Paris School of Business (PSB)



Compréhension et utilisation de 5 packages de R: evir, evd, R.miner, graphics, regtest

MSc Data Management

Projet : $\underline{\mathbf{R}}$

par:

Thuy AUFRERE, Nina ZOUMANIGUI, Arnaud Bruel YANKO

Sous la direction de :

M. Henri LAUDE

Enseignant

Année académique 2020-2022

Table des matières *

	0.7.5	Applications	21
	0.7.3	Applications	27
	0.7.2	Description et installation d'un package graphics R	25
	0.7.1	Introduction	25
0.7	packag	ge graphics R	25
	0.6.3	Evaluation: mmetric, mgraph, mining	23
	0.6.2	Modeling: fit, predict	22
	0.6.1	Data preparation : CasesSeries, delevels	21
0.6	Packag	ge RMiner: Data Mining Classification et Méthode de Regression	20
	0.5.1	Contexte	5
0.5	Applic	ations	5
	0.4.1	Jeux de données	4
0.4	Accéd	ler au contenu d'un package R chargé	3
0.3	Installa	ation et chargement d'un package R	3
	0.2.1	Les fonctions d'Evir et Evd	2
0.2	Evir e	t evd ···	2
0.1		action	1

0.1 Introduction

Dans une session R, nous avons accès à un bon nombre de fonctions et de jeux de données. Les objets accessibles sont ceux contenus dans les packages R chargés à l'ouverture de la session. Un package R est simplement un regroupement de fonctions et de données documentées. Ce tutoriel à pour but de vous faire une presentation de certains packages qui ont été choisis à savoir evir, evd, R.miner, graphics, regtest, packages qui nous permettrons :

- D'explorer le monde de la théorie des valeurs extrêmes (Cours intéressant sur la Statisque des valeurs extrêmes) en modélisant les risques extrêmes;
- Travailler d'une manière générale sur la Régression.

Projet R 1 PSB 2020-2022

0.2 Evir et evd

Evir et Evd : Qu'est ce que c'est?

Evir est un package de R utiliser dans la théorie de valeurs extrêmes, qui peuvent être divisées dans le groupes suivants; analyse exploratoire des données, maxima de bloc, pics au-dessus d'un seuil (univarié et bivarié), processus ponctuels, distributions gev/gpd.

Quant-à

Evd elle étend les fonctions de simulation, de distribution, de quantile de valeurs extrêmes paramétriques univariées et multivariées, et fournit d'ajustement qui calculent les estimations du maximum de vraisemblance pour des modèles maxima univariées et bivariées, et pour les modèles à seuil univariées et bivariées.

0.2.1 Les fonctions d'Evir et Evd

Ici nous allons recenser certaines fonctions qui peuvent être utilisée dans ces deux packages, commençont par *Evid*.

Listes des fonctions de Evir

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 d'ajuster les valeurs des valeurs extrêmes généralisées

gpd : Permet d'ajuste le modèle Pareto généralisé

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

interpret.gpdbiv: Interprétation des résultats de l'ajustement pgd bivarié

rgpd : Distribution de la Pareto généralisée

Listes des fonctions de Evd

Distributoion de frchet

failure: Temps d'échec

dextreme: Distributions des maxima et es minima

```
dmvevd: Distributions paramétriques et valeurs extrêmes multivariées
```

fextreme : Ajustement du maximum de vraisemblance des maxima et des minima

afrechet : Distribution de Frchet

qnweibull : distribution inverse de weibull

rorder: Distributions des statistiques d'ordres

confint.evd: Calcule les intervalles de confiance

0.3 Installation et chargement d'un package R

L'installation d'un package et le chargement d'un package sont deux étapes distinctes.

Certains packages R sont installés automatiquement lors de l'installation de R.

La fonction installed.packages retourne des informations à propos des packages R installés sur l'ordinateur local.

Il est simple de charger en R des packages supplémentaires à ceux chargés par défaut. Il suffit d'utiliser les commandes comme dans l'exemple suivant :

```
<mark>Install</mark>.packages("evir")
<mark>libr</mark>ary(evir)
```

```
Arnaud <- installed.packages()
> head(Arnaud, n = 3)
              Package
                                                                                                                  abind" "C:/Users/yanko/Documents/R/win-library/3.6" "1.4-5" NA 'actuar" "C:/Users/yanko/Documents/R/win-library/3.6" "2.3-3" NA 'asserthat" "C:/Users/yanko/Documents/R/win-library/3.6" "0.2.1" NA
abind
               "abind"
actuar
assertthat
                                                  KO/DOCUMENTS/ N/...
Enhances License License NA "LGPL (>= 2)" NA
                                                                                                                                       "tools
                                                                                 License_is_FOSS License_restricts_use OS_type MD5sum NeedsCompilation
              LinkingTo Suggests
abind
               NA
                            NA
                                                                                                                                                         "no'
                            "MASS"
                            "testthat, covr" NA
assertthat
              NΔ
              Built
               "3.6.0"
abind
               '3.6.2"
actuar
assertthat "3.6.3"
```

0.4 Accéder au contenu d'un package R chargé

Une fois un package chargé en R avec la commande library, son contenu est accessible dans la session R. Nous avons vu dans des notes précédentes comment fonctionne l'évaluation d'expressions en R. Nous savons donc que le chargement d'un nouveau package ajoute un environnement dans le chemin de recherche de R, juste en dessous de l'environnement de travail. Le

chargement de certains packages provoque aussi le chargement de packages dont ils dépendent. Ainsi, parfois plus d'un environnement est ajouté au chemin de recherche de R lors du chargement d'un package. L'environnement d'un package contient les fonctions publiques et les données du package.

