Statistique décisionnelles - TP1

Charles Vin

Exercice 1

Question 1

Donnée

Soit $(X_1, ..., X_n)$ iid. de vecteur loi p.

Soit p_n vecteur loi d'une $Unif([0, 1.5])$.

On découpera l'intervale en 5 intervalles uniformes de longueur 0.3 (de la forme $[a; b[$)

Calculons $\bar{p}_n$ en comptant les valeur dans chaques intervales

```
valeur = c(0.02,0.02,0.07,0.07,0.07,0.09,0.11,0.13,0.16,0.16, 0.2,0.2,0.21,0.25,0.29,0.32,0.35,0.35,0.37
n = 50

# Création de la table et comptage des valeurs dans l'intervale
interval_table = data.frame(
  table(
    cut(valeur,
        right = FALSE, # [a,b[
        breaks=seq(0, 1.5, length.out = 6),
        labels=c("[0;0.3[","[0.3 ; 0.6[","[0.6 ; 0.9[","[0.9;1.2[","[1.2;1.5[")
      )
  )
)
colnames(interval_table)<-c("Intervale","Freq") # On change le nom des colonnes.

p_n_bar = interval_table$Freq # On prépare p_n_bar
interval_table
```

```
##     Intervale Freq
## 1     [0;0.3[   15
## 2 [0.3 ; 0.6[   14
## 3 [0.6 ; 0.9[   13
## 4   [0.9;1.2[    5
## 5   [1.2;1.5[    3
```

Générons p_n indiquant les probabilité d'une loi $Unif([0; 1.5])$ d'être dans chaque intervale.

```
# On initialise les intervales
intervals = list()
p_n = c()
a = c(0, 0.3, 0.6, 0.9, 1.2)
b = c(0.3, 0.6, 0.9, 1.2, 1.5)

# Définition des intervales et calcul de p_n Uniforme
```

```r
# Pas trouver de méthode sans boucle for :(
for (i in 1:length(a)) {
  intervals[[i]]= c(a[i], b[i])
  p_n = c(p_n,
          punif(intervals[[i]][2], max = 1.5) - punif(intervals[[i]][1], max = 1.5))
}
p_n
```

```
## [1] 0.2 0.2 0.2 0.2 0.2
```

Ajoutons le à la data frame `interval_table` avec également les $n * p_{n,i}$:

```r
interval_table["p_n"] = p_n
interval_table["n*p_i"] = p_n*n
interval_table
```

```
##     Intervale Freq p_n n*p_i
## 1     [0;0.3[   15 0.2    10
## 2 [0.3 ; 0.6[   14 0.2    10
## 3 [0.6 ; 0.9[   13 0.2    10
## 4   [0.9;1.2[    5 0.2    10
## 5   [1.2;1.5[    3 0.2    10
```

Condition

- On a regroupé les valeurs pour créer des classes et tester l'adéquation à une **loi discrète**
- $n = 50$ et les $np_{n,i} \geq 5$

Les conditions du test du $\mathcal{X}^2$ d'adéquation sont remplis.

Hypothèse

- $H_0 = p = p_n$: Les X_i suivent loi Uniforme sur $[0; 1]$
- $H_0 = p \neq p_n$: Les X_i ne suivent pas une loi Uniforme sur $[0; 1]$

Test

```r
chisq.test(p_n_bar, p=p_n)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  p_n_bar
## X-squared = 12.4, df = 4, p-value = 0.01461
```

On retrouve :

- La statistique de test $D(p_n, \bar{p_n}) = \sum_{k=0}^{5} \frac{(N_{n,k} - n * p_{n,k})^2}{n * p_{n,k}} = 12.4$
- Et la $p - valeur = 0.01461$

Conclusion

Au vu de la p-valeur, on rejette H_0. Les X_i ne suivent pas une loi Uniforme sur $[0; 1.5$.

Question 2

Donnée

Soit $(X_1, ..., X_n)$ iid. de vecteur loi p.

Soit Y une variable aléatoire de densité $f(x) = \cos(x)\not\Vdash_{x\in[0;pi/2]}$. Soit p_n vecteur loi de Y sur les intervales précédents.

On a donc :

$$P(a < Y < b) = \int_a^b cos(x)\not\Vdash_{[a,b]\subset[0;pi/2]}dx = sin(b) - sin(a)\not\Vdash_{[a,b]\subset[0;pi/2]}$$

Dans notre cas tous nos intervales sont inclus dans $[0, pi/2]$

Calculons p_n avec cette nouvelle densité :

```
p_n = c()
for (i in 1:length(a)) {
  p_n = c(p_n,
          sin(intervals[[i]][2]) - sin(intervals[[i]][1])
        )
}
interval_table["p_n"] = p_n
interval_table["n*p_i"] = p_n*n
interval_table
```

```
##     Intervale Freq       p_n     n*p_i
## 1     [0;0.3[   15 0.2955202 14.776010
## 2 [0.3 ; 0.6[   14 0.2691223 13.456113
## 3 [0.6 ; 0.9[   13 0.2186844 10.934222
## 4   [0.9;1.2[    5 0.1487122  7.435609
## 5   [1.2;1.5[    3 0.0654559  3.272795
```

Condition

On remarque que les $np_{n,i}$ ne sont pas tous supérieur à 5. Fusionnons donc les deux dernières classes en reprenant les étapes d'avant.

```
# Création de la table et comptage des valeurs dans l'intervale
interval_table = data.frame(
  table(
    cut(valeur,
        right = FALSE,
        breaks=c(0,0.3, 0.6,0.9,1.5),
        labels=c("[0;0.3[","[0.3 ; 0.6[","[0.6 ; 0.9[","[0.9;1.5[")
      )
  )
)
colnames(interval_table)<-c("Intervale","Freq") # On change le nom des colonnes.

p_n_bar = interval_table$Freq # On prépare p_n_bar
interval_table
```

```
##     Intervale Freq
## 1     [0;0.3[   15
## 2 [0.3 ; 0.6[   14
## 3 [0.6 ; 0.9[   13
## 4   [0.9;1.5[    8
```

```r
intervals = list()
p_n = c()
a = c(0, 0.3, 0.6, 0.9)
b = c(0.3, 0.6, 0.9, 1.5)

# Redéfinition des intervales
for (i in 1:length(a)) {
  intervals[[i]]= c(a[i], b[i])
}

# Calcul des p_n de chaque intervale
for (i in 1:(length(a)-1)) {
  p_n = c(p_n,
          sin(intervals[[i]][2]) - sin(intervals[[i]][1])
        )
}
p_n = c(p_n, 1-sum(p_n)) # En veillant à ce que la somme fasse 1

#On remet le tout dans notre tableau final
interval_table["p_n"] = p_n
interval_table["n*p_i"] = p_n*n
interval_table
```

```
##     Intervale Freq       p_n    n*p_i
## 1     [0;0.3[   15 0.2955202 14.77601
## 2 [0.3 ; 0.6[   14 0.2691223 13.45611
## 3 [0.6 ; 0.9[   13 0.2186844 10.93422
## 4   [0.9;1.5[    8 0.2166731 10.83365
```

Nos conditions sont maintenant remplis !

