Supervised Study Project in Mathematics, Comparaison des tests de normalite

MORVAN Guy, DOUSSET de la street, Mel de la dance

Dans longtemps en Mai, 2020

Table des matières

1 Introduction

1.1 Notions requises

$H0$ *est vraie*
— Accepter $H0$: Décision juste (probabilité = 1 - α)
— Rejeter $H0$: Erreur de $1^{ière}$ espèce : rejet de $H0$ à tort (probabilité = α)
$H0$ *est fausse*
— Accepter $H0$: Erreur de $2^{ième}$ espèce : acceptation de $H0$ à tort (probabilité = β)
— Rejeter $H0$: Décision juste (probabilité = 1 - β)

On appelle P la puissance d'un test statistique : $P = 1 - \beta$ C'est la probabilité de rejeter l'hypothèse nulle alors qu'elle est fausse.

La courbe de puissance du test est la représentation graphique de cette probabilité $1 - \beta$ quand la taille de l′échantillon n augmente. Cette fonction "puissance du test"" qui associe à n la probabilité 1 - β de rejeter avec raison H0 est une fonction croissante dont la limite à l'infini est 1. Un test est dit "meilleur" qu'un autre si la croissance de sa fonction puissance est plus rapide.

Rappels importants :

Un test statistique implique un estimateur T($X_1 \ldots X_n$), c'est une variable aléatoire (fonction des variables aléatoire X_i et de n). elle a donc une densité de probabilité et une Loi. On peut conditionner cette variable aléatoire par les 2 évènements complémentaires : + les $(X_1, ..., X_n)$ vérifient $H0$ + les $(X_1, ..., X_n)$ v ? ?rifient $H1$

On note : $T(X_1, ..., X_n)|H0$ et $T(X_1, ..., X_n)|H1$. Celles-ci sont des variables aléatoires conditionnelles qui ont aussi une densité et une loi.

La loi qui nous intéresse dans un test statistique est la loi sachant H0 : $T(X_1, ..., X_n)|H0$ dont on veut les quantiles afin de définir une Région critique de risque $\alpha = 5\%$. On veut une preuve forte contre $H0$ et c'est la $RC(\alpha)$ qui nous l'apporte. Si $T(x_1, ..., x_n)$ (notre réalisation de $T(X_1, ..., X_n)$), appartient à $RC(\alpha)$ alors il y a moins de 5% de chance (preuve forte) que notre échantillon (x1. . . .xn) vérifie $H0$: Donc on rejette l'hypothese que $(x_1, ..., x_n)$ vérifie $H0$

Maintenant que tout est clair on peut attaquer le TER NON NON NON ET NON reformule moi cette merde ! ! ! ! ! ! ! :

2 Théorie et méthodologie :

2.1 Estimation de la puissance d'un test de normalité

Soit $X_1, ..., X_n$ un échantillon de variables aléatoires de densité f non-normale.

Un test de normalité, par définition, permet de tester l'hypoth?se : $H0 : f \in \mathcal{N}(\mu, \sigma^2)$ contre $H1 : f \notin \mathcal{N}(\mu, \sigma^2)$ (μ, σ^2 quelconques)

Ici on sait que pour notre échantillon non normalement distribué, $H0$ est fausse. Par conséquent, la probabilité de rejet de $H0$ correspond bien à $1 - \beta$. Si on simule $N = 10000$ fois l'échantillon $X_1, ..., X_n$ et qu'on les soumet à un même test de normalité (α fixé), il devrait y avoir un nombre théorique de rejet de $H0$ égal à $(1 - \beta)N$. Donc, la puissance du test (à α fixé) peut être estimée par : $P = 1 - \beta \approx$ Taux de rejet de

$$H0 = \frac{"Nombre de fois que H0 est rejet??e"}{N}$$

2.2 La Région critique

Pour savoir si $H0$ est rejetée ou non, il va falloir au préalable définir la région critique.

Chaque test de normalité a son propre estimateur T : par exemple, KS estime la fonction de répartition, tandis que SW estime un coefficient de corrélation. Ces estimateurs sont plus compliqués que ceux vus jusqu'à maintenant et ne suivent pas forcement (sous $H0$) des lois connues dont les quantiles sont connus. Leurs lois vont également dépendre de la taille de l'échantillon n (tout comme une $\chi^2(n)$ ou une $St(n)$ dépendent de n).

