hormo Maxime

01/02/2021

Je me permettrai d'evaluer les packages suivants sous cinq (5) criteres qui sont:

Formalisation : Savoir si le travail a été reduit aux structures formelles de Rmarkdown dans la redaction.

L'importance du package : les auteurs mettent ils en relief l'importance de l'utilisation du package si oui son champs d'application.

La comprehension : le travail est-il assez explicite pour permettre une bonne comprehension.

Niveau de difficulté : presence ou non des notions mathematiques compliquées.

Vulgarisation : les termes utilisés vulgarisent ou pas le travail pour reduire le niveau de difficulté.

UTILISISATION DU PACKAGE "EVIR" EN R

Auteurs : Thuy Aufrere, Nina Zoumanigui, Arnaud Bruel

[Lien github package evir] (https://github.com/T-AUF/PSBX/blob/main/Packages_evd_evid_graphics.Rmd)

1. Introduction

Le choix de ce package avait pour but precis de presenter une notion important en maths et statistiques, celle de la theorie des valeurs extremes [Cours intéressant sur la Statisque des valeurs extrêmes] (http://irs.math.cnrs.fr/2017/pdf/majumdar.pdf) en modélisant les risques extrêmes;

Evir : qu'est ce que c'est? Evir est un package sous *R* utiliser dans la representation des valeurs extremes qui peuvent être divisées dans les groupes suivants; analyser exploratoire des données, maxima de bloc, pics au-dessus d'un seuil (univarié et bivarié), processus ponctuels, distributions $ gev / gpd $.

2.Quelques fonctions sous Evir

Fonctions	Rôles
dgev	Renvoie la distribution des valeurs extrêmes généralisées
dgpd	Distribution de la $ Pareto $ généralisée
emplot	Graphique de la fonction de distribution empirique
findthresh	Permet de trouver le seuil
gev	Permet de trouver le seuil

Fonctions	Rôles
gpd	Permet d'ajuster les valeurs des valeurs extrêmes généralisées
gumbel	Permet d'ajuste la distribution de $ Gumbell $
nidd.annual	Les données de la rivière $ Nidd $
pgev	Donne la valeur de la distribution des valeurs extrêmes généralisées
interprét.gpdbiv	Interprétation des résultats de l'ajustement $ pgd $ bivarié
rgpd	Distribution de la $ Pareto $ généralisée

3.Contexte et application

L'on installe le package avec la syntaxe suivante sous R : install.packages(evir) et le jeu de données utilisés sous ce package est le nid.annual avec 35 observations.

Nous allons juste nous permettre de commenter le bloc de code suivant et non l'executer.

Nous allons donc les mettres sous forme de commentaire.

```
# nidd2<-sort(nidd, decreasing = TRUE)
# nidd2
# seuil=seq(70,300,by =5)
# #seuil<-70:300
# seuil
# taille_seuil=length(seuil)
# taille_nidd2=length(nidd2)
```

```
# X=matrix(0,nrow=length(seuil),ncol=length(nidd2))
# head(X, n=3)
# #View(X)
# for(i in 1:length(seuil))
# {for(j in 1:length(nidd2))
#   {X[i,j]=max(nidd2[j]-seuil[i],0)}
# }
# head(X, n=3)
```

```
# somme=rep(0,length(seuil))
# somme
# Compteur=rep(0,length(seuil))
# Compteur
# e=c()
# e
# for(i in 1:length(seuil))
# {
#   for(j in 1:length(nidd2))
#   {
#       if ( X[i,j]>0 )
#       {
#           somme[i]=somme[i]+X[i,j]
#           Compteur[i]=Compteur[i]+1
#       }
#   }
```

```
#   e[i]=somme[i]/Compteur[i]
# }
# e
#plot(seuil,e,type='l')
```

Bloc de code 1 : le premier bloc de code permet de lire les données du fichier *Nidd* et les classer par ordre décroissant dans un vecteur noté *nidd*. On crée un vecteur *seuil* ordonné par ordre.

Bloc de code 2 : le deuxieme bloc permet de creer une matrice dont le nombre de lignes correspond à la taille du vecteur "*seuil*" et le nombre de colonnes correspond à la taille du vecteur "*nidd*" telle que chaque ligne i correspond aux excès au delà du $seuil[i]$. les termes negatifs sont remplacés par zero.

Bloc de code 3 : permet de calculer pour chaque seuil la moyenne des exces. le graphique nous donne la valeur extreme de notre fonction moyenne des exces.