Hypothèse

- $H_0 = p = p_n$: la loi des X_i est la même que celle des Y_i
- $H_0 = p \neq p_n$: la loi des X_i n'est pas la même que celle des Y_i

Test

```r
chisq.test(p_n_bar, p=p_n)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  p_n_bar
## X-squared = 1.1568, df = 3, p-value = 0.7634
```

On retrouve :

- La statistique de test $D(p_n, \bar{p_n}) = \sum_{k=0}^{5} \frac{(N_{n,k} - n*p_{n,k})^2}{n*p_{n,k}} = 1.15$
- Et la $p - valeur = 0.7634$

Conclusion

Au vu de la p-valeur, on conserve H_0. Les X_i semble suivre une densité $f(x) = cos(x)\mathbb{1}_{x \in [0,pi/2]}$.

Exercice 2

Enoncé (+code)

Un jardinier souhaite évaluer efficacité de la mise en place d'un paillage. Cette technique consiste à mettre de la paille au niveau de chaque pied pour mieux retenir l'humidité. On suppose que cela aurait une influence sur le poids des tomates.

Pour cela, le jardinier a mis du paillage uniquement d'un côté de sa serre. Le tout en plantant les semis d'un même lots et en arrosant uniformément des deux côtés.

Voici le poids de chaque tomate, récolté **sans** paillage (arrondie à l'unité)

```
# echo=FALSE <- permet de cacher le code pour obtenir un énoncé propre.
n_1=20
m = 200
sigma = 5

ech = round(rnorm(n_1, m, sigma), 1)
X_n_bar = round(mean(ech),2)
V_n = round(var(ech),2)

cat(ech)
```

```
## 191.5 195.6 197.9 206.4 200.8 202 198.9 201.6 207.4 200.8 203.5 204.6 202.2 202.2 198.6 192.1 198.3 1
```

```
cat(paste("\nIl y a eu", n_1, "tomates"))
```

```
##
## Il y a eu 20 tomates
```

```
cat(paste("\nLa moyenne vaut :", X_n_bar))
```

```
##
## La moyenne vaut : 200.5
```

```
cat(paste("\nLa variance vaut :", V_n))
```

```
##
## La variance vaut : 17.85
```

Voici le poids de chaque tomate, récolté **avec** paillage (arrondie à l'unité)

```
# echo=FALSE <- permet de cacher le code pour obtenir un énoncé propre.
n_2 = 22
ech_paillage = round(rnorm(n_2, m+10, sigma), 1)
X_n_bar_paillage = round(mean(ech_paillage),2)
V_n_paillage = round(var(ech_paillage),2)

cat(ech_paillage)
```

```
## 194.3 202.9 208.4 216.8 204 206.9 204.7 207.7 206.6 201.4 209.1 207.4 209 204.2 200.3 207.6 204.5 21
```

```
cat(paste("\nIl y a eu", n_2, "tomates"))
```

```
##
## Il y a eu 22 tomates
```

```
cat(paste("\nLa moyenne vaut :", X_n_bar_paillage))
```

```
##
```

```
## La moyenne vaut : 206.95
```

```r
cat(paste("\nLa variance vaut :", V_n_paillage))
```

```
##
## La variance vaut : 35.54
```

1) D'après sa balance/caisse enregistreuse, le poids des tomates récolté lors des années précédentes sans paillage suit une loi normale de moyenne $m = 200$ et de variance $\sigma^2 = 25$. Vérifier que c'est toujours le cas cette année avec $\alpha = 5\%$ de se tromper.

Pour vous aider, voici la table des valeurs de la fonction de répartition d'une $\mathcal{N}(200, 5^2)$ pour les valeurs de l'échantillon.

```r
# echo=FALSE <- permet de cacher le code pour obtenir un énoncé propre.
data.frame(
  t = sort(ech),
  "F_X(t)" = pnorm(sort(ech), 200, 5)
)
```

```
##        t     F_X.t.
## 1  191.5 0.04456546
## 2  192.1 0.05705343
## 3  195.6 0.18942965
## 4  197.9 0.33724273
## 5  198.0 0.34457826
## 6  198.3 0.36692826
## 7  198.6 0.38973875
## 8  198.9 0.41293558
## 9  200.8 0.56355946
## 10 200.8 0.56355946
## 11 201.6 0.62551583
## 12 202.0 0.65542174
## 13 202.2 0.67003145
## 14 202.2 0.67003145
## 15 203.5 0.75803635
## 16 203.5 0.75803635
## 17 204.1 0.79389195
## 18 204.6 0.82121362
## 19 206.4 0.89972743
## 20 207.4 0.93056338
```

2) Supposons maintenant que la moyenne sur les années précédentes n'est pas accessible à cause de l'écran cassé de la balance/caisse enregistreuse. En supposant que le paillage n'est pas changer la variance, tester au niveau $\alpha = 5\%$ si celui-ci a augmenté la moyenne.

Correction

Question 1

Effectuons un test d'adéquation de Kolmogorov-Smirnov.

Donnée Soit $(X_1, ..., X_n)$ iid. de fonction de répartition F_X inconnu.

Soit F_0 la fonction de répartition d'une $\mathcal{N}(200, 40)$.

Conditions

- Les X_i viennent d'une loi continue. Les valeurs répétées viennent des arrondis.
- n est petit, on utilisera la version non asymptotique.