0.4.1 Jeux de données

Souvent, les jeux de données inclus dans un package se retrouvent directement dans l'environnement d'un package dans le chemin de recherche. C'est le cas, par exemple, des jeux de données du package datasets.

```
\frac{head(ls("package : datasets"), n = 8)}{}
```

```
> head(ls("package:datasets"), n = 8)
[1] "ability.cov" "airmiles" "AirPassengers" "airquality" "anscombe" "attenu" "attitude" "austres"
> |
```

Dans notre cas espèce seulement les 8 premiers éléments de la liste sont affichés ici, car cette liste compte normalement 104 éléments.

Cependant, les jeux de données sont parfois cachés. Ils sont alors traités différemment des fonctions privées et ne se retrouvent même pas dans l'espace de noms du package.

La fonction data est très utile dans ce cas. Cette fonction a en fait plusieurs utilités.

Premièrement, elle permet d'énumérer tous les jeux de données contenus dans un package.

**data(package = "evir")*

Dans notre cas espèce, nous allons à partir du packages "evir", afficher le jeux de données nidd.annual, ces données représentent les niveaux maximaux annuels de la $rivière\ Nidd$ dans le Yorkshire..

On a:

```
install.packages("evir")
library(evir)
donnes<-data(nidd.annual)
donnes<-nidd.annual
donnes</pre>
```

Et on a la sortie suivante:

(–) Format

Nous obtenons en sortir un vecteur numérique contenant 35 observations.

0.5 Applications

Pour application on utilise les données des crues annuelles de la rivière Nidd, dans le Yorkshire.

0.5.1 Contexte

La hauteur d'une rivière est modélisée par une variable aléatoire X.

On dispose de $\{X_1, \dots, X_n\}$ un échantillon de hauteurs d'eau annuelles. On note $X_{1,n} \le X_{2,n} \le \dots, \le X_{n,n}$ l'échantillon ordonné, avec $n \in \mathbb{N}$.

Deux problèmes complémentaires :

* Calculer la probabilité p d'une hauteur d'eau h extrême

$$p = \mathbf{P}(X \ge h)$$
 avec $h > X_{n,n}$.

* Calculer le niveau d'eau h qui est atteint ou dépassé une seule fois sur T > n, i.e. résoudre $\frac{1/T}{T} = \mathbf{P}(X \ge h)$

Modélisation : Le but ici étant d'écrire un programme permettant de calculer la fonction empirique des excès moyens et de tracer les différents seuils :

Application 1

Allons y:

 (-) On Charge au préalable nos deux packages, les bouts de code suivant permettent de faire ce travail :

```
library(evd)
library(evir)
```

(-) Nous allons commencer par lire les données du fichier Nidd et les classé par ordre décroissant dans un vecteur noté nidd. On creé un vecteur seuil ordonnées par ordre.

```
data(nidd.annual)
nidd<-nidd.annual
nidd

hidd2<-sort(nidd, decreasing = TRUE)
nidd2
seuil=seq(70,300,by =5)
#seuil<-70:300
seuil
taille_seuil=length(seuil)
taille_nidd2=length(nidd2)</pre>
```

(-) Afin de calculer la moyenne des excès pour chaque seuil, on crée 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 de seuil[i].

Puisque certains excès peuvent être négatifs, nous mettons les zéros à la place des termes négatifs :

```
X=matrix(0,nrow=length(seuil),ncol=length(nidd2))
X
#View(X)

for(i in 1:length(seuil))
{for(j in 1:length(nidd2))
{X[i,j]=max(nidd2[j]-seuil[i],0)}
}
```

(–) On calcule pour chaque seuil la moyenne des excès :

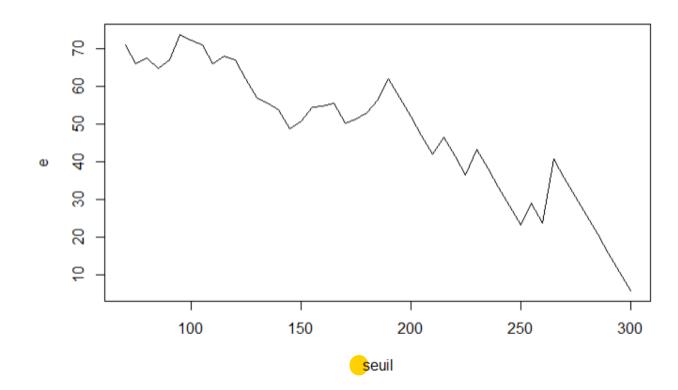
```
somme=rep(0,length(seuil))
somme
Compteur=rep(0,length(seuil))
Compteur
e=c()
e

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]=1
      }
   e[i]=somme[i]/Compteur[i]
}
```

(-) La fonction moyenne des excès est une méthode permettant de répondre à l'une des dificultés de la modélisation des sinistres extrêmes qui est la détermination du seuil. Ainsi, graphiquement il est possible de déterminer le seuil le plus adéquat, en prenant la valeur à partir de laquelle la fonction moyenne des excès est linéaire.

Regardons ensemble ce que cela produit :

plot(seuil,e,type='l')

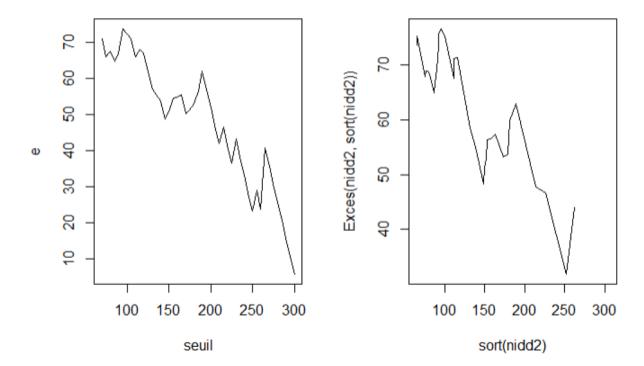


Dans notre cas, La fonction moyenne des excès est linéaire à partir de la valeur 260.