Or la Région critique $RC(\alpha)$ d'un test dépends des quantiles de sa loi (sous $H0$) et de α Donc il va falloir estimer les quantiles par nous-même pour chaque test et pour chaque n (coucou les statistiques d'ordre et quantiles empiriques).

Certains test sont unilatéraux, d'autres bilatéraux :

$$RC(\alpha) = \left\{T < q_0(\frac{\alpha}{2})\right\} \cup \left\{T > q(\frac{1-\alpha}{2})\right\}$$

si test bilatéral (et asymétrique)

$$RC(\alpha) = \{T < q_0(\alpha)\}$$

si test unilatéral gauche

$$RC(\alpha) = \{T > q_0(1 - \alpha)\}$$

si test unilatéral droite

2.3 Les quantiles empiriques de la Loi sous $H0$ associée á un test de normalité :

Comme les lois sous $H0$ de ces estimateurs sont inconnues, il va falloir falloir estimer, pour chaque lois les quantiles q_0 par les quantiles empiriques.

Méthode : Statistiques d'ordre

On fixe $\alpha = 5\%$ ou $\alpha = 10\%$ Pour des valeurs de n différentes (15 dans l'article mais on va en faire 43 pour lisser les courbes de puissance) :

1. Simuler $N = 50000$ échantillons $(Z_1, .., Z_n) \sim \mathcal{N}(0, 1)$ (donc $H0$ vraie)
   2.Générer alors $N = 50000$ variables aléatoires $T = T(Z_1, ..., Z_n)$ (suivant bien la loi sous $H0$ du Test de normalité car $H0$ vraie)
2. Ordonner les N variables $(T(1), ..., T(N))$
3. Estimer les quantiles q_0 par les quantiles empiriques 5.Bonus : Créer la Fonction quantile de la loi sous $H0$ de l'estimateur T

Avec celle-ci on est capable de calculer tous les quantiles de la loi sous $H0$ de l'estimateur

$$T = T(X_1, ..., X_n)|H0$$

- le quantile empirique $T(N\alpha)$ est un estimateur consistant de $q_0(\alpha)$
- le quantile empirique $T(N(1 - \alpha))$ est un estimateur consistant de $q_0(1 - \alpha)$
- le quantile empirique $T(\frac{N\alpha}{2})$ est un estimateur consistant de $q_0(\frac{\alpha}{2})$ (prendre les parties entières supérieures)

2.4 Les différents tests de normalité :

Les types de Tests de Normalité :

— Régression et coefficient de corrélation : SW, SF, RJ

— Chi-2 : CSQ

— Fonction de répartition empirique : KS, LL, AD, CVM

— Méthode des Moments : test du coefficient d'asymétrie, $Kurtosis$, JB, DP

— Espacements : Rao, $Greenwood$

— Autres

L'article compare les 8 tests en rouge.

JB : trop long en calcul (pour $n = 1000$ ça prend 30min et on a 43 valeurs de n à tester) on laisse tomber

KS : ce test fonctionne pour μ et σ^2 donnés : test obsolète, on laisse tomber

on fera donc 6 tests disponibles sous R : SW, CSQ, LL, AD, CVM, DP

— shapiro.test()

— lillie.test()

— cvm.test()

— ad.test()

— dagoTest() nommé DP dans l'article

— pearson.test() nommé CSQ dans l'article

Les différents estimateurs $T(X_1, ..., X_n)$: voir les formules dans l'article

— SW : estime un coefficient de corrélation

— LL : estime la FDR par la FDR empirique (KS am ? ?lioré, plus besoin de fixer μ, σ^2 à l'avance)

— CVM :idem

— AD : idem

— CSQ : truc du χ^2 (suit une $\chi^2(10)$ sous $H0$)

— DP : suit une $\chi^2(2)$ sous $H0$

On va devoir appliquer les points 2) et 3) pour différentes valeurs de n, à ces 6 estimateurs. On aura ainsi toutes nos $RC(\alpha)$.

Ensuite appliquer le point 1) à des lois non-normales judicieusement choisies afin de mesurer la puissance.

2.5 Comment choisir nos lois alternatives pour mesurer la puissance des Tests ?

Prendre des Lois de distributions variées : symétriques, asymétriques, plus ou moins larges. . .

1) GLD : 4 paramètres, permet beaucoup de flexibilités, contrôle de la largeur, asymétrie etc. . .
2) $\mathcal{U}[0, 1]$
3) Trunc
4) LAPLACE
5) Gamma Γ
6) Beta β
7) Chi2 χ^2
8) Weibull
9) lognormale
10) etc. . . . .