Hypothèse

- $H_0 : F_X = F_0$ Le poids des tomates récoltés sans paillage cette année suit la même loi que les années précédentes
- $H_1 : F_X \neq F_0$ Le poids des tomates récoltés sans paillage cette année ne suit pas la même loi que les années précédentes

Statistique de test

$$D = \sup_{t\in\mathbb{R}}\{\frac{1}{n}\sum_{i=1}^{n} 1_{X_i\leq t} - F_0(t)\} = \max_{1\leq i\leq n}(\max(\left|\frac{i}{n} - F_0(t)\right|, \left|\frac{i-1}{n} - F_0(t))\right|))$$

Zone de rejet

- Sous $H_0 : D \to 0$
- Sous $H_1 : D \to h$

$$\mathcal{R} = \{D \geq 0.2941\}$$

Calcul de la statistique de test

```
ks.test(ech, pnorm, m, sigma)
```

```
## Warning in ks.test(ech, pnorm, m, sigma): ties should not be present for the
## Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  ech
## D = 0.18724, p-value = 0.4847
## alternative hypothesis: two-sided
```

On obtient une statistique de test $D \notin \mathcal{R}$. On conserve H_0, la poids des tomates suit la même loi que les années précédentes.

Question 2

Nous allons effectuer un test de Student à 2 échantillons gaussien de variance égale

Données

- Soit $X_1, ..., X_{n_1}$ variables aléatoires iid. $\mathcal{N}(m_1, 5^2)$ représentant le poids de la ième tomate récoltée sans paillage
- Soit $Y_1, ..., Y_{n_2}$ variables aléatoires iid. $\mathcal{N}(m_2, 5^2)$ représentant le poids de la ième tomate récoltée avec paillage
- Les échantillons sont indépendants.

Hypothèse

- $H_0 : m_1 = m_2$ Le poids des tomates récoltés sans paillage à même loi que celle récoltée avec paillage. Le paillage n'a aucun effet sur le poids moyen des tomates

- $H_1 : m_1 < m_2$ Le paillages augmente le poids moyen des tomates

Statistique de test

$$D = \frac{1}{\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}\frac{\bar{X}_{n_1}-\bar{Y}_{n_2}}{\sqrt{W}}$$

Avec $W = \frac{(n_1-1)V^X_{n_1}+(n_2-1)V^Y_{n_2}}{n_1+n_2-2}$

Zone de rejet

- Sous $H_0 : m_1 = m_2$

$$D \sim \mathcal{T}(n_1+n_2-2)$$

- Sous $H_1 : m_1 < m_2, D$ prend des valeurs plus petites que sous H_0.

$$\mathcal{R} = \{D < h_\alpha\}$$

  avec h_α quantile d'ordre alpha d'une $\mathcal{T}(n_1+n_2-2)$

Calcul de la statistique de test Dans notre cas :

```
W = (1/(n_1 + n_2 - 2))*((n_1 - 1)*V_n + (n_2 -1) * V_n_paillage)
 str = paste(
   "W = \\frac{(",n_1," - 1)",
   V_n,
   " + (",n_2,"- 1)",
   V_n_paillage,
   "}{",n_1," + ",n_2," - 2} = ",
   W
 )
writeLines(c("$$", str, "$$"))
```

$$W = \frac{(20-1)17.85 + (22-1)35.54}{20+22-2} = 27.13725$$

Et :

```
D = ((1)/(sqrt(1/n_1 + 1/n_2))) * ((X_n_bar - X_n_bar_paillage)/sqrt(W))

 str = paste("  D =
          \\frac{1}{\\sqrt[]{\\frac{1}{",n_1,"} + \\frac{1}{",n_2,"}}}
          \\frac{",X_n_bar," - ",X_n_bar_paillage,"}{\\sqrt[]{",W,"}} ="
  , round(D,4)
)
writeLines(c("$$", str, "$$"))
```

$$D = \frac{1}{\sqrt{\frac{1}{20}+\frac{1}{22}}}\frac{200.5-206.95}{\sqrt{27.13725}} = -4.0075$$

```
t.test(ech, ech_paillage, alternative = "greater", var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  ech and ech_paillage
## t = -4.0102, df = 40, p-value = 0.9999
## alternative hypothesis: true difference in means is greater than 0
```

```
## 95 percent confidence interval:
##  -9.164773       Inf
## sample estimates:
## mean of x mean of y
##  200.5000  206.9545
```

On rejette fortement H_0, le paillage a augmenter la moyenne du poids des tomates récoltées.

(Les différences de valeurs entre le calcul à la main et fait par R viennent des arrondies sur les moyennes et variances)

Exercice 3

Définissons la fonction qui estime un quantile à partir d'une proba et d'un échantillon.

```r
Q = function(n, p, ech) {
  return(ech[as.integer(p*n)])
}
```

Question 1

Pour $n = 1000$ il faut regarder

$$Q_n(0.95) = X_{(n*0.95)} = X_{(950)} Q_n(0.975) = X_{(n*0.975)} = X_{(975)}$$

Question 2

On a

$$Q_n(p) \pm \frac{1}{\sqrt{n}}$$

Pour garantir les deux premières décimale soit fixe il faut que l'erreur soit plus petite que 0.01

$$\frac{1}{\sqrt{n}} < 0.01 \Leftrightarrow \frac{1}{0.01} > \sqrt{n} \Leftrightarrow 100 > \sqrt{n} \Leftrightarrow 10000 > n$$

On utilisera finalement un minimum de 10 000 valeurs pour estimer nos quantiles.

Question 3

```r
n = 10000
sorted_echantillon = sort(rnorm(n, 0, 1)) # Tirage de n loi normale + tri dans l'ordre croissant
p = seq(0.8, 0.95, 0.01)
df = data.frame(
  proba = p,
  estimated_quantile = Q(n, p, sorted_echantillon),
  real_quantile = qnorm(p)
)
df["error"] = abs(df$estimated_quantile - df$real_quantile)
df
```

