Projet fraudes covid

LUCEA Lenny, VALDEYRON Mathieu, CHERY Fanny

2022-05-26

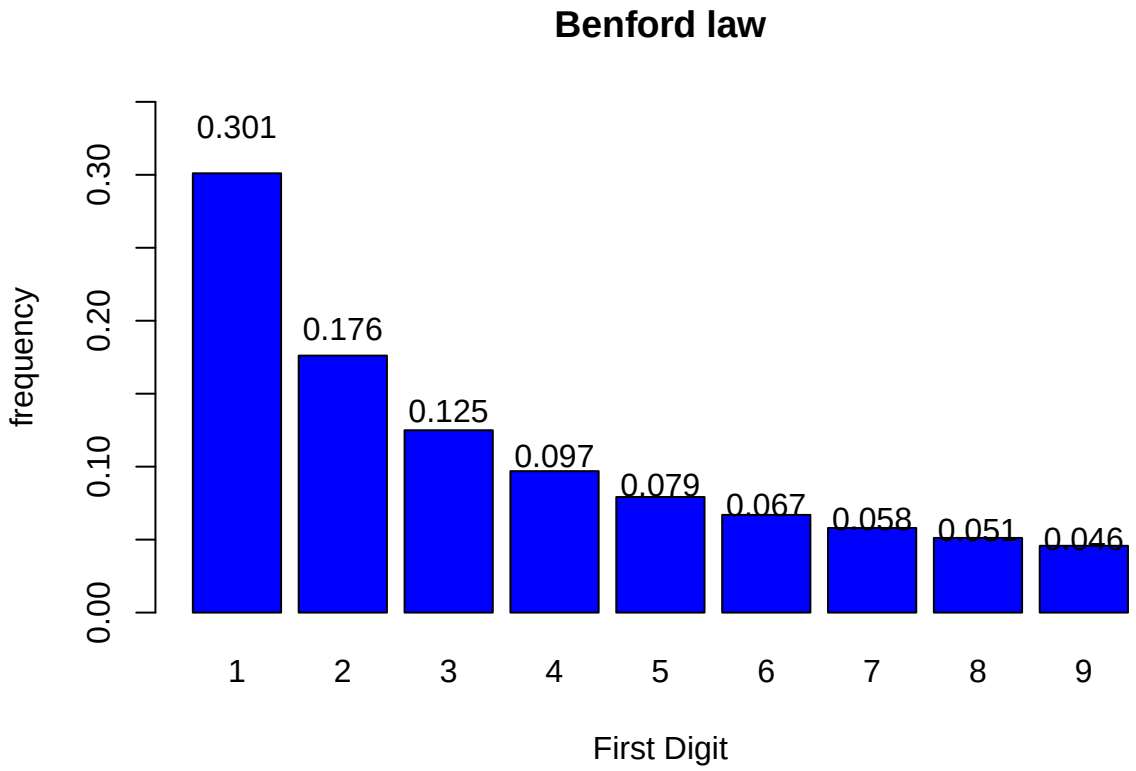
Sommaire

Explication du T_2

Comme observé précédemment il existe de nombreux tests d'adéquation pour la détection de fraudes via la loi de Newcomb-Benford. Par soucis de performance on décidera pour la suite d'appliquer le test du T_2 considéré comme l'un des plus puissants parmis les tests d'adéquations lisses pour la loi de Newcomb-Benford.

Avant de rentrer dans le vif du sujet nous allons donc tout d'abord faire un point sur les tests pour la loi de newcomb-benford.

Fanny ..



Passons alors à une brève introduction sur les testes lisses T_k.

Théorème Soit $X_1, \ldots, X_n$ des copies indépendantes d'une variable aléatoire X de densité $f(.)$ par rapport à une mesure v. Soit $\{h_0(.) := 1, h_k(.), k = 1, 2, ...\}$ une suite de fonctions orthonormales par rapport à $f(.)$; plus précisément, $\int h_k(x)h_{k'}(x)f(x)dv(x) = \delta_{kk'}$, la fonction delta de Kronecker. Soit $U_k = n^{-1/2}\sum_{i=1}^{n} h_k(X_i)$ et pour un entier $K \geq 1$, soir $T_k = \sum_{k=1}^{K} U_k^2$. Alors sous H_0, $T_K \xrightarrow{L} \chi^2_K$, la loi khi-deux à K degrés de liberté, et un test de niveau asymptotique α rejette H_0 si la valeur observée de T_K dépasse $x^2_{K,1-\alpha}$, le quantile d'ordre $1-\alpha$ de cette loi χ^2_K.