Finalement, pour chaque Loi non-normale, on sera capable de tracer $1 - \beta = f(n)$ pour nos 6 tests. Puis comparer les résultats (certains tests seront meilleurs pour des loi symétriques, mais moins bien pour une asymétrique. . . etc)

2.5.1 La Loi GLD

2.5.1.1 Definition

The generalized lambda distribution (GLD) is a highly flexible four-parameter probability distribution function designed to approximate most of the well-known parametric distributions. With the proper choice of parameters, it can accurately approximate, normal, uniform, Student's t, exponential, lognormal, Weibull distributions, among others.

En gros c'est une Loi tout-en-un à 4 paramètres qui permet de retrouver plein de Loi usuelles

The GLD parametrizes the quantile function instead of PDF

More info

2.5.1.2 Code R :

programme divisé en 4 Scripts :

Le premier nommé Base.R est à exécuter avant tous les autres. Il contient les packages, les variables et les fonctions qui seront utilisés par les autres algorithmes. Le deuxième, RC.R est une boucle qui calcule les régions critiques des 6 tests pour 43 valeurs de n diff??rentes. Le temps de calcul est environ 30 min (penser à enregistrer son espace de travail afin de conserver toutes les matrices et ne pas recommencer le calcul à chaque fois) Le troisième : graph.R permet de tracer, pour $n = 100$, les FdR, les Fonctions quantiles et les densités sous $H0$ des 6 Tests de normalité. Ce script n???est pas obligatoire et n'apporte rien de plus qu'un support visuel. Le Dernier script : puissance.R, permet finalement de calculer puis tracer les puissances des tests pour différentes lois alternatives.

2.5.1.2.1 Base.R

```
install.packages("nortest")
install.packages("fBasics")
library("nortest")
library("fBasics")

alphalist=c(0.05,0.1) #vecteur contenant les valeurs de alpha
nlist=c(seq(20,200,5),seq(300,500,100),seq(1000,2000,500))#vecteur contenant les 43 valeurs de n , tail
```

Nos 6 tests de normalité Puisque les tests retournent une liste, on crée ces fonctions permettant de garder uniquement la valeur de la statistique de test :

```
shapiro.statistic=function(X){
  return(shapiro.test(X)$statistic)}

lillie.statistic=function(X){
  return(lillie.test(X)$statistic)}

cvm.statistic=function(X){
  return(cvm.test(X)$statistic)}

ad.statistic=function(X){
  return(ad.test(X)$statistic)}

pearson.statistic=function(X){
  return(pearson.test(X)$statistic)}

dagostino.statistic=function(X){
  return(dagoTest(X)@test$statistic[1] )}
```

2.5.1.2.2 RC.R

```
########### Calculs des r??gions critiques des 6 tests pour chaque valeurs de n

RC=rep(0,length(nlist))
```

```r
names(RC)=nlist

RC_SW=RC
RC_LL=RC
RC_CVM=RC
RC_AD=RC
RC_CSQ=RC
RC_DP=RC

N=50000
c=1
a=0.05
for (n in nlist){
  print(c) #permet de suivre l'avanc??e de la boucle en temps r??el
  Z=replicate(N, rnorm(n=n,mean=0,sd=1)) #chaque colonne de Z corrrespond ?? 1 ??chantillon de taille n


  #Vecteurs contenant chacun N VA simul??es selon la loi sous H0 de l'estimateur
  SW=apply(Z,2,shapiro.statistic)
  CVM=apply(Z,2,cvm.statistic)
  LL=apply(Z,2,lillie.statistic)
  AD=apply(Z,2,ad.statistic)
  CSQ=apply(Z,2,pearson.statistic)
  DP=apply(Z,2,dagostino.statistic)

  #-----------------------statistiques d'ordres--------------------------------
  SW_ordre=sort(SW, decreasing = FALSE)
  CVM_ordre=sort(CVM, decreasing = FALSE)
  LL_ordre=sort(LL, decreasing = FALSE)
  AD_ordre=sort(AD, decreasing = FALSE)
  CSQ_ordre=sort(CSQ, decreasing = FALSE)
  DP_ordre=sort(DP, decreasing = FALSE)

  #-----------------------quantiles empiriques-------------------------------------

  #left-tailed tests
  q_SW=SW_ordre[ceiling(N*a)]

  #right-tailed tests
  q_AD=AD_ordre[ceiling(N*(1-a))]
  q_LL=LL_ordre[ceiling(N*(1-a))]
  q_CVM=CVM_ordre[ceiling(N*(1-a))]
  q_DP=DP_ordre[ceiling(N*(1-a))]
  q_CSQ=CSQ_ordre[ceiling(N*(1-a))]

  RC_SW[c]=q_SW
  RC_LL[c]=q_LL
  RC_CVM[c]=q_CVM
  RC_AD[c]=q_AD
  RC_CSQ[c]=q_CSQ
  RC_DP[c]=q_DP

  c=c+1
```

```
}#fin boucle for
```