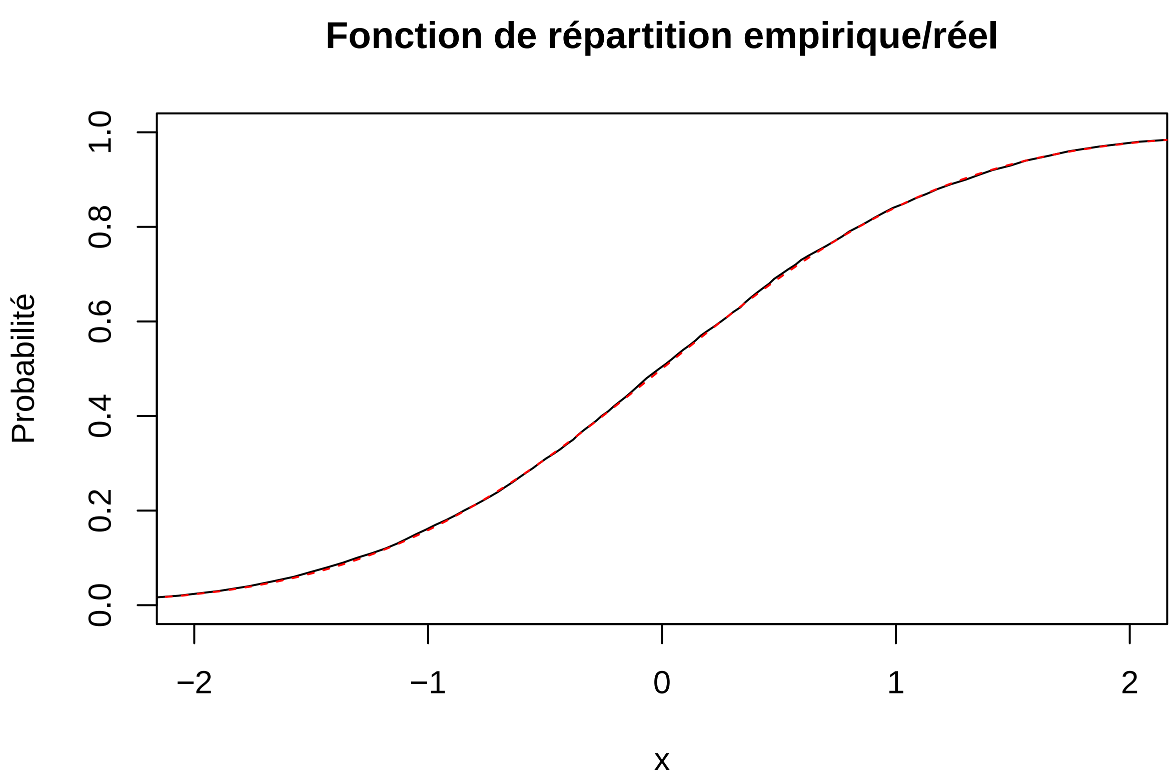
```
##    proba estimated_quantile real_quantile        error
## 1   0.80          0.8390108     0.8416212 0.0026104174
## 2   0.81          0.8761447     0.8778963 0.0017515951
## 3   0.82          0.9108719     0.9153651 0.0044931438
## 4   0.83          0.9482933     0.9541653 0.0058719780
## 5   0.84          0.9869496     0.9944579 0.0075083326
```

```
## 6   0.85          1.0391274     1.0364334 0.0026939954
## 7   0.86          1.0818814     1.0803193 0.0015620580
## 8   0.87          1.1325993     1.1263911 0.0062081294
## 9   0.88          1.1781992     1.1749868 0.0032124132
## 10  0.89          1.2342232     1.2265281 0.0076951163
## 11  0.90          1.2994198     1.2815516 0.0178682089
## 12  0.91          1.3548691     1.3407550 0.0141140522
## 13  0.92          1.4123414     1.4050716 0.0072697964
## 14  0.93          1.4920802     1.4757910 0.0162892195
## 15  0.94          1.5548872     1.5547736 0.0001135632
## 16  0.95          1.6510158     1.6448536 0.0061621831
```

Les erreurs sont en effet plus petite que 10^{-2}.

Testons $\forall p \in [0, 1]$ et traçons les deux fonctiondes de répartiton. En rouge la vrais fonction de répartition, en noir l'empirique.

```r
p = seq(0.01, 0.99, 0.01) #Cette fois ci pour tout p en évitant les frontière qui provoque des erreurs
df = data.frame(
  proba = p,
  estimated_quantile = Q(n, p, sorted_echantillon),
  real_quantile = qnorm(p, 0, 1)
)
plot(df$estimated_quantile,df$proba, xlab="x", ylab="Probabilité", type="l", xlim=c(-2,2), ylim=c(0,1))
title("Fonction de répartition empirique/réel")
lines(df$real_quantile, df$proba, col="red", lty=2)
```



Les deux fonctions de répartiton se superpose !

Quelle valeur de n faut-il prendre pour obtenir une précision de 10^-2 ?

Question 4

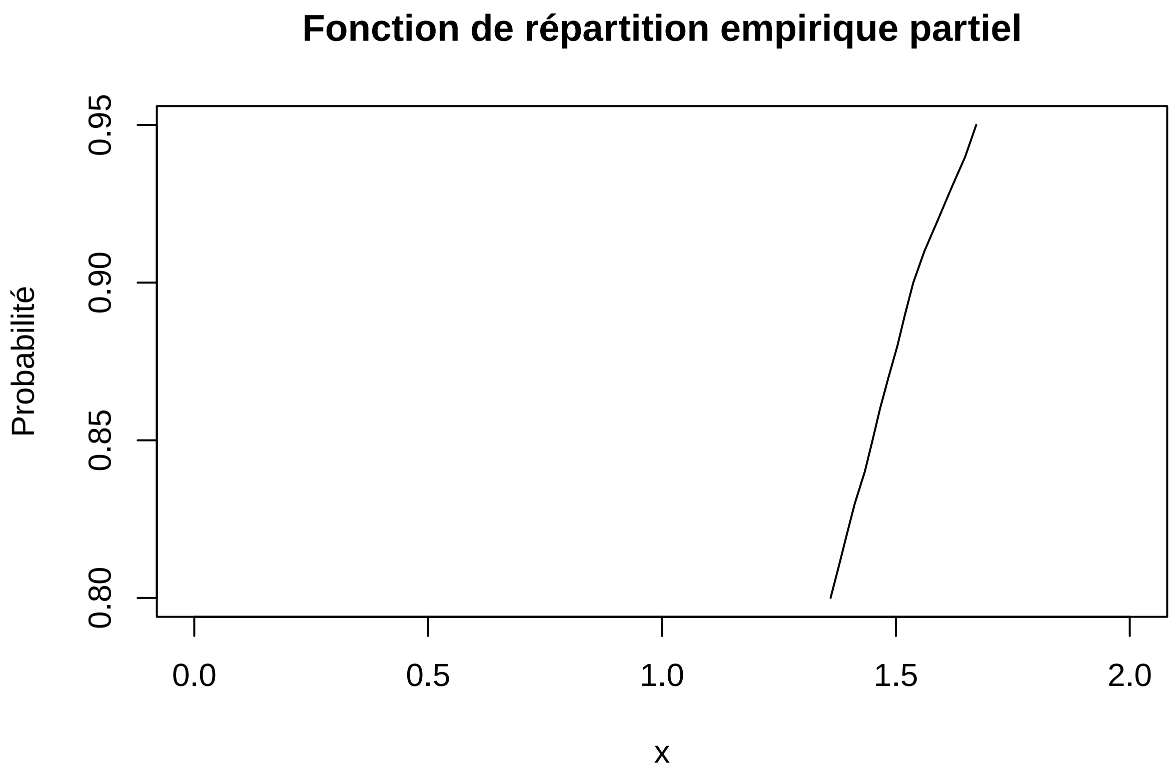
Calcul des échantillons

```
p = seq(0.8, 0.95, 0.01)
sorted_echantillon = sort(runif(n, 0, 1) + runif(n,0,1))
df = data.frame(
  proba = p,
  estimated_quantile = Q(n, p, sorted_echantillon)
)
df
```

```
##    proba estimated_quantile
## 1   0.80           1.360384
## 2   0.81           1.377930
## 3   0.82           1.394900
## 4   0.83           1.412260
## 5   0.84           1.433467
## 6   0.85           1.450076
## 7   0.86           1.466103
## 8   0.87           1.484286
## 9   0.88           1.503346
## 10  0.89           1.519626
## 11  0.90           1.537224
## 12  0.91           1.561067
## 13  0.92           1.590139
## 14  0.93           1.618646
## 15  0.94           1.648322
## 16  0.95           1.671707
```