EVALUATION ET CONCLUSION

Il s'agit d'un travail de qualité qui a été redigé de façon simple pour permettre une bonne comprehension sauf que certaines notions n'ont pas été vulgarisées et donc sa comprehension ne pourrait etre à la portée de tous. Structure de redaction tres formelle.

UTILISATION DU PACKAGE RPART

Auteur : Salah [lien github du trvail] (https://github.com/Salah1920/psbx/blob/main/Package-Rpart.pdf)

Ayant moi meme travailler sur le package Rpart, je me suis permis de choisir le travail fait par un autre camarade pour juger de la qualité de nos deux travaux en étant le plus objectif possible.

1. Fonctions dans Rpart

Fonctions	Rôles
rpart.internal	Fonction interne
rpart.control	Controle pour Rpart fits
prune.rpart	complexité de cout d'un objet
text.rpart	placer du texte sur un tracé
residuals.rpart	residus d'un objet rpart ajusté
xpred.rpart	Renvoie des predictions à validation croisée
meanvar.rpart	graphique de la variance
summary.rpart	resumé d'un objet Rpart
predict.rpart	prediction à partir d'un objet
na.rpart	gere les valeurs manquantes
plot.rpart	Tracé d'un objet Rpart

2.Contexte et application de Rpart

Rpart permet de constuire des modeles de classification ou de regression d'une structure en representant tout cela sous forme d'arbres binaires.

Nous allons inserer quelques blocs de codes que nous commenterons :

```
# set.seed (28062012)
# appindex = sample (1: nrow (spam), floor (nrow (spam) * 2/3), replace = FALSE)
# app = spam [appindex,]
# val = spam [-appindex,]
```

Ce bloc servirait à decouper le jeu de données en deux, un jeu de données qui servirait à l'apprentissage du modele et l'autre servirait à estimer la performance de la prediction faite.

```
# model = rpart (type ~., data = app, method = "class")
# tracé (modèle, uniforme = VRAI, branche = 0,5, marge = 0,1)
# texte (modèle, tout = FALSE, use.n = TRUE)
```

Ce bloc avec la premiere ligne de code qui permet de constuire l'arbre et les autres syntaxes permettent d'obtenir une representation graphique de l'arbre.

EVALUATION ET CONCLUSION

L'accent a été plus mis sur les syntaxes quant à l'utilisation de ce package sous R que sur l'explication du package, ce à quoi il sert par contre le travail est tres intuitif. Aucun degré de complexité ce qui est tout à fait appreciable et à la portée de la comprehension de tous neamoins manque de formalisation et structure moyenne.

UTILISATION DU PACKAGE PLUMBER

Auteurs : Marion, Imen, Olfa Lien github du travail

1.Introduction

Plumber est un package R open source convertisant le code R en API REST. A partir de quelques simples commentaires à ajouter dans le code R, l'on crée des API. Il peut etre utilisé pour integrer un gaphique en R dans un site web.

API par definition est l'acroynyme Application Programming Interface, un moyen d'interagir par programmation avec un composant logiciel ou une ressource distincte. L'API repertorie un ensemble d'operations que les developpeurs utilisent et la description de ces utilisations.

2.Contexte et Application

Les types API

API WEB : une API accessible via HTTP, il est possible de creer une telle API en utilisant Java, NET etc.

API REST : un ensemble de principes auquels doit adherer un developpeur avant de considerer son API Web comme "RESTFUL". les principes sont les suivants : Architecture client-serveur, Mise en cache, Sans etat(non stockage de l'info), Uniformité de l'interface, Systeme en couches, Code sur demande (facultatif).

Nous commenterons une fois de plus quelques blocs de code utiles sous forme de commentaires sans les executer:

```
# #library(plumber)
#
# #* @apiTitle Plumber Example API
#
# #* Echo back the input
# #* @param msg The message to echo
# #* @get /echo
# function(msg = "") {
#     list(msg = paste0("The message is: '", msg, "'"))
# }
```

Bloc de code : plumber avec la methode GET et le chemin Echo : - #* @apiTitle Plumber Example API : indique le nom de l'API - #* Echo back the input : commentaire sur la fonction - #* @get /echo : methode http utilisée d'ou le GET. - function(msg = "") {....} c'est le code en R

```
# #* Plot a histogram
# #* @png
# #* @get /plot
# function() {
#     rand <- rnorm(100)
#     hist(rand)
# }
```

Bloc de code : plumber avec la methode GET et le chemin plot, une fois la fonction executée, le grpahique en est affiché (hsitogramme).