(-) Ici on trace sur deux graphiques juxtaposés : la fonction empirique des excès moyens en fonction des statistiques d'ordre $x_{(n-k)}$ et en fonction de k.

Le 1er représente la moyenne des dépassements des crues annuelles de la rivière Nidd par rapport à un certain seuil (indépendants de nos données de départ). Le 2eme représente la moyenne des dépassements de la crue annuelle de la rivière Nidd d'une année par rapport aux autres.

```
Exces=function(observations, seuil)
 Exces2=matrix(0,nrow=length(seuil),ncol=length(observations))
 for(i in 1:length(seuil))
   for(j in 1:length(observations))
     Exces2[i,j]=max(observations[j]-seuil[i],0)
 Exces2
 somme2=rep(0,length(seuil))
 commet cap(), length(seuil))
compteur2=rep(0,length(seuil))
compteur2
 for(i in 1:length(seuil))
    for(j in 1:length(observations))
       if (Exces2[i,j]>0)
         somme2[i]=somme2[i]+Exces2[i,j]
         Compteur2[i]=Compteur2[i]+1
    e2[i]=somme2[i]/Compteur2[i]
 e2
 Exces(nidd2,sort(nidd2))
par(mfrow=c(1,2))
plot(seuil,e,type='l')
plot(sort(nidd2),Exces(nidd2,sort(nidd2)),type='l')
```



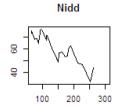
Les deux graphique sont la même allure, et la fonction moyenne des excès devient linéaire aux environs de la valeur 260.

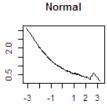
Pour aller plus loin l'idéal serait d'appliquer cela aux différentes lois usuelles : La loi

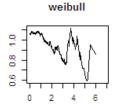
Pareto, Weibull, Frechet, Gamma, Cauchy, Normal et bien d'autres : (Les differents lois en Statisque des valeurs extrêmes)

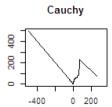
(–) Application aux lois usuelles

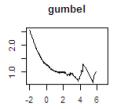
```
##########101 Normal Centree reduite
Nidd_normal=rnorm(1000,0,1)
Nidd_normal_dec=sort(Nidd_normal, decreasing = TRUE)
seuil_stat_ord=sort(Nidd_normal, decreasing = FALSE)
#Exces(Nidd_normal_dec,seuil_stat_ord)
nage(mfnuer(2,4))
par(mrow=c(3,4))
plot(sort(nidd2), Exces(nidd2, sort(nidd2)), type='l', main='Nidd', xlab = "", ylab = "")
plot(seuil_stat_ord, Exces(Nidd_normal_dec, seuil_stat_ord), type='l', main='Normal', xlab = "", ylab = "")
prot(seuri_stat_ord,exces\nidu_normar_dec,seuri_stat_o
##########|oi weibull
Nidd_weibull=rweibull(1000, 1)
Nidd_weibull_dec-sort(Nidd_weibull, decreasing = TRUE)
seuri_stat_ord=sort(Nidd_weibull, decreasing = FALSE)
#Exces(Nidd_weibull_dec,seuil_stat_ord)
#par(mfrow=c(2,2))
"par(mirow=c(2,2))
#plot(sort(nidd2),Exces(nidd2,sort(nidd2)),type='l')
plot(seuil_stat_ord,Exces(Nidd_weibull_dec,seuil_stat_ord),type='l',main='weibull',xlab = "" , ylab = "")
#############flot Cauchy
##########floi Cauchy
Nidd_cauchy=rcauchy(1000, 0, 1)
Nidd_cauchy_dec=sort(Nidd_cauchy, decreasing = TRUE)
Seuil_stat_ord=sort(Nidd_cauchy, decreasing = FALSE)
#Exces(Nidd_cauchy_dec_seuil_stat_ord)
#par(mfrow=(2,2))
#par(sort(Nidd)_record(Nidd)_record(Nidd)
#par(mirrow=((2,2))
#plot(sort(nidd2),Exces(nidd2,sort(nidd2)),type='1')
plot(seuil_stat_ord,Exces(Nidd_cauchy_dec,seuil_stat_ord),type='1',main='Cauchy',xlab = "" , ylab = "")
 #plot(sort(nidd2),Exces(nidd2,sort(nidd2)),type='l')
plot(seuil_stat_ord,Exces(Nidd_gumbel_dec,seuil_stat_ord),type='l',main='gumbel',xlab = "" , ylab = "")
*/pa(univer-(2,2))
#plot(serin_sxees(nidd2,sort(nidd2)),type='1')
plot(seuil_stat_ord,Exces(Nidd_frechet_dec,seuil_stat_ord),type='1',main='frechet',xlab = "" , ylab = "")
##########ioi expo
Nidd_Expo=exp(1000,1)
Nidd_Expo_exp(1000,1)
Nidd_Expo_dec=Sort(Nidd_Expo, decreasing = TRUE)
seuil_stat_ord=sort(Nidd_Expo, decreasing = FALSE)
#Exces(Nidd_Expo_dec,seuil_stat_ord)
#par(mfrow=c(2,2))
 #|Dot(sort(nidd2),Exces(nidd2,sort(nidd2)),type='1')
plot(seuil_stat_ord,Exces(Nidd_Expo_dec,seuil_stat_ord),type='1',main='Expo',xlab = "" , ylab = "")
#plot(sort(nidd2),Exces(nidd2),exces(nidd2),type='1')
plot(seut].stat_ord,Exces(Nidd_unifo_dec,seuil_stat_ord),type='l',main='uniforme',xlab = "" , ylab = "")
##############|oi Gamma
Nidd_LOGamma=rlgamma(1000,1,1)
Nidd_LOGamma_dec=sort(Nidd_LOGamma, decreasing = TRUE)
Seuil_stat_ord=sort(Nidd_LOGamma, decreasing = FALSE)
#Exces(Nidd_Gamma_dec,seuil_stat_ord)
"pa (millow-L(z,z))
#jlot(sort(nidd2), Exces(nidd2, sort(nidd2)), type='l')
plot(seuil_stat_ord, Exces(Nidd_LoGamma_dec, seuil_stat_ord), type='l', main='LoGamma', xlab = "" , ylab = "")
plot(seu11_stat_uru,cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsurus_cactsu
#pa (uniform-(2,2))
#plot(serin_stat_ord,Exces(niddz,sort(niddz)),type='1')
plot(seuil_stat_ord,Exces(Nidd_Pareto_dec,seuil_stat_ord),type='1',main='Pareto',xlab = "" , ylab = "")
```