Fonction de répartition empirique pour $p \in [0.8, 0.95]$

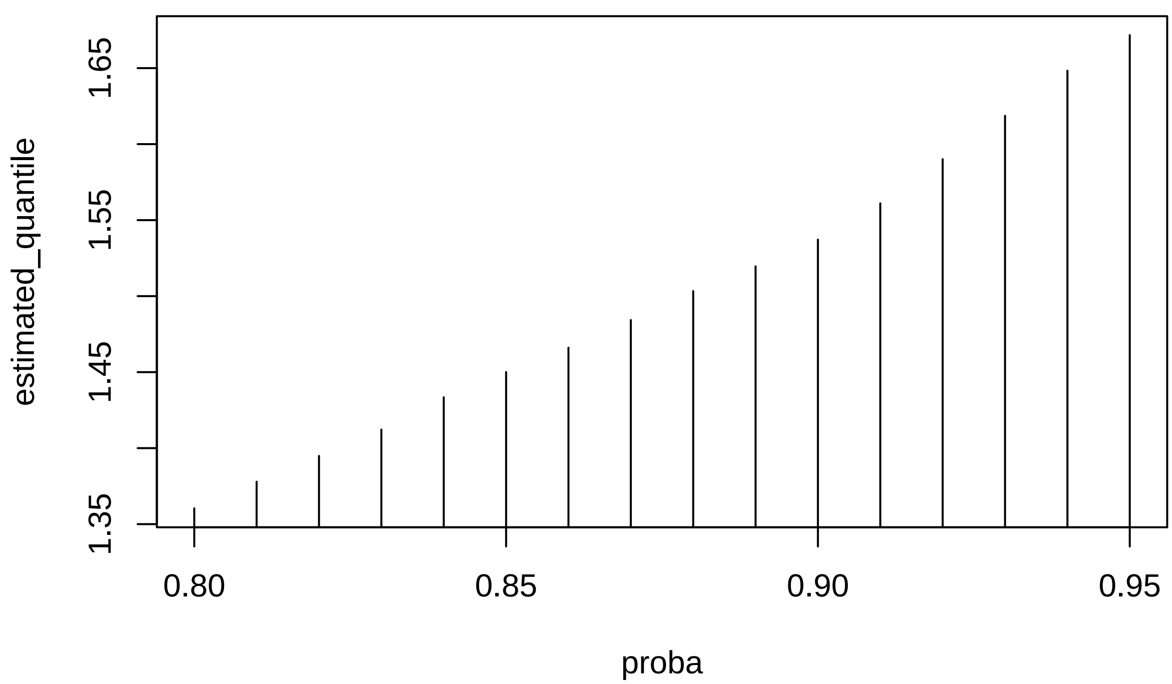
```
plot(df$estimated_quantile,df$proba, xlab="x", ylab="Probabilité", type="l", xlim=c(0,2))
title("Fonction de répartition empirique partiel")
```



Histograme des estimation pour $p \in [0.8, 0.95]$

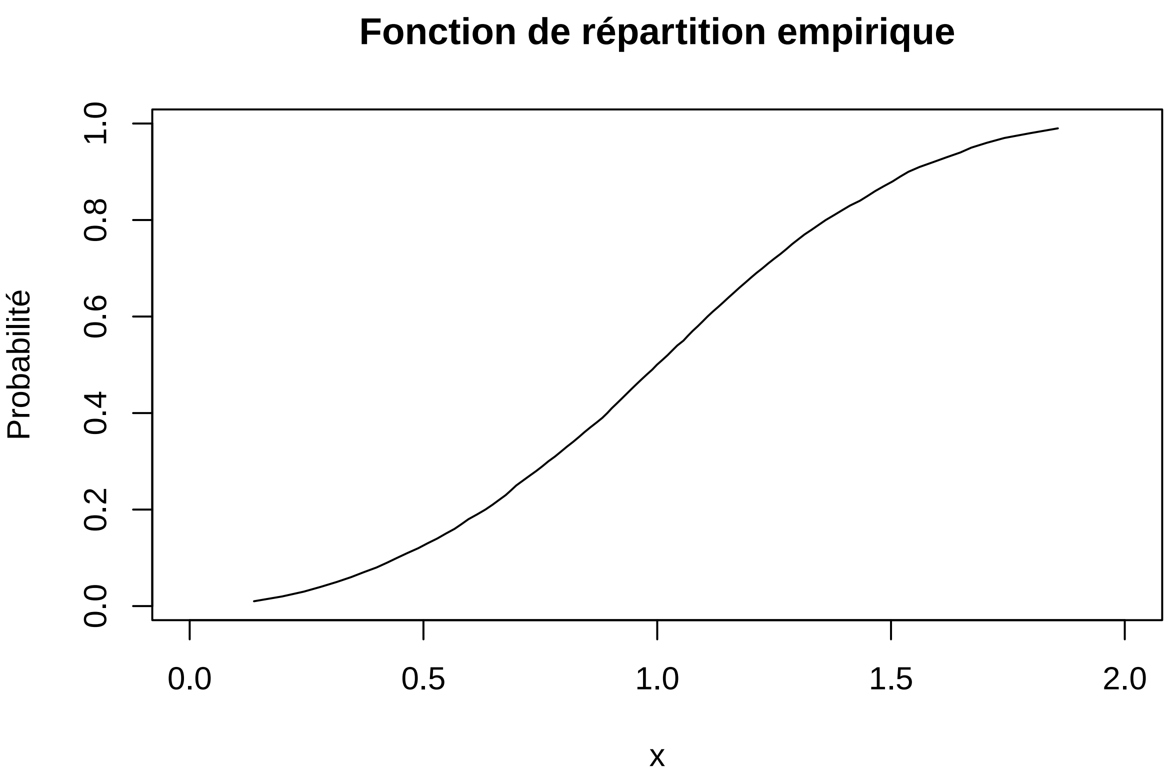
```
plot(df, type="hist")
```

```
## Warning in plot.xy(xy, type, ...): plot type 'hist' will be truncated to first
## character
```