EVALUATION ET CONCLUSION

Le package a été presenté en details et dans son ensemble, l'accent a été mis sur l'importance du package, et chaque syntaxe a été detaillé, le travail bien formalisé.

UTILISATION DU PACKAGE KSCORRECT

Auteurs: Gasmi Chaymae, Ridadarajat Zakaria, Daif Hakim Lien du Github

1.Introduction

Ce package sert à estimer certains parametres inconnus lors d'un test d'ajustement. Il complete logiquement le Ks.test qui sert à tester si un un echantillon suit une loi quelconque donnée.

2.Contexte et Application

Installation des packages :

```
#install.packages(devtools)
#library(devtools)
#install.packages(infer)
```

Quelques Usages des syntaxes

Les syntaxes suivantes sont repris dans le bloc de code qui va suivre tout en bas.

- `dlunif(x, min, max, base = exp(1))` : donne la densité.
- `plunif(q, min, max, base = exp(1))` : donne la distribution de la fonction.
- `qlunif(p, min, max, base = exp(1))` : donne le quantile de la fonction.
- `rlunif(n, min, max, base = exp(1))` : génére des nombres aléatoires

Un exemple d'application Un exemple d'application des fonctions du package (KScorrect) :

```
# plot(2:200, dlunif(2:200, exp(1), exp(20)), type="l", main="Loguniform density")
# plot(log(2:200), dlunif(log(2:200), log(1), log(20)), type="l", main="Loguniform density")
# plot(2:200, plunif(2:200, exp(1), exp(20)), type="l", main="Loguniform cumulative")
# plot(qlunif(ppoints(200), exp(1), exp(20)), type="l", main="Loguniform quantile")
#
#
# hist(rlunif(2000, exp(1), exp(20)), main="random loguniform sample")
# hist(log(rlunif(20000, exp(1), exp(20))), main="random loguniform sample")
# hist(log(rlunif(20000, exp(1), exp(20), base=10), base=10), main="random loguniform sample")
```

Evaluation et Conclusion

L'importance du package a été mis en evidence et ceux qui ont pour mission de realiser des tests souvent comprendront encore mieux l'interet d'utiliser ce package pour estimer des parametres inconnus, la formalisation de ce travail a été bien faite, aucune complexité niveau comprehension pour ceux qui s'interesseraient au sujet en question en revanche manque d'un peu plus d'explication sur l'utilisation du package en question.

UTILISATION DU PACKAGE SP

Auteurs : Jordy, Kabirou Lien du Github

1. Introduction

Il s'agit d'un package qui fournit des classes et methodes pour les données spatiales en d'autres terme un package fournissant des methodes pour les types de données spatiales. Pour la representation des données spatiales sous R defois on fait appel à plusieurs packages et SP permet aussi une interaction entre tous ces packages et donc pas moins de 350 packages d'analyse spatiale utilisent les données spatiales implementées dans SP.

2.Contexte et Application

Tout type de données spaciales a une structure fondamentale avec deux parametres : - une boite de delimitation - un objet de classe CRS pour definir le systeme de reference des coordonnées.

Installation du package et son chargement :

```
# install.packages(sp)
# install.packages(raster)
# library(sp)
# install.packages(raster)
```

Application : Affichage d'un point sur la carte de la france avec des valeurs de longitude/latitude

```
# adm_fr <- getData('GADM', country='FRA', level=1)
# # GADM pour indiquer que je veux des contours administratifs
# # FRA est le code de la France*
# #   1 indique la granularité de l'affichage, ici la région
# plot(adm_fr,lwd=0.5, border="grey", col="#DFFAB1", bg="black");box()
# points(0,44, col="red", pch=16, cex=2) # coordonnées du point à afficher
```

Evaluation et Conclusion

Grace a leur travail, j'ai appris un peu plus sur la representation spaciale des données, sur le chargement des données, la creation d'objets et leur mise en forme graphique, bien explicite et facile à comprendre en revanche quelques exemples sur de la datavisualisation d'une carte avec ce package serait appreciable. Travail de qualité.