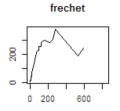


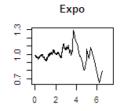




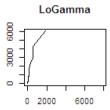


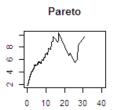












(-) utilisons la fonction "meplot" du package "evir" pour tracer la fonction empirique des excès moyens pour différentes lois.

Normal = rnorm(1000,0,1)

Weibull = rweibull(1000, 1)

Cauchy = rcauchy(1000, 0, 1)

Frechet = rfrechet(1000, 10c=0, scale=1, shape=1)

Expo=rexp(1000,1)

UniTo=runif(1000,0,1)

Pareto-pareto(1000,2,2)

Close.screen(all = TNUE)

par(mfrowc(3,4))

meplot(vidu 3, type="1", main="NIDO")

meplot(Normal, 3, type="1", main="NEJUL")

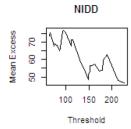
meplot(Couchull, 3, type="1", main="CAUCHY")

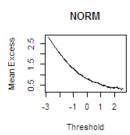
meplot(Cauchy, 2, type="1", main="CAUCHY")

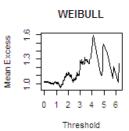
meplot(Frechet, 3, type="1", main="RECHET")

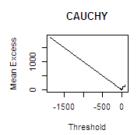
meplot(Frechet, 3, type="1", main="RECHET")

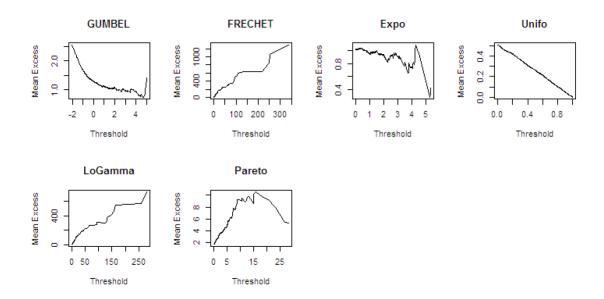
meplot(Unifo, 3, type="1", main="Unifo")











Application 2 : On termine le seuil avec le Graphique de HILL

le graphique de Hill est un estimateur qui est défini uniquement pour $\psi > 0$, c'est-à-dire dans le cas où la distribution des valeurs extrêmes correspond à une distribution de Fréchet. Le seuil, dans ce cas, corresponds à la valeur à partir de laquelle la fonction de Hill est assimilable à une droite horizontale.

Implémentation

1. Commençons par simuler une GPD à l'aide de rgpd:

```
GP=rgpd(1000,0.4,0.9)

#GP=rgpd(1000, 0.30)

#GP=rgpd(500,0.4,0,1)

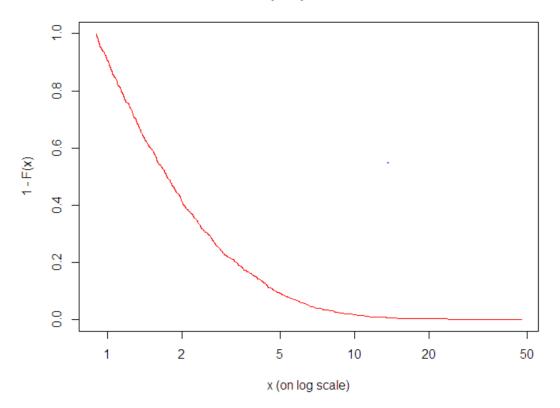
GP
```

2. On trace la fonction de survie empirique des excès à l'aide de la fonction *emplot* :

```
emplot(GP, type="l", main="Fonction empirique des excès GPD",col="red")
```

Le graphe obtenu est le suivant :

Fonction empirique des excès GPD



3. On utilise la fonction gpd pour calculer les estimateurs de γ et σ :

Par la méthode de maximum de vraisemblance

4. On trace les fonctions de survie des deux *GPD* estimés sur un même graphe :

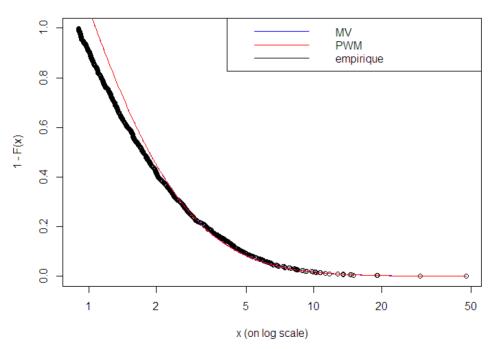
```
fgpd=function(loi,x){
    n=length(loi)

sigma_ml=gpd(loi, nextremes=n, method="ml")$par.ests[1]
    gamma_ml=gpd(loi, nextremes=n, method="ml")$par.ests[2]
    sigma_pwm=gpd(loi, nextremes=n, method="pwm")$par.ests[1]
    gamma_bwm=gpd(loi, nextremes=n, method="pwm")$par.ests[2]

emplot(GP, main="Fonction empirique des excès GPD")
    lines(x,1-pgpd(x,sigma_ml,gamma_ml),col="blue",main="Fonction de survie",type="l")
    lines(x,1-pgpd(x,sigma_pwm,gamma_pwm),col="red")
    legend(legend=c("MV","PWM","empirique"),"topright",col=c("blue","red","black"),lty=c(1,1,1))
}
x=sort(GP)
x=seq(x[1],x[length(x)], 0.01)|
fgpd(GP,x)
```