Fonction de répartition empirique complète

```
p = seq(0.01, 0.99, 0.01)
df = data.frame(
  proba = p,
  estimated_quantile = Q(n, p, sorted_echantillon)
)
plot(df$estimated_quantile,df$proba, xlab="x", ylab="Probabilité", type="l", xlim=c(0,2))
title("Fonction de répartition empirique")
```



Question 5

$$h_{10} = \sup_{t \in \mathbb{R}} \{ \frac{1}{10} \sum_{i=1}^{10} 1_{U_i \leq t} - t \} = \max_{1 \leq i \leq n} (\max(\left| \frac{i}{n} - t \right|, \left| \frac{i-1}{n} - t) \right|))$$

Fonction pour obtenir un échantillon de `sample_lenght` statistique de KS de paramètre `n`.

```
ks_stat_unif = function(n, sample_lenght){
  ks_ech = c()
  # On créé un échantillon de longueur sample_lenght
  for(i in 1:sample_lenght){
    tmp = c()
    # On effectue un tirage de n loi uniforme pour calculer la stat de test
    sorted_unif = sort(runif(n, 0, 1))

    # Puis on cherche le max suivant la formule précédante
    for(i in 1:n){
      tmp = c(tmp,max(
              abs(i/n - sorted_unif[i]),
              abs((i-1)/n - sorted_unif[i])
```

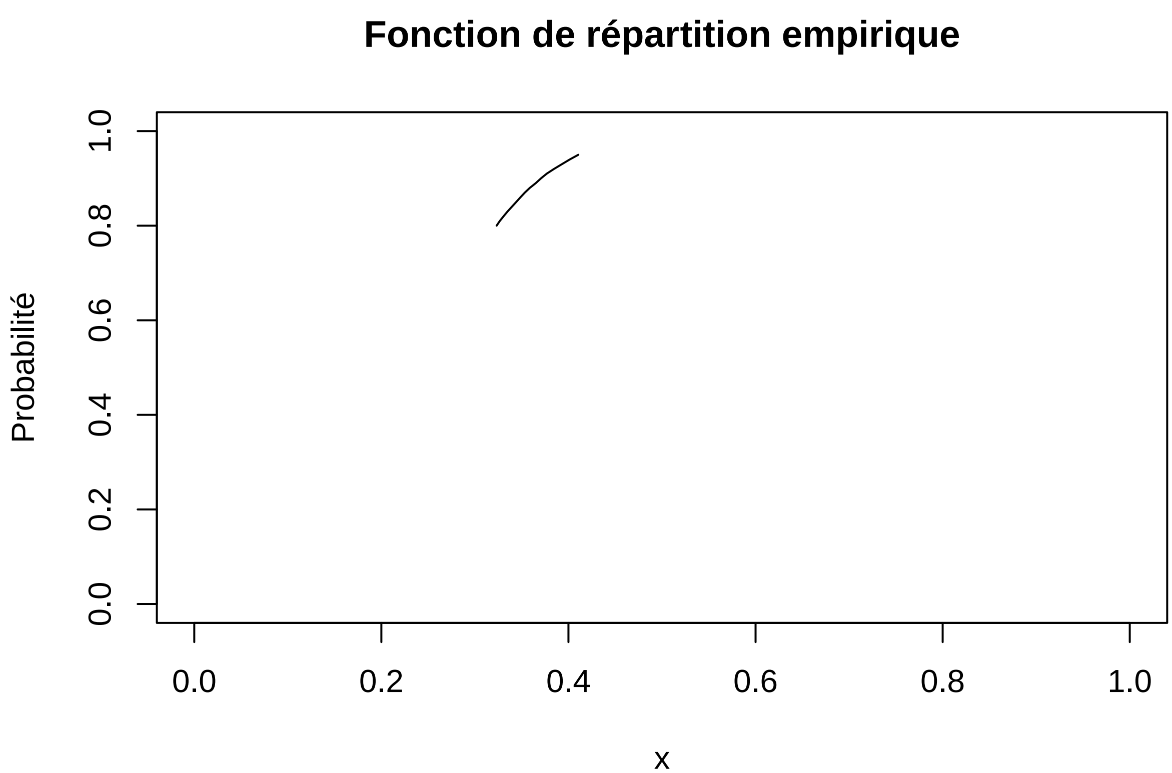
```
            )
        )
    }
    ks_ech = c(ks_ech, max(tmp))
  }
  return(ks_ech)
}
```

Calcul des échantillons

```
p = seq(0.8, 0.95, 0.01)
sorted_echantillon = sort(ks_stat_unif(10, n)) #tirage de 10 000 stat de KS de paramètre 10
df = data.frame(
  proba = p,
  estimated_quantile = Q(n, p, sorted_echantillon)
)
```