2.5.1.2.3 graph.R

```
N=50000
a=0.05
n=100
Z=replicate(N, rnorm(n=n,mean=0,sd=1))
#Vecteurs contenant chacun N VA simul??es selon la loi sous H0 de l'estimateur
SW=apply(Z,2,shapiro.statistic)
CVM=apply(Z,2,cvm.statistic)
LL=apply(Z,2,lillie.statistic)
AD=apply(Z,2,ad.statistic)
CSQ=apply(Z,2,pearson.statistic)
DP=apply(Z,2,dagostino.statistic)


#-----------------------statistiques d'ordres -----------------------------------------------
SW_ordre=sort(SW, decreasing = FALSE)
CVM_ordre=sort(CVM, decreasing = FALSE)
LL_ordre=sort(LL, decreasing = FALSE)
AD_ordre=sort(AD, decreasing = FALSE)
CSQ_ordre=sort(CSQ, decreasing = FALSE)
DP_ordre=sort(DP, decreasing = FALSE)

#----------------------- quantiles empiriques -----------------------------------------------

#left-tailed tests
q_SW=SW_ordre[ceiling(N*a)]

#right-tailed tests
q_AD=AD_ordre[ceiling(N*(1-a))]
q_LL=LL_ordre[ceiling(N*(1-a))]
q_CVM=CVM_ordre[ceiling(N*(1-a))]
q_DP=DP_ordre[ceiling(N*(1-a))]
q_CSQ=CSQ_ordre[ceiling(N*(1-a))]


#----------------------Fonctions Quantiles ---------------------------------------------------
par(mfrow=c(2,3))
plot(seq(1/N,1,1/N),SW_ordre, main="SW", type="l")
plot(seq(1/N,1,1/N),CVM_ordre,main="CVM", type="l")
plot(seq(1/N,1,1/N),LL_ordre,main="LL", type="l")
plot(seq(1/N,1,1/N),AD_ordre,main="AD", type="l")
plot(seq(1/N,1,1/N),CSQ_ordre,main="CSQ", type="l")
plot(seq(1/N,1,1/N),DP_ordre,main="DP", type="l")
mtext("Fonctions quantile empiriques des estimateurs sous H0 ", side = 3, line = -1.5, outer = T)
#--------------------Fonctions de r??partition empiriques----------------------------------------

FDR_SW=function(t){return(sum(SW<t)/N)}
FDR_LL=function(t){return(sum(LL<t)/N)}
FDR_CVM=function(t){return(sum(CVM<t)/N)}
FDR_AD=function(t){return(sum(AD<t)/N)}
```

```r
FDR_CSQ=function(t){return(sum(CSQ<t)/N)}
FDR_DP=function(t){return(sum(DP<t)/N)}

par(mfrow=c(2,3))
curve(Vectorize(FDR_SW)(x), main="SW",xlim = c(0,2))
curve(Vectorize(FDR_LL)(x),main="LL")
curve(Vectorize(FDR_CVM)(x),main="CVM")
curve(Vectorize(FDR_AD)(x),main="AD")
curve(Vectorize(FDR_CSQ)(x),main="CSQ",xlim = c(0,25))
curve(Vectorize(FDR_DP)(x),main="DP",xlim = c(0,11))

mtext("Fonctions de r??partition empiriques des estimateurs sous H0 ", side = 3, line = -1.5, outer = T)


#---------------------densit??s sous H0-----------------------------------------------------------
par(mfrow=c(2,3))
plot(density(SW), main="SW")
abline(v=q_SW, col="blue",lwd=2)
text(0.95,50,"Zone de rejet " , col = "blue",cex = 2 )
plot(density(CVM), main="CVM")
abline(v=q_CVM, col="blue",lwd=2)
text(0.3,10,"Zone de rejet " , col = "blue",cex = 2 )
plot(density(LL), main="LL")
abline(v=q_LL, col="blue",lwd=2)
text(0.13,16,"Zone de rejet " , col = "blue",cex = 2 )
plot(density(AD), main="AD")
abline(v=q_AD, col="blue",lwd=2)
text(1.7,1.7,"Zone de rejet " , col = "blue",cex = 2 )
plot(density(CSQ), main="CSQ")
curve(dchisq(x, df =10),
      type = 'l',
      lwd = 1,
      col = "red",
      add = T)
text(3,0.09,expression(chi^2*(10)) , col = "red",cex = 1.5 )
abline(v=q_CSQ, col="blue",lwd=2)
text(32,0.06,"Zone de rejet " , col = "blue",cex = 2 )
plot(density(DP), main="DP",xlim=c(0.1,10))
curve(dchisq(x, df =2),
      type = 'l',
      lwd = 1,
      col = "red",
      add = T)
text(3,0.2,expression(chi^2*(2)) , col = "red",cex = 1.5 )
abline(v=q_DP, col="blue",lwd=2)
text(8.5,0.30,"Zone de rejet " , col = "blue",cex = 2 )
mtext(expression("Lois des estimateurs sous H"[0]*" pour n=100"), side = 3, line = -1.5, outer = T)



##-------------LOIS NON NORMALES: exemple d'une loi sous H1---------------------------
```

```
Z=replicate(N, rnorm(n=100,mean=0,sd=1))
X=replicate(N,runif(100))

SW_X=apply(X,2,shapiro.statistic)

par(mfrow=c(1,1))
plot(density(SW_X),
     main=expression("Comparaison de la loi de l'estimateur SW sous H"[0]* " et sous H"[1]*" pour n=100
     xlim=c(0.9,1),
     ylim=c(0,80))
abline(v=RC_SW["100"], col="blue",lwd=2)
lines(density(SW), main="SW")

text(0.95, 30, expression(H[1]) )
text(0.99, 70, expression(H[0]))
text(0.94,70,"Zone de rejet " , col = "blue" ,cex = 2 )
```