Visualisation

Fonction empirique des excès GPD

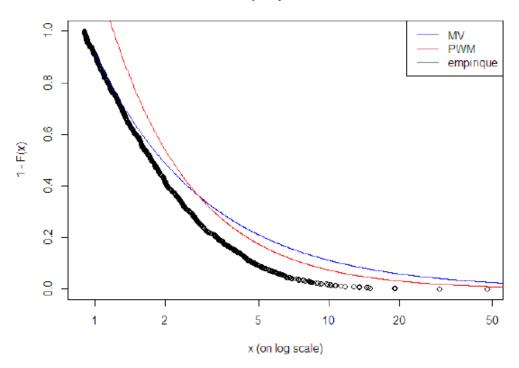


5. Appliquons ceci à d'autres lois usuelles

Loi LogGamma:

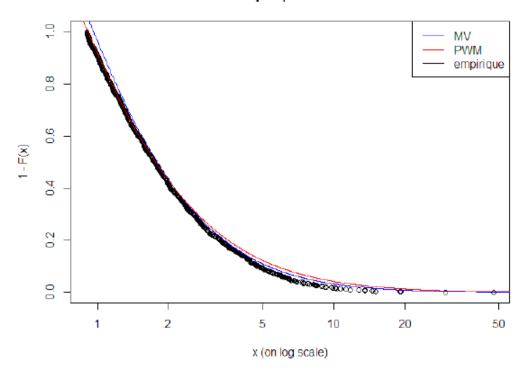
```
logamma=rlgamma(1000,1,1)
x=sort(logamma)
x=seq(x[1],x[length(x)], 0.01)
fgpd(logamma,x)
```

Fonction empirique des excès GPD



Loi Pareto: Pareto=rpareto(1000,2,2) x=sort(Pareto) x=seq(x[1],x[]ength(x)], 0.01) fgpd(Pareto,x)

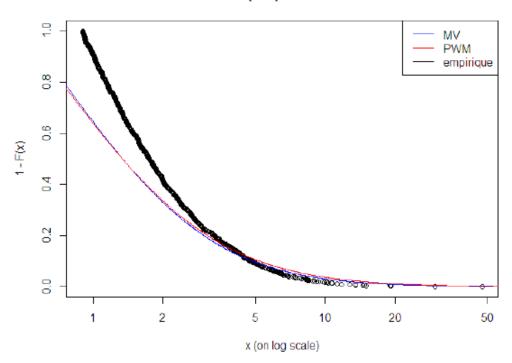
Fonction empirique des excès GPD



Loi bur:

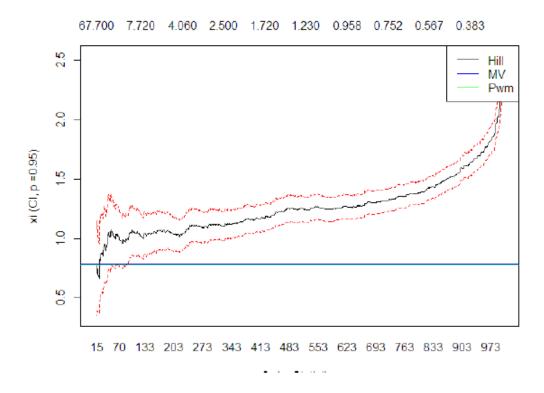
```
bur=rburr(1000,1,1,2)
x=sort(bur)
x=seq(x[1],x[length(x)], 0.01)
fgpd(bur,x)
```

Fonction empirique des excès GPD

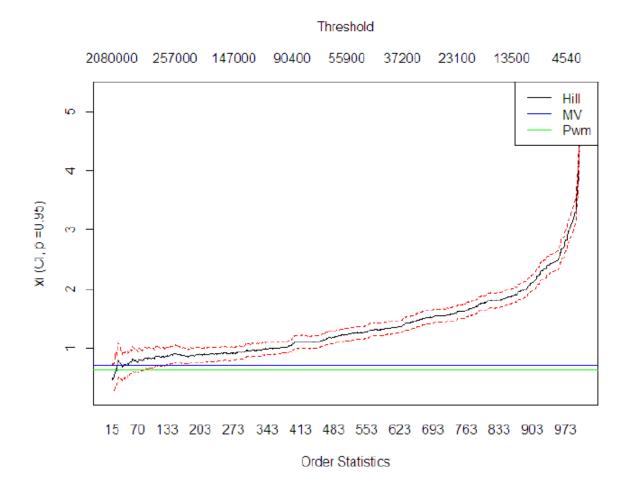


6. On trace sur un même graphique les 3 estimateurs γ : PWM, EMV, Hill

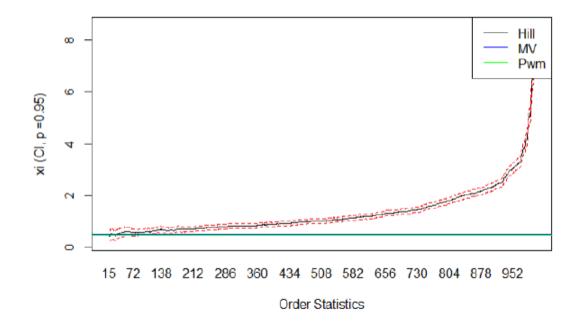
Loi Frechet:



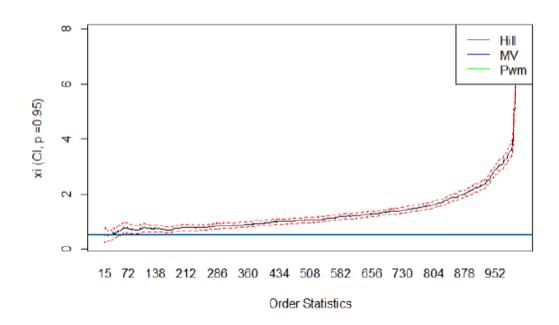
Loi log-gamma



Loi pareto



Loi bur



En conclusion, à partir des graphes obtenus precédemment, pour les lois considérées, la fonc-

tion Hill devient une droite horizontale aux environs de la valeur 260, est l'estimateur de Hill est proche de EMV et PWM. On peut donc considérer que le domaine d'attraction de la distribution des crues est Fr chet.

0.6 Package RMiner : Data Mining Classification et Méthode de Regression

Rminer: kesako?

Le package Rminer aide à la manipulation des données algorithmiques pour le data mining concernant la classification et la régression (incluant la prévision de series temporelles). Pour utiliser le package Rminer et ses fonctions :

- vérifier si le package est installé dans R;
- sinon installer le package puis l'appeler avec la library

Install.packages(« rminer »)
Library(rminer)

Les fonctions de Rminer

Ci-desous la liste des fonctions du package Rminer. Nous n'allons vous présenter que les fonctions qui nous semblent les plus pertinentes dans ce tutoriat.

Si vous souhaitez approfondir votre connaissance sur les autres fonctions, nous vous invitons à aller sur la documentation R du package : Lien Si vous souhaitez consulter l'aide, taper dans la commande R (help(package=rminer))

Liste des fonctions de Rminer

```
|CasesSeries|
|crossvaldata|
|delevels|
|fit|
|holdout|
|Importance|
|imputation|
|lforecast|
|mgraph|
|mining|
|mmetric|
|mparheuristic|
|predict.fit|
|savemining|
|sa fri1|
|sin1reg|
|vecplot|
```

Ce tutoriat se concentra sur les fonctions suivantes du package Rminer qui peuvent être réparties dans 3 phases dans l'analyse des données :

• Data preparation : CasesSeries, delevels, imputation,

• Modeling: fit, predict, mining,

• Evaluation : mmetric, mgraph, mining

0.6.1 Data preparation: CasesSeries, delevels

Cases Series: créer un data frame d'une série temporelle en utilisant la fenêtre glissante.

Fenêtre glissante : (« sliding window ») : Fenêtre temporelle utilisée dans l'exploration des flots de données (« Data stream mining ») pour en extraire des motifs. Les fenêtres peuvent avoir une taille W fixe et la fouille de données s'effectue sur les W dernières transactions, ou sur les W dernières unités de temps, elles peuvent aussi avoir une taille variable, dans la détection de la dérive conceptuelle.

```
# Exemple
> t=1:20
> d=CasesSeries(1:10,c(1,3,4))
> print(d)
[1] lag4 lag3 lag1 y
  1 2 4 5
    2
        3
    3
        4
        5
5
   5 6 8 9
    6
        7
            9 10
> d=CasesSeries(1:10,c(1,2,3))
> print(d)
[1] lag3 lag2 lag1 y
   1 2 3 4
2 3 4 5
       3
2
3
   3
        4
4
    4
        5
             6 7
5
    5
        6
             7 8
    6
        7
             8 9
            9 10
==> la fonction retourne un data frame où y est cible de l'output et les inputs sont le temps de latence
```

Delevels : réduire, remplace ou transforme les niveaux du data frame et la variable du facteur

```
> f=factor(c("A","A","B","B","C","D","E"))
> print(table(f))
[1] f
ABCDE
2 2 1 1 1
# remplacer "A" en "a":
> f1=delevels(f,"A","a")
> print(table(f1))
[1] f1
a B C D E
2 2 1 1 1
# combiner c("C","D","E") en "CDE":
> f2=delevels(f,c("C","D","E"),"CDE")
> print(table(f2))
[1] f2
A B CDE
  2 2 3
# combiner c("B","C","D","E") en _OTHER:
> f3=delevels(f,c("B","C","D","E"))
> print(table(f3))
[1] f3
    A _OTHER
    2
```

0.6.2 Modeling: fit, predict

Rminer comprend 14 méthodes de classification et 15 méthodes de régression, tout directement disponible à travers les facteurs *fit, predict, mining*. *Fit* la fonction ajuste un modele selectionné de jeux de données et peur automatique ajuster les hyperparamètres. Les hyperparamètres sont des paramètres réglables qui vous permettent de contrôler le processus d'entraînement du modèle.

Predict: la fonction donne un modèle ajusté et calcule les prédictions pour un jeu de données. Les fonctions *fit* et *predict* proposent plusieurs modèles tels que :

- rpart (arbre de décision),
- randomForest,
- lm (régression linéaire ou multiple),
- cv.glmnet : modèle linéaire généralisé.

Par defaut, le type du modèle Rminer (classification ou régression) dépend du type d'output.

- Si c'est un facteur (discret) alors c'est une probabilité de classificaiton
- Si c'est numérique (int, num) alors c'est une régression qui est executée.