Fonction de répartition empirique pour $p \in [0.8, 0.95]$

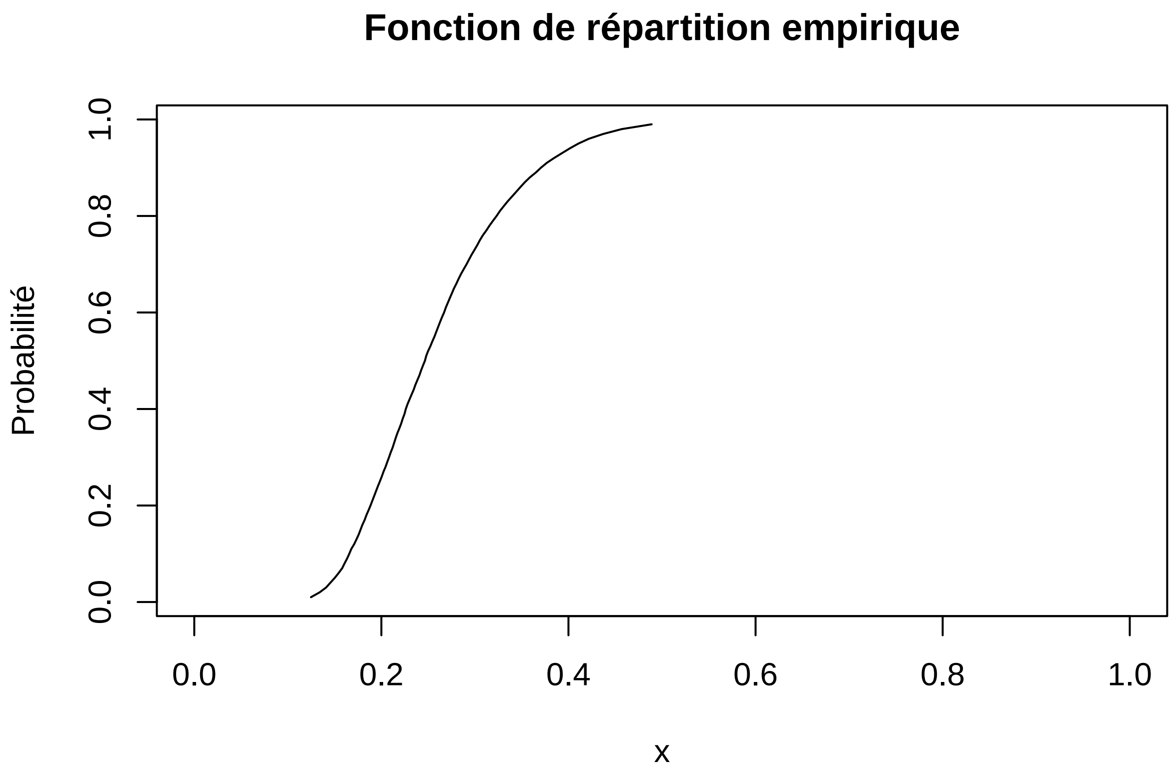
```
plot(df$estimated_quantile,df$proba, xlab="x", ylab="Probabilité", type="l", xlim=c(0,1), ylim=c(0,1))
title("Fonction de répartition empirique")
```



Fonction de répartition empirique complète

```
p = seq(0.01, 0.99, 0.01)
df = data.frame(
  proba = p,
  estimated_quantile = Q(n, p, sorted_echantillon)
```

```
)
plot(df$estimated_quantile,df$proba, xlab="x", ylab="Probabilité", type="l", xlim=c(0,1))
title("Fonction de répartition empirique")
```



Comparaison avec les quantiles de notre table

```
df[c(80, 90, 95, 98, 99),]
```

```
##    proba estimated_quantile
## 80  0.80          0.3231620
## 90  0.90          0.3705561
## 95  0.95          0.4105604
## 98  0.98          0.4568470
## 99  0.99          0.4889369
```

Cela correspond bien à ce qu'on a dans notre table avec une précision à 10^{-2}

Question 6

Fonction pour obtenir un échantillon de `sample_lenght` statistique de Lilliefors de paramètre `n`.

```
lilliefors_stat = function(n, sample_lenght){
  lilliefors_ech = c()
  # On créé un échantillon de longueur sample_lenght
  for(i in 1:sample_lenght){
    tmp = c()
    # On effectue un tirage de n loi normale pour calculer la stat de test
    sorted_norm = sort(rnorm(n, 0, 1))
    X_bar = mean(sorted_norm)
```

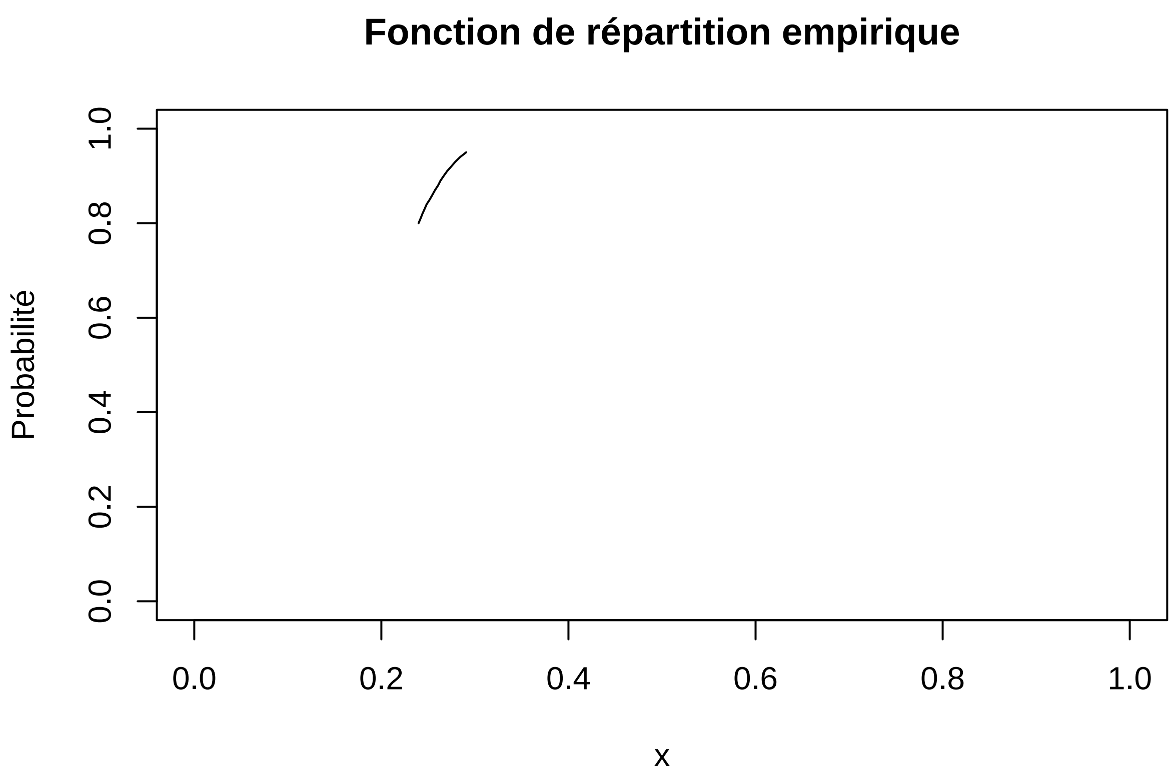
```
    V_n = var(sorted_norm)

    # Puis on cherche le max suivant la formule précédante
    for(i in 1:n){
      tmp = c(tmp,max(
              abs(i/n - pnorm(sorted_norm[i], X_bar, V_n)),
              abs((i-1)/n - pnorm(sorted_norm[i], X_bar, V_n))
            )
        )
    }
    lilliefors_ech = c(lilliefors_ech, max(tmp))
  }
  return(lilliefors_ech)
}
```