Explication des procédés d'analyse

Nous décidons alors dans un premier temps d'appliquer le test du T_2 sur la fréquence de distribution du premier chiffre significatif des nouveaux cas quotidiens de COVID-19 rapportés par 215 pays. Puis, dans un second temps nous procéderons à l'estimation des p-values "ajustées" à l'aide de la méthode de correction de Benjamini, Hochberg & Yekutieli.

En effet, lorsque des comparaisons multiples sont réalisées, le risque de se tromper n'est plus contrôlé.

Dans notre situation il semble donc nécessaire d'ajuster les p-values obtenues pour chacun des multiples tests, afin que le risque de se tromper pour l'ensemble de ces tests, soit à nouveau contrôlé, à niveau 5%.

Analyse des données observées (*p_value significative ou non*)

Après analyse, on observe qu'on rejette H_0 : *l'échantillon suit une loi de Newcomb-Benford contre* H_1 : *l'échantillon ne suis pas une loi de Newcomb-Benford* pour 33 pays. On remarque alors qu'on ne rejette pas H_0 pour les 182 autres pays avec un risque d'erreur $\alpha = 5\%$.

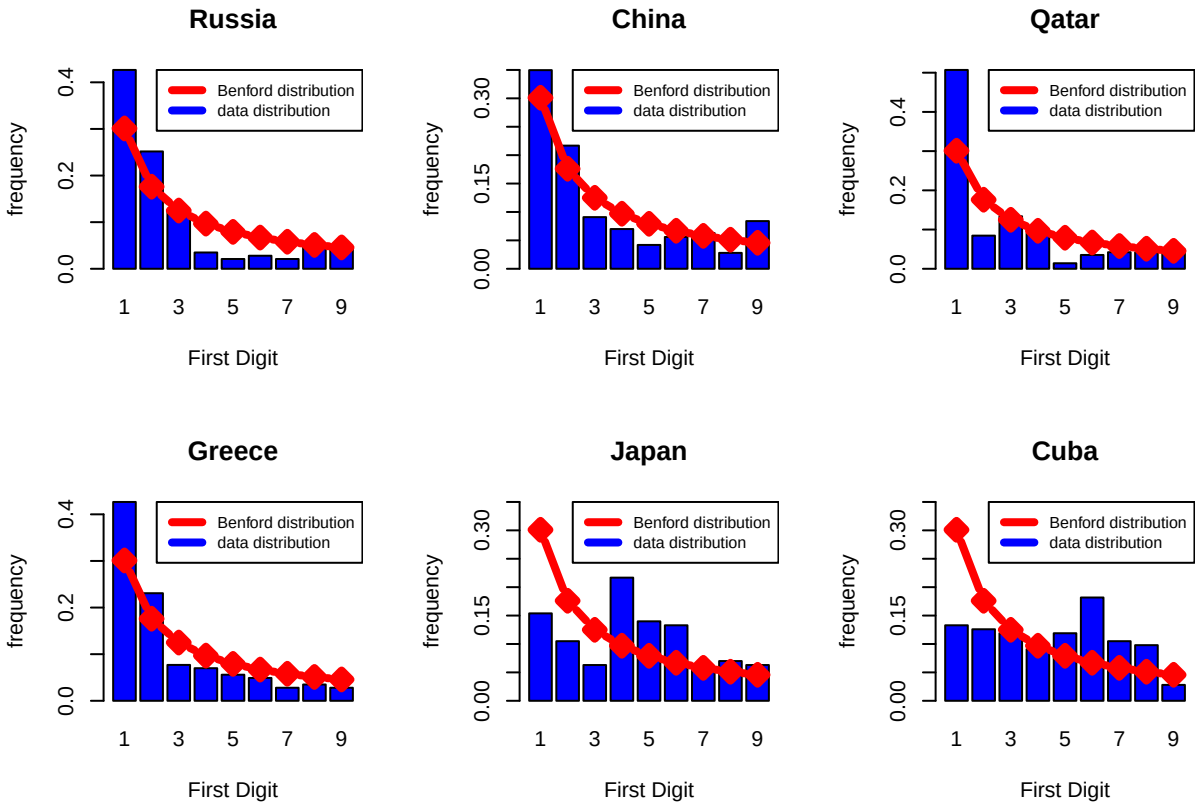
Cette étude nous permet de lever un drapeau d'alerte face aux pays qui ne passent pas le test.