Voir tous les modèles de fit et predict : lien fit et lien predict

```
# Exemple qui montre comment la transformation fonctionne avec `fit` et `predict`:

M=fit(y~.,data=sa_ssin,model="mr") # régression linéaire

P=predict(M,data.frame(x1=-1000,x2=0,x3=0,x4=0,y=NA)) # P devrait être négatif

print(P)

[1] -0.4144042

M=fit(y~.,data=sa_ssin,model="mr",transform="positive")

P=predict(M,data.frame(x1=-1000,x2=0,x3=0,x4=0,y=NA)) # P n'est pas négatif

print(P)

[1] 0
```

0.6.3 Evaluation: mmetric, mgraph, mining

Rminer comprend une large sélection de métriques d'évaluation et de graphes qui peut être utilisée pour évaluer la qualité des modèles ajsutés et extraire les données apprises du modèle data-driven.

Mmetric : fonction qui calcule les erreurs métriques de classification ou régression. Cidessous les quelques mesures de mmetric :

- ALL : sort toutes les mesures de mmetric
- F1 score, [0 100%],
- TPR (true positive rate) : c'est la sensibilité, correspondant au taux de vrais positifs, [0-100%],
- PRECISION : la précision, c'est-à-dire le nombre de documents pertinents retrouvés rapporté au nombre de documents total proposé pour une requête donnée,
- ACC: taux d'exactitude de classification, [0-100%],
- ACCLASS: taux d?exactitude de classification par classe, [0 100%],
- TPR (true positive rate) : c'est la sensibilité, correspondant au taux de vrais positifs, [0-100%].

Voir tous les modèles de *mmetric* : lien

```
# Exemple
> y=factor(c("a","a","a","a","b","b","b","b"))
> x=factor(c("a","a","b","a","b","a","b","a"))
> print(mmetric(y,x,"CONF")$conf)
[1] pred
target a b
   a 3 1
   b 2 2
> print(mmetric(y,x,metric=c("ACC","TPR","ACCLASS")))
[1] ACC TPR1 TPR2 ACCLASS1 ACCLASS2
  62.5 75.0 50.0 62.5 62.5
> print(mmetric(y,x,"ALL"))
                       BER KAPPA CRAMERV ACCLASS1 ACCLASS2 BAL_ACC1
[1] ACC
           CE
62.5000000 37.5000000 37.5000000 25.0000000 0.0000000 62.5000000 62.5000000 62.5000000
BAL ACC2 TPR1 TPR2 TNR1 TNR2 PRECISION1 PRECISION2 F11
62.5000000 75.0000000 50.0000000 50.0000000 75.0000000 60.0000000 66.6666667 66.6666667
 F12 MCC1 MCC2
57.1428571 0.5163978 0.5163978
```

Sin1reg : un simple jeu de données avec 1000 points où y=0.7*sin(pi*x1/2000)+0.3*sin(pi*x2/2000)

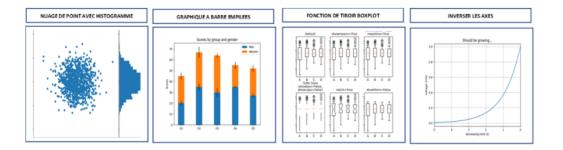
Mgraph : fonction graphique

Mining : la fonction réalise plusieurs exécutions ajustées et de prédictions, selon une méthode de validation et donne un nombre d'exécutions.

0.7 package graphics R

0.7.1 Introduction

Le logiciel R est l'un des logiciel inconpublement reconnu pour ces beaux graphiques hyper convaincant qu'il produit dont entre autre (Les bandes linéaires, les bandes en pointillé ou même les bandes de nuages, \cdots) Voir image ci-dessous



QU'EST-CE QUE LE BUT D'UN GRAPHIQUE

- Mieux comprendre le problème;
- Mieux Expliquer un phénomène
- Bref le but est de nous nous aider à rendre une situation abstraite en information convaincante.

Pour faire ces graphiques , le logiciels "R" a mis à disposition un packages à cet effet qui , de par ces multiples commandes met en exécution une fonction de tracé.

Nous allons donner une brève description de ces $\frac{1}{2}$ trois packages en mettant un accent particulier sur la commande "PLOT" ci-dessous.

0.7.2 Description et installation d'un package graphics R

La bibliothèque graphique est le package graphique de base standard fourni avec l'installation R. Cette bibliothèque a été développée à l'origine pour R. De nombreux tracés standard

tels que des tracés de base de points et de lignes, des histogrammes, des graphiques à barres, des camemberts, des bplots, etc. peuvent être dessinés avec cette bibliothèque.

À toutes fins pratiques, la bibliothèque graphique est suffisamment silencieuse pour créer des tracés et des graphiques d'excellente qualité que nous utilisons généralement pour l'analyse des données et les statistiques.

Ce package est fourni avec l'installation de R.

Le paquet de grille

<< Le système graphique de <u>grille</u> a ensuite été développé par Paul Murrel et ajouté à R. Il s'agit d'un système graphique de bas niveau qui permet de dessiner et d'organiser des formes géométriques de base telles que des polygones, des courbes, des images raster, etc. Le paquet de grille contient des fonctions pour accéder au canevas et permet la création de plusieurs régions appelées fenêtres sur une seule toile.

Ce package est fourni avec l'installation R. Nous devons charger la bibliothèque de grille dans R avant de l'utiliser. (Le package graphique se charge automatiquement lorsque nous démarrons R).

Pour charger la bibliothèque de grille dans R, tapez

> bibliothèque ("grille")

Le paquet lattice

Le <u>treillis</u>, développé par Deepayan Sarkar, est un système de visualisation de haut niveau basé sur une méthode appelée graphique en treillis. Ce package gère très efficacement les données multivariées.

Le package lattice se compose de fonctions de haut niveau pour chaque tâche. Ces fonctions renvoient des objets qui peuvent être convertis en graphiques par les fonctions plot () du package R de base. Ce package est basé sur le moteur graphique de grille mentionné cidessus.