Calcul des échantillons

```
p = seq(0.8, 0.95, 0.01)
sorted_echantillon = sort(lilliefors_stat(10, n)) #tirage de 10 000 stat de Lilliefors de paramètre 10
df = data.frame(
  proba = p,
  estimated_quantile = Q(n, p, sorted_echantillon)
)
```

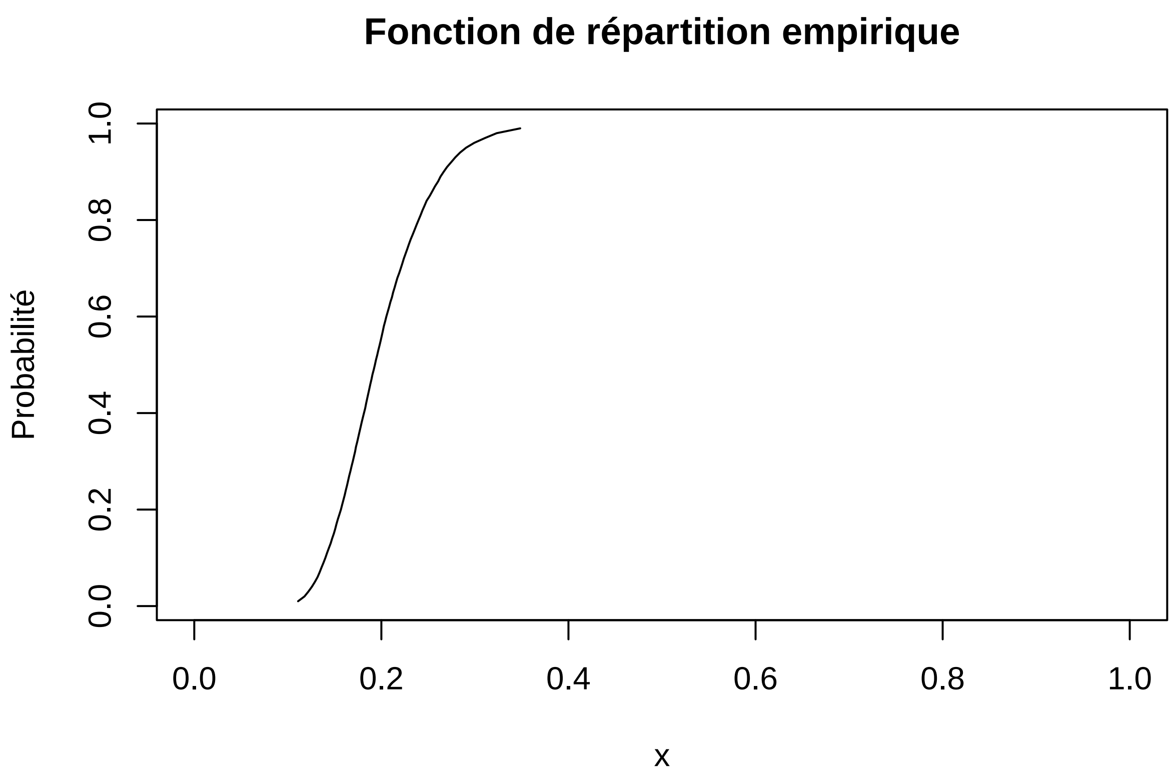
Fonction de répartition empirique pour $p \in [0.8, 0.95]$

```
plot(df$estimated_quantile,df$proba, xlab="x", ylab="Probabilité", type="l", xlim=c(0,1), ylim=c(0,1))
title("Fonction de répartition empirique")
```



Fonction de répartition empirique complète

```
p = seq(0.01, 0.99, 0.01)
df = data.frame(
  proba = p,
  estimated_quantile = Q(n, p, sorted_echantillon)
)
plot(df$estimated_quantile,df$proba, xlab="x", ylab="Probabilité", type="l", xlim=c(0,1))
title("Fonction de répartition empirique")
```



Comparaison avec les quantiles de notre table

```
df[c(90, 95, 99),]
```

```
##    proba estimated_quantile
## 90  0.90          0.2666081
## 95  0.95          0.2906601
## 99  0.99          0.3484613
```

Cela correspond pas vraiment à ce qu'il y a dans notre table. Essayons d'augmenter l'échantillon.

```
p = c(0.90, 0.95, 0.99)
n = 10000
sorted_echantillon = sort(lilliefors_stat(10, n*10))
sorted_echantillon_small = sort(lilliefors_stat(10, n))
df = data.frame(
  proba = p,
  estimated_quantile_big_n = Q(n*10, p, sorted_echantillon),
  estimated_quantile_smaller_n = Q(n, p, sorted_echantillon_small)
)
```

```r
df
```

```
##   proba estimated_quantile_big_n estimated_quantile_smaller_n
## 1  0.90                0.2663579                    0.2647339
## 2  0.95                0.2910344                    0.2894681
## 3  0.99                0.3437076                    0.3461248
```