2.5.1.2.4 puissance.R

```
###########################courbe puissance###############################
P=rep(0,length(nlist))
names(P)=nlist
P_SW=P
P_LL=P
P_CVM=P
P_AD=P
P_CSQ=P
P_DP=P

c=1
N2=10000
for (n in nlist) {
  print(c)
  Unif=replicate(N2,runif(n))

  SW_Unif=apply(Unif,2,shapiro.statistic)
  CVM_Unif=apply(Unif,2,cvm.statistic)
  LL_Unif=apply(Unif,2,lillie.statistic)
  AD_Unif=apply(Unif,2,ad.statistic)
  CSQ_Unif=apply(Unif,2,pearson.statistic)
  DP_Unif=apply(Unif,2,dagostino.statistic)

  ##left-tailed tests
  P_SW[c]=sum(SW_Unif<RC_SW[c])/N2
  #right-tailed tests
  P_LL[c]=sum(LL_Unif>RC_LL[c])/N2
  P_CVM[c]=sum(CVM_Unif>RC_CVM[c])/N2
  P_AD[c]=sum(AD_Unif>RC_AD[c])/N2
  P_CSQ[c]=sum(CSQ_Unif>RC_CSQ[c])/N2
  P_DP[c]=sum(DP_Unif>RC_DP[c])/N2

  c=c+1
```

```
}

#------------------------Graphiques puissance-----------------------------------------
par(mfrow=c(2,3))

plot(nlist,P_SW,log="x", type="o", col="blue", pch=15, lty=3 , xlab="n", ylab=expression(1-beta), main=
points(nlist,P_LL, col="brown",pch=16)
lines(nlist,P_LL ,col="brown", lty=3)
points(nlist,P_CVM, col="orange",pch=17)
lines(nlist,P_CVM ,col="orange", lty=3)
points(nlist,P_AD, col="black",pch=18)
lines(nlist,P_AD ,col="black", lty=3)
points(nlist,P_CSQ, col="dark green",pch=3)
lines(nlist,P_CSQ ,col="dark green", lty=3)
points(nlist,P_DP, col="dark red",pch=4)
lines(nlist,P_DP ,col="dark red", lty=3)
grid(lwd = 2)

legend(x="bottomright",legend=c("SW","LL","CVM","AD","CSQ","DP"), col=c("blue","brown","orange", "black
       pch=c(15:18,3:4), ncol=1, lty=3)
mtext("Courbes Puissance ", side = 3, line = -2, outer = T)
```

3 Conclusion