Nous listerons ces pays dans le tableau ci-dessous :

Pays	T_2	P_value	Pays	T_2	P_value
Africa	14.371	0.032	Latvia	27.235	0
Asia	99.288	0	Libya	28.853	0
Australia	16.357	0.013	Malaysia	21.142	0
Austria	14.602	0.029	Moldova	16.864	0.01
Belarus	19.117	0.006	Morocco	17.035	0.01
Cook Islands	17.467	0.01	Myanmar	19.548	0.006
Cuba	39.368	0	Oceania	31.464	0
Cyprus	16.007	0.013	Poland	18.842	0.006
Djibouti	16.094	0.01	Qatar	22.296	0
Egypt	81.358	0	Russia	24.994	0
Estonia	18.337	0.006	Serbia	52.819	0
Gibraltar	15.726	0.017	Syria	14.114	0.035
Greece	16.425	0.013	Taiwan	18.496	0.006
Iran	24.989	0	Trinidad and Tobago	41.511	0
Ireland	21.69	0	United Kingdom	28.201	0
Japan	33.564	0	Vietnam	56.115	0
Kuwait	24.557	0			

Parmi eux on retrouve notamment Cuba, le Qatar, la Russie, le Japon, le Royaume-Uni ou encore Taiwan.

Ci-dessous nous décidons alors de représenter la distribution des premiers chiffres significatifs allant de 1 à 9 des données quotidiennes sur les nouveaux cas provenant des pays qui sont non conformes à la loi de Newcomb-Benford.



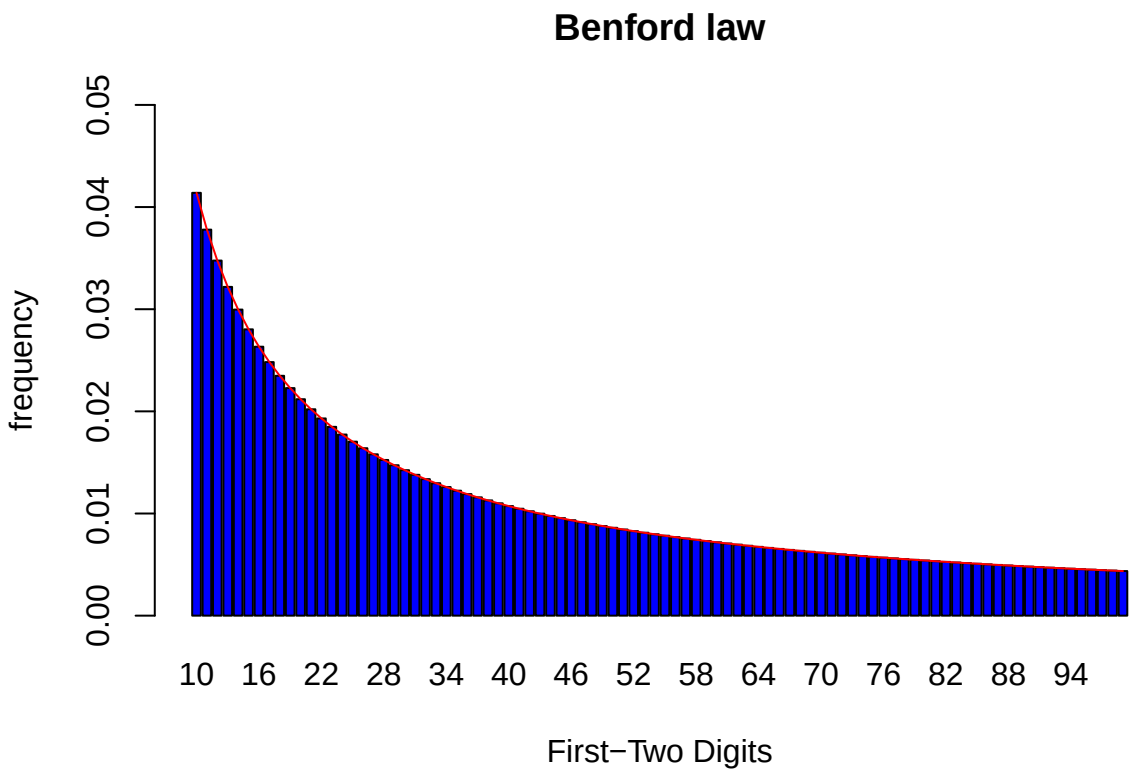
Nous observons effectivement des discordances non négligeables entre les répartitions théoriques des premiers chiffres siginificatifs suivant la loi de Newcomb-Benford et la répartition des premiers chiffres significatifs des échantillons observés. Ceci nous réconforte dans l'hypothèse de rejet de H_0.

Nous observons tout de même la présence de pays pour lesquels nous ne pouvons pas conclure.

Il serait intéressant de se questionner sur la véracité des données de nouveaux cas de COVID-19 des pays comme le Vatican ou encore Wallis et Futuna qui n'ont répertorié aucun cas depuis décembre 2021.