Ce package est également fourni avec l'installation de R et nécessite le chargement du package grDevices .

Pour charger la bibliothèque grDevices dans R, tapez

> bibliothèque ("grDevices")

Le paquet ggplot2

Le ggplot2 est une bibliothèque graphique pour R, créée par Hadley Wiskham. Il est mentionné dans sa <u>page d'accueil</u> que "ggplot2 est un système de traçage pour R, basé sur la grammaire des graphiques, qui essaie de prendre les bonnes parties des graphiques de base et de treillis et aucune des mauvaises parties". (Le <u>Grammer of Graphics</u> mentionné ici est un livre classique sur les méthodes graphiques pour la visualisation de données scientifiques écrit par Leyland Wilkinson).

Nous pouvons créer des parcelles très élégantes avec cette bibliothèque.

Le ggplot2 est un package externe qui doit être téléchargé de l'intérieur R. Pour installer ce package en ligne à partir de l'invite R, tapez

> install.packages ("ggplot2")

Après avoir installé une fois, pour charger le ggplot2 dans R, tapez

> bibliothèque ("ggplot2") >>

0.7.3 Applications

R avec plot(), points(), lines(), polygon

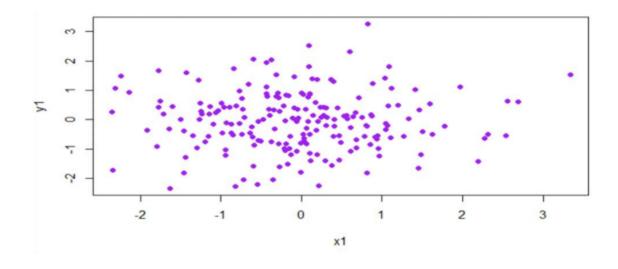
La fonction plot () - tracer des points et des lignes

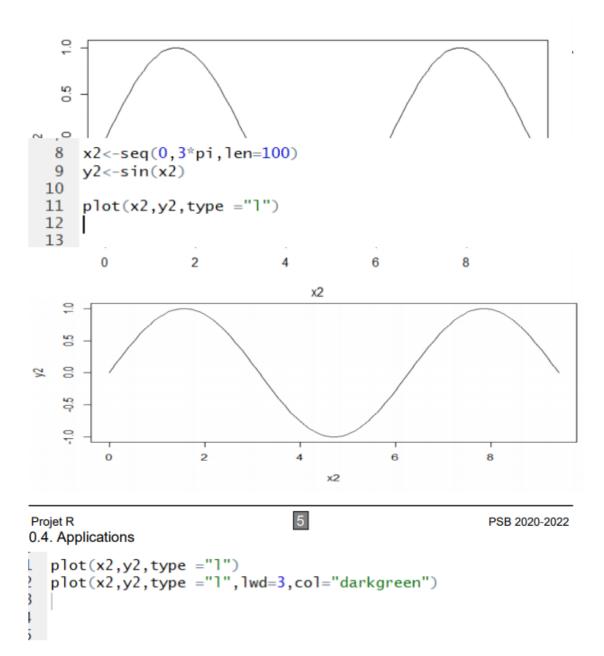
Le tracé par défaut . Les tracés de points et de lignes peuvent être produits en utilisant ter-rain() fonction, qui prend les points x et y sous forme de vecteurs ou de nombre unique avec de nombreux autres paramètres. Les paramètres x et y sont nécessaires. Pour d'autres, la valeur par défaut sera utilisée en l'absence de la valeur. Dans les lignes de commande ci-dessous, nous créons d'abord une paire de séquences x et y et les passons en paramètres au terrain() fonction :

Commande d'exécution de plot avec un graphique en nuage de point

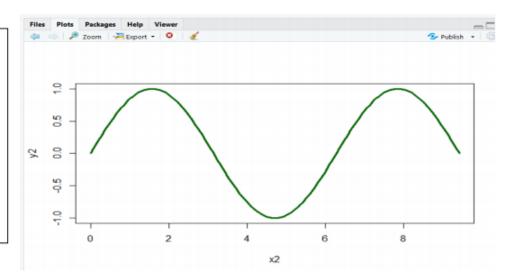
```
Untitled1* × O Untitled2 × Telco_customer_churn_status × O Untitled3* × D Name of the source of the
```

Résultat de l'application avec plot (Graphique nuage de point)



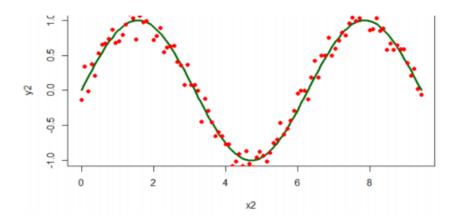


Avec ce graphique nous constatons que la couleur de la ligne a changé et l'épaisseur de la ligne ...



Exemple de graphique en pointillé pris aléatoirement

```
14 y2.rand<-y2+rnorm(100,0,0.1)
15 points(x2,y2.rand,pch=16,col="red")
16
17
```



Bibliographie *

- [1] Site le CRAN https://cran.r-project.org/web/packages/available packages by name.html available-packages-E
- [2] https://cran.r-project.org/web/packages/
- [3] Sidney I. Resnick Extreme Values, Regular Variation and Point Processes
- [4] Stuart Coles An Introduction to Statistical Modeling of Extreme Values
- [5] Laurens de Haan Extreme value theory
- [6] Documentation R sur Rminer
- [7] Cortez, P. (2010). Data Mining with Neural Networks and Support Vector Machines using the R/rminer Tool. In Perner, P., editor, Advances in Data Mining Applications and Theoretical Aspects, 10th Industrial Conference on Data Mining, pages 572583, Berlin, Germany. LNAI 6171, Springer.
- [8] Wikipedia