UTILISATION DU PACKAGE GGPLOT2

Auteurs : Moi meme et Siva Chanemougam lien du github

1.Introduction

Ggplot2 est un package specialement conçu pour la visualisation de données qui est une partie tres importante de la data, comme un adage le dit "Une image vaut mille mot" et l'on tendance à visualiser nos jeux de données et le travail effectué pour voir la tendance de ce qui en ressort. ce package fournit des belles parcelles sans tracas qui prennent en compte des details infimes comme les legendes et la representation de celles-ci. les tracés peuvent etre interactive et ce qu'il faut retenir de ce package est qu'il fonctionne sous une grammaire denommée "Grammaire des graphiques", l'interet donc de ce dernier est que l'utilisateur n'est pas limité à des graphiques pre-specifiés. le concept grammaire des graphiques a été créer en 2005 par Wilkinson.

2.Contexte et Application

L'utilisation du package est precede de son installation sur R car il est pas installé par defaut, la syntaxe suivante permet l'installation :

```
#install.packages(ggplot2)
#library(ggplot2)
#help(ggplot2)
```

Library (ggplot2) nous permet de charger le package une fois installé et help est la fonction qui nous renvoie toutes les aides disponibles sous ce package.

Nous avons sous ce package des graphiques dit conventionnels utilisant des jeux de données incorporés dans R, parmi ces graphiques on a des nuages de points, des boites à moustaches, des histogrammes etc. Le bloc de code suivant reprend le jeu de données Airquality pour representer ces graphiques cités ci-dessus:

- Nuage de points:

```
# data("airquality")#chargement du jeu de donnÃ©es airquality
# head(airquality)
# plot(Ozone~Wind, data = airquality)#affiche le graphique reliant les deux variables 'vent' et 'ozone'
```

- Boite à moustaches:

```
#Creer une nouvelle variable dans un dataframe
# airquality$Max <- NA
# airquality$zone <- cut(airquality$Wind, breaks = quantile(airquality$Wind))
# levels(airquality$zone) <- c("Est","Nord","Ouest","Sud")
# table(airquality$zone, airquality$Wind) #Nous permet d'avoir une idÃ©e sur la quantitÃ© de vent dans
# boxplot(Wind~zone, data = airquality)#Visualisation du graphique avec les valeurs aberantes
```

- Histogramme :

```
#Histogramme
# hist(airquality$Wind, main="Histogramme Vent", probability = FALSE, xlab = "vent", col = "lightblue")
```

Exemple d'application : Nous allons nous interesser à un des multiples graphiques contenus dans le travail effectué :

Le graphique barres divergentes Il s'agit d'un graphique qui permet de visualiser un jeu de données divisées en categories opposées ou divergentes, l'exemple qui suit (Nous n'allons pas afficher le graphique mais le code si) est la representation d'un certain nombre de marques de vehicules dont le kilometrage moyen a été calculé, et ces vehincules selon classés selon que leurs kilometrages est inferieur ou superieur à la moyenne.

```
# #On crÃ©e une nouvelle colonne denommÃ© car name
# mtcars$`car name` <- rownames(mtcars)
# # calcul des mpg normalisÃ©s (km)
# mtcars$mpg_z <- round((mtcars$mpg - mean(mtcars$mpg))/sd(mtcars$mpg), 2)
# # au dessus/ au dessous de zero
# mtcars$mpg_type <- ifelse(mtcars$mpg_z < 0, "below", "above")
# #Tri
# mtcars <- mtcars[order(mtcars$mpg_z), ]

#Conertissons les valeurs de la nouvelle colonne en facteur poir le traitement et le tracÃ©
# mtcars$`car name` <- factor(mtcars$`car name`, levels = mtcars$`car name`)

# ggplot(mtcars, aes(x=`car name`, y=mpg_z, label=mpg_z))+geom_bar(stat='identity',aes(fill=mpg_type),w
```

Evaluation et Conclusion

De façon objectif, on a essayé au maximum de decrire le package, son utilité et meme expliquer des syntaxes basiques comme comment modifier une legende, modification des axes bref plusieurs parametres qui font parti du package.Nous avons simplifier le travail en utilisant deja des jeux de données compris dans R pour que toute personne qui tombe sur ce travail puisse reprendre les blocs de codes avec les memes jeux de données. Nous avons mis un accent particulier sur la formalisation du travail ecrit, pas de difficulé mathematiques liée a l'absence d'une quelques notion en mathematiques. En revanche nous aurions du explorer aussi des pistes sur les graphiques en 3 dimendions (3D).