Block de deux

Après cette première analyse, nous décidons d'approfondir la recherche en analysant les deux premiers chiffres significatifs afin de repérer d'autres potentiels fraudeurs.



Nous obtenons alors parmi les individus

Pays	T_2	P_value	Pays	T_2	P_value
Africa	15.738	0.016	Latvia	29.333	0
Asia	96.327	0	Libya	27.72	0
Australia	17.671	0.007	Malaysia	21.719	0.002
Austria	13.305	0.044	Maldives	12.916	0.05
Belarus	13.935	0.034	Moldova	18.425	0.006
Cambodia	14.102	0.033	Montserrat	17.314	0.009
Comoros	13.558	0.04	Morocco	14.808	0.024
Cook Islands	18.139	0.006	Myanmar	16.104	0.014
Cuba	39.159	0	Oceania	34.985	0
Cyprus	20.681	0.003	Poland	19.698	0.004
Djibouti	31.834	0	Qatar	18.294	0.006
Egypt	72.478	0	Russia	20.552	0.003
Eritrea	17.751	0.007	Sao Tome and Principe	17.639	0.007
Estonia	17.796	0.007	Serbia	47.553	0
Falkland Islands	14.012	0.05	South Korea	14.889	0.023
Gibraltar	18.043	0.007	Sri Lanka	13.263	0.044
Guinea-Bissau	12.897	0.05	Taiwan	16.282	0.013
Iran	23.575	0.001	Trinidad and Tobago	41.847	0
Ireland	20.235	0.003	United Kingdom	27.431	0
Japan	30.688	0	Vietnam	45.682	0
Kuwait	27.21	0			

[1] Pays : nom des différents pays;

[2] T_2 : valeur de la statistique du T_2;

[3] p_value : p_value ajustée associée.

On remarque que pour nos deux analyses on retrouves 21 pays identiques dont la Russie, le Qatar ou encore le Royaume-Uni ce qui permet de confirmer le doute.

Par cette méthode on découvre également d'autre suspects que nous n'avions pas réussi à repérer précédemment qui sont :

- La République de Chypre
- L'Australie
- L'Érythrée
- Gibraltar
- São Tomé-et-Principe

```
#T_2 U^2
testopt <- function(df){
  w= 0
  x=twodigitst2(df,2)
```

```
  y=usq.benftest(vec(df),digits = 2)
  if(x[2]<=0.025|y$p.value<=0.025){
    w = 1
  }
  return(w)
}
```

```
# Rodriguez
library(BenfordTests)
w=10000
Q = seq(from=-5, to=5, length=30)
Q[length(Q)+1] = 0
Q[length(Q)+1] = -1

ER50 = matrix(0, nrow= 102, ncol= 3)
ER100 = matrix(0, nrow= 102, ncol= 3)
ER150 = matrix(0, nrow= 102, ncol= 3)

v = 1
for (n in c(50,100,150)) {

  C = matrix(0, nrow= 102, ncol= 3)

  z = 1
  for (gamma in Q) { # On parcout plusieurs gamma pour les différents tests.

    M = rep(0, 6) # Moyenne empirique = "puissance"
    L = matrix(0, nrow=6, ncol=w) # Liste des Y(l)

    proba = rep(0, 90) # vecteur des probas H1 : Rodriguez(gamma) pour sample
    for (i in c(1:90)) {

      if (gamma == -1) {
        proba[i] = log(1 + 1/(i+9))/log(10)
      }

      if (gamma == 0) {
        proba[i] = (1/810)*( 9+19*log(10)+9*(i+9)*log(i+9) - 9*(i+10)*log(i+10) )
      }

      if ( (gamma != 0) && (gamma != -1) ) {
        proba[i] = (gamma+1)/(90*gamma) - ( (i+10)^(gamma+1) - (i+9)^(gamma+1) )/( gamma*( 100^(gamma+1)
      }

    }

    for (l in c(1:w)) {

      data = rep(0, n)
      for (i in c(1:n)) { # Génération des Xi sous Rodriguez(gamma) de taille n

        data[i] = sample(x= c(10:99), size=1, prob= proba )

      }
```

```
      p = rep(0, 6) # Vecteur des p.values.

      p[1] = testopt(data)

      for (k in c(1:6)) {

        if (p[k] == 1 ) {
          M[k] = M[k] + 1
          L[k, l] = 1
        }

      }

      }
    M = M/w

    S = rep(0, 6) # Variance empirique
    for (l in c(1:w)) {
      for (k in c(1:6)) {
        S[k] = S[k] + (L[k,l] - M[k])^2
      }
    }
    S = S/w

    e = rep(0, 6)
    for (k in c(1:6)) {
      e[k] = 1.96*sqrt(S[k]/w) # erreur d'approximation (à 95%)
    }


    C[z, ] = c(M[1], e[1], gamma)
    z = z+1

    cat("\n", v/0.96, "%")
    v = v+1

  } # Fin de la boucle sur gamma.


  if (n == 50) {
    ER50 = C
  }
  if (n == 100) {
    ER100 = C
  }
  if (n == 150) {
    ER150 = C
  }

}
```

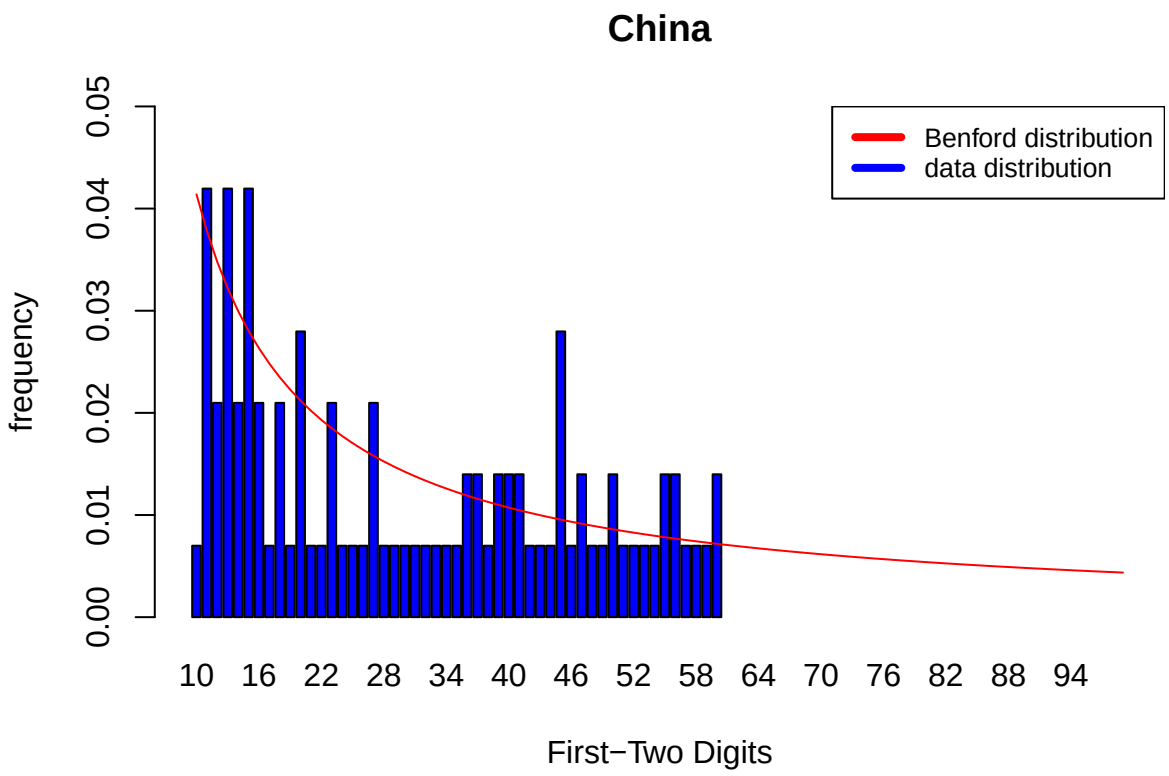
```
library(BenfordTests)
```

```
w = 10000
Q = seq(from=-5, to=5, length=30)
Q[length(Q)+1] = 0
Q[length(Q)+1] = 0

EP50 = matrix(0, nrow= 102, ncol= 3)
EP100 = matrix(0, nrow= 102, ncol= 3)
EP150 = matrix(0, nrow= 102, ncol= 3)

v = 1
for (n in c(50,100,150)) {

  C = matrix(0, nrow= 102, ncol= 3)

  z = 1
  for (gamma in Q) { # On parcout plusieurs gamma pour les différents tests.

    M = rep(0, 6) # Moyenne empirique = "puissance"
    L = matrix(0, nrow=6, ncol=w) # Liste des Y(l)

    proba = rep(0, 90) # vecteur des probas H1 : Pietronero(gamma) pour sample
    for (i in c(1:90)) {

      if (gamma == 0) {
        proba[i] = log(1 + 1/(i+9))/log(10)
      }

      if (gamma != 0) {
        proba[i] = ( (i+9)^(-gamma) - (i+10)^(-gamma) )/( 10^(-gamma) - 100^(-gamma) )
      }

    }

    for (l in c(1:w)) {

      data = rep(0, n)
      for (i in c(1:n)) { # Génération des Xi sous Pietronero(gamma) de taille n

        data[i] = sample(x= c(10:99), size=1, prob= proba )

      }

      p = rep(0, 6) # Vecteur des p.values.

      p[1] = testopt(data)

      for (k in c(1:6)) {

        if (p[k] == 1 ) {
          M[k] = M[k] + 1
          L[k, l] = 1
        }
```

```
      }

      }
    M = M/w

    S = rep(0, 6) # Variance empirique
    for (l in c(1:w)) {
      for (k in c(1:6)) {
        S[k] = S[k] + (L[k,l] - M[k])^2
      }
    }
    S = S/w

    e = rep(0, 6)
    for (k in c(1:6)) {
      e[k] = 1.96*sqrt(S[k]/w) # erreur d'approximation (à 95%)
    }


    C[z, ] = c(M[1], e[1], gamma)
    z = z+1

    cat("\n", v/0.96, "%")
    v = v+1

  } # Fin de la boucle sur gamma.


  if (n == 50) {
    EP50 = C
  }
  if (n == 100) {
    EP100 = C
  }
  if (n == 150) {
    EP150 = C
  }

}
```

Nous pouvons alors nous questionner sur la raison pour laquelle on ne rejette pas H_0 cette fois-ci.

Afin d'y voir plus claire rien de mieux qu'une modélisation graphique.



Annexe :

```
w = givenames(b)
pval = 0
test = 0
for(i in 1:length(w)){
bq <- b[b$location==w[i],]
T_k <- twodigitst(bq,2)
pval[i] <- T_k[2]
test[i] <- T_k[1]
}
pvaladj = p.adjust(pval,method="BY")
tab <- as.data.frame(matrix(c(w,test,pvaladj),ncol=3))
colnames(tab) = c("Pays","T_2","P_value")
benfo <- tab[as.numeric(tab$P_value)<=0.05,] # pays qui ne suivent pas une benford.
rownames(benfo) <- NULL
benfn <- tab[as.numeric(tab$P_value)>0.05,] # pays qui ne suivent pas une benford.
rownames(benfn) <- NULL
write.csv(benfn,"twdh0.csv")
write.csv(benfo,"twdh1.csv")
```

write.csv(tab,“twd.csv”)

```
#--------------------------- Test lisse first digit -------------------------
library(dplyr)
library(tidyr)
library(data.table)
library(BenfordSmoothTest)
library(BenfordTests)
library(kableExtra)

#----------------------- Importation des données ----------------------------
b <-read.csv("covid6.csv",sep=",")
b <- data.table(b)
b <- b[b$date > "2021-12-01",]
drop_na(b)
b <- b[b$location!="World",]
b <- b[b$location!="Europe",]
b <- b[b$location!="European Union",]
b <- b[b$location!="Lower middle income"]
b <- b[b$location!="High income"]

# on commence à partir de septembre 2021 on s'intéresses aux données récentes.

#------------------------------------------------------------------------------
#------------------------------ Fonction graphes ---------------------------
library(tidyr)
library(dplyr)
vec <- function(df){
  x <- c(df$new_cases)
  x <- x[x!=0]
  j=1
  for(i in 1:length(x)){
  if(is.na(x[j])){
    x <- x[-j]
  }
  else{
    j <- j +1
  }
  }
  return(x)
}
benf <- function(dd){
x <- vec(dd)
benlaw <- function(d) log10(1 + 1 / d)
digits <- 1:9
firstDigit <- function(x) substr(gsub('[0.]', '', x), 1, 1)
pctFirstDigit <- function(x) data.frame(table(firstDigit(x)) / length(x))
df <- pctFirstDigit(x)
if(length(firstDigit(x))<=10){
    return()
}
else{
baseBarplot <- barplot(df$Freq[1:9], names.arg = digits, xlab = "First Digit",ylab="frequency", ylim = c
lines(x = baseBarplot[,1], y = benlaw(digits), col = "red", lwd = 4,
```

```
      type = "b", pch = 23, cex = 1.5, bg = "red")
legend(x="topright", legend=c("Benford distribution","data distribution"),
       col=c("red","blue"), lty=1,lwd=4, cex=0.8)
}
}
#-------------------------------------------------------------------------

#------------- Fonction nom de pays pouvant passer le test -------------------
# Récupère les noms des pays ayant minimum 5 données
library(BenfordTests)
givenames <- function(df){
  name <- distinct(df,df$location)
  w <- 0
  x <- as.vector(name$`df$location`)
  t=0
  for(i in 1:length(x)){
    d <- df[df$location==x[i],]
    v <- vec(d)
    dx <- firstDigit(v)
    if(length(dx)<=10){
      t=t
    }
    else{
      t=t+1
      w[t]=x[i]
    }
  }
  return(w)
}
#-------------------------------------------------------------------------
#--------------------------- Test du T_2 first digit -------------------------
# fonction T_2
library(BenfordSmoothTest)
T2 <- function(df){
  x <- vec(df)
  benf <- BenfordSmooth.test(x)
  return(benf)
}
#-------------------------------------------------------------------------
#---------------------------- Test lisse bloc de deux ------------------------
Mu = rep(0,8)

for (k in c(1:8)) { # Calcul de Mu(k) :

  S = 0
  for (y in c(10:99)) {

    S = S + (y^k)*log(1 + 1/y)/log(10)

  }

  Mu[k] = S
}
```

```
for (k in c(1:4)) { # Calcul de M(k)^(-1) :

  A = matrix(1, ncol=k, nrow=k)

  for (i in c(0:(k-1)) ) {
    for (j in c(0:(k-1)) ) {

      if ( (i != 0) || (j != 0) ) {
        A[i+1, j+1] = Mu[i+j]
      }

    }
  }

  if (k == 1) { M1_ = solve(A) }
  if (k == 2) { M2_ = solve(A) }
  if (k == 3) { M3_ = solve(A) }
  if (k == 4) { M4_ = solve(A) }

}

M_ = list(M1_, M2_, M3_, M4_)

c_ = rep(0,4)

for (k in c(1:4)) { # Calcul de h(k) :

  Mu2 = matrix(0, nrow= k, ncol= 1)

  for (i in c(1:k)) {
    Mu2[i, 1] = Mu[ k-1+i ]
  }

  c_[k] = 1/(sqrt( Mu[2*k] - ( t(Mu2) %*% M_[[k]]  %*% Mu2 ) )) # 1/sqrt(c(k))

  A = -1*c_[k] * M_[[k]] %*% Mu2

  L = rep(0, k+1)
  for (i in c(1:k)) {
    L[i] = A[i]
  }
  L[k+1] = c_[k]

  print(L) # Les coéfficients de h(k)

}

# Fonctions h(k) :

h = function (k, x) {

  if (k == 1) {
    return( -1.54751771 + 0.04010177 *x )
```

```
  }

  if (k == 2) {
    return( 2.795867953 - 0.167441591 *x + 0.001736457 *x^2 )
  }

  if (k == 3) {
    return( -5.187810e+00  + 4.869054e-01 *x - 1.155519e-02 *x^2  + 7.630402e-05 *x^3 )
  }

  if (k == 4) {
    return( 9.725235e+00 - 1.237520e+00 *x + 4.769440e-02 *x^2 - 6.950207e-04 *x^3  + 3.371880e-06 *x^4
  }

}

# Statistique T(k) :       ( 1<= k <= 4)

T = function (k, data) {

  n = length(data)

  T_ = 0 # la statistique T(k)
  for (l in c(1:k)) {

    U = 0 # U(l)
    for (i in c(1:n)) {

      U = U + h(l, data[i])

    }
    U = U*(1/sqrt(n))

    T_ = T_ + U^2

  }

  return(T_)
}

pvalue = function(t, n, k, d, w) { # d et w pour Monte-Carlo

   if ( ( (n <= 100) || (d == 1) ) && (d != 2) ) { ### 1) approximation Monte-Carlo ###

     M = 0 # Moyenne empirique = p.value
     L = rep(0, w) # Liste des Y(l)

     proba = rep(0, 90) # vecteur des probas sous H0 pour sample
     for (i in c(1:90)) {

       proba[i] = log(1 + 1/(i+9) )/log(10)

     }
```

```
    for (l in c(1:w)) {

      data = rep(0, n)
      for (i in c(1:n)) { # Génération des Xi sous H0 de taille n

        data[i] = sample(x= c(10:99), size=1, prob= proba )

      }

      a = T(k, data)

      if (a >= t) {
        M = M+1
        L[l] = 1
      }

    }
    M = M/w

    S = 0 # Variance empirique
    for (l in c(1:w)) {

      S = S + (L[l] - M)^2

    }
    S = S/w

    e = 1.96*sqrt(S/w) # erreur d'approximation (à 95%)


    return( c(M, e) )


  } else { ### 2) approximation Khi2 ###

    M = 1 - pchisq(t, df=k)

    return( c(M, 0) )

  }

}

# 1 <= k <= 4.
# data : sous forme de vecteur c(...).
# d = 1 : force le calcul de la p.value par Monte-Carlo.
# d = 2 : force le calcul de la p.value par approximation Khi2
# Si n <= 100 alors d=1 par défault, et si n > 100 alors d=2 par défault.
# w = taille de l'échantillon pour Monte-Carlo.
# texte = FALSE : n'affiche pas de texte, et return c(Tk, p.value, erreur M-C).

Test.lisse = function (k, data, d=0, w=10000, texte=TRUE) {
```

```
  t = T(k, data) # Valeur de la statistique T(k)

  n = length(data)
  A = pvalue(t, n, k, d, w) # On récupère la p.value associé

  if ( texte == TRUE ) {

    cat("\n")
    cat("k =", k, "\n")
    cat("Statistique Tk =", t, "\n \n")

    if ( ( (n <= 100) || (d == 1) ) && (d != 2) ) {
      cat("Approximation de la p.value par Monte-Carlo :", "\n \n")
      cat("--> p.value =", A[1], "\n")
      cat("Erreur d'approximation en valeur absolue (à 95%) <=", A[2])

    } else {
      cat("Approximation asymptotique de la p.value par une Khi2(k) :", "\n \n")
      cat("--> p.value =", A[1])
    }
    cat("\n")

  } else {

    return( c(t,A[1], A[2]) )

  }

}
#----------------------------- Test appliqué aux données --------------------
twodigitst <- function(df,k){
  x<-vec(df)
  w <-signifd(x = x, digits = 2)
  if(length(w)<=50){
    y<-Test.lisse(k,w,texte=F,d=1)
  }
  else{
    y<-Test.lisse(k,w,texte=F,d=2)
  }

  return(y)
}
twodigitst2 <- function(df,k){
  x<-vec(df)
  w <-signifd(x = x, digits = 2)
  y<-Test.lisse(k,w,texte=F,d=2)

  return(y)
}
#-------------------------------------------------------------------------------
#----------------------------- Fonction p_adjust ---------------------------------
p_adjusted <- function(w,k){
```

```
  n <- givenames(w)
  pval = 0
  for( i in 1:length(n)){

    bq <- w[w$location==n[i],]
    Tk = twodigitst(bq,k)
    pval[i]= Tk[2]
  }
  padj <- p.adjust(pval,method="holm")
  return(padj)
}
#-------------------------------------------------------------------------------------------
#-----visualisation graphique de la distribution comparé à benford-------
name <- distinct(b,b$location)
x <- as.vector(name$`b$location`)
for(i in 1:length(x)){
  bq <- b[b$location==x[i],]
  benf(bq)
}
#-------------------------------------------------------------------------
#--------------------------- Creation tableau first digit ---------------
w = givenames(b)
pval = 0
test = 0
for(i in 1:length(w)){
  bq <- b[b$location==w[i],]
  T_k <- T2(bq)
  pval[i] <- as.numeric(T_k[3,2])
  test[i] <- as.numeric(T_k[2,2])
}
pvaladj = p.adjust(pval,method="BY")
tab <- as.data.frame(matrix(c(w,round(test,3),round(pvaladj,3)),ncol=3))
colnames(tab) = c("Pays","T_2","P_value")
benfo <- tab[as.numeric(tab$P_value)<=0.05,] # pays qui ne suivent pas une benford.
rownames(benfo) <- NULL
benfn <- tab[as.numeric(tab$P_value)>0.05,] # pays qui ne suivent pas une benford.
rownames(benfn) <- NULL
write.csv(benfn,"h0.csv")
write.csv(benfo,"h1.csv")
write.csv(tab,"tabadj.csv")
#-----------------------------------------------------
#---------------- tableau two digits -----------------
w = givenames(b)
pval = 0
test = 0
for(i in 1:length(w)){
  bq <- b[b$location==w[i],]
  T_k <- twodigitst(bq,2)
  pval[i] <- T_k[2]
  test[i] <- T_k[1]
}
pvaladj = p.adjust(pval,method="BY")
tab <- as.data.frame(matrix(c(w,round(test,3),round(pvaladj,3)),ncol=3))
```

```
colnames(tab) = c("Pays","T_2","P_value")
benfo <- tab[as.numeric(tab$P_value)<=0.05,] # pays qui ne suivent pas une benford.
rownames(benfo) <- NULL
benfn <- tab[as.numeric(tab$P_value)>0.05,] # pays qui ne suivent pas une benford.
rownames(benfn) <- NULL
write.csv(benfn,"twdh0.csv")
write.csv(benfo,"twdh1.csv")
write.csv(tab,"twd.csv")

#---------------------- Fonction new -------------------------------
```

Sources :

- <https://github.com/CSSEGISandData/COVID-19>