```
# Exercice 9 : intervalles de confiance
#
#a) Generer n = 10 ...
n = 10; x = rnorm(n)
res = t.test(x, conf.level = 0.99)
res$conf.int
#b) Estimer le niveau empirique du test ...
K = 20000; a = 1:K
for( i in 1 : K)
{
  x = rnorm(n)
  res = t.test(x, conf.level = 0.95)
  a[i] = (res$conf.int[1] < 0)&(res$conf.int[2] > 0)
}
sum(a)/K
#
# Exercice 10 : tests d'adequation
#
#a) Reecrire le test d'adequation ...
chi2.adequation = function(x,intervals)
{
  # taille echantillon
  nx = length(x)
  # parametres loi
  mx = mean(x); sdx = sd(x)
  # nombre classes
  k = length(intervals)-1
  # initialise effectif empirique
  n.empi = 1:k
  # initialise effectif theorique
  n.theo = 1:k
  # calcul de la statistique
  for(i in 1: k)
  {
    n.empi[i] = sum(x >= intervals[i] & x <
intervals[i+1])
    n.theo[i] = nx*(pnorm(intervals[i+1], mx, sdx)-
pnorm(intervals[i], mx, sdx))
  }
  if(any(n.theo<5))
  {
    print("WARNING : effectif theorique trop petit");
    break
  }
  # calcul de la statistique
```

```
  D = sum((n.empi-n.theo)^2/n.theo)
  # retourner la p-value
  list(p.value = 1-pchisq(D, k-1-2))
}
#### pareto
chi2.pareto = function(x,intervals,p)
{ nx = length(x); k = length(intervals)-1
  n.empi = 1:k; n.theo = 1:k
  for(i in 1: k)
  { n.empi[i] = sum(x >= intervals[i] & x <
intervals[i+1])
    n.theo[i] = nx*(ppareto(intervals[i+1], p)-
ppareto(intervals[i], p))
  }
  if(any(n.theo<5))
  { print("WARNING : effectif theorique trop petit");
    break
  }
  D = sum((n.empi-n.theo)^2/n.theo)
  list(p.value = 1-pchisq(D, k-1))
}

dpareto <- function(x, p){p/(x^(p+1))};
qpareto <- function(y, p){(1/(1-y))^(1/p)}
rpareto <- function(n, p){qpareto(runif(n),p)}
ppareto <- function(x, p){
  y = length(x); y[x<=1] = 0;
  y[x>1] = 1-1/(x[x>1]^p);
  y
}
ppareto(0:2, 3)
# c) tester
x = rpareto(100, 3);
intervals = c(-Inf, 1.5, 2, Inf)
chi2.pareto(x, intervals, 2)

nessai = 1000; rejet.chi = 0; rejet.ks = 0;
nx = 100; k = 3; p = 3;
for(i in 1:nessai)
{ x = rpareto(nx, 2);
  #x = runif(nx, min = 1, max = 3);
  pvalue.chi = chi2.pareto(x,intervals, p)$p.value
  pvalue.ks = ks.test(x, "ppareto", p)$p.value
  if(pvalue.chi < 0.05) rejet.chi = rejet.chi+1
  if(pvalue.ks < 0.05) rejet.ks = rejet.ks+1}
```

```
rejet.chi/nessai; rejet.ks/nessai;
# b) Tester l'hypothese d'adequation ...
# generer les variables normales et les intervalles ...
x = rnorm(100); intervals = c(-Inf,-1 : 1, Inf)
# taille echantillon
nx = length(x)
# parametres loi
mx = mean(x); sdx = sd(x)
# nombre classes
k = length(intervals)-1
# initialise effectif empirique
n.empi = 1:k
# initialise effectif theorique
n.theo = 1:k
# calcul de la statistique
for(i in 1: k)
{
  n.empi[i] = sum(x >= intervals[i] & x < intervals[i+1])
  n.theo[i] = nx*(pnorm(intervals[i+1], mx, sdx)-
pnorm(intervals[i], mx, sdx))
}
if(any(n.theo<5))
{
  printf("WARNING : effectif theorique trop petit");
  break
}
# calcul de la statistique
T = sum((n.empi-n.theo)^2/n.theo)
# retourner la p-value
p.value = 1-pchisq(T, k-1-2)

chi2.adequation(x,intervals)$p.value

chisq.test(n.empi, p = n.theo, rescale.p = T)
1-pchisq(T, k-1)
###
nessai = 1000; rejet1 = 0; rejet2 = 0; rejet3 = 0
intervals = c(-Inf,-1 : 1, Inf); nx = 100; k = 4
for(i in 1:nessai)
{
  # x = rnorm(nx)
  x = runif(nx, min = -2, 2)
  pvalue1 = chi2.adequation(x,c(-Inf,-1 : 1, Inf))$p.value
  mx = mean(x); sdx = sd(x)
  n.empi = 1:k; n.theo = 1:k; n.theo2 = 1:k
```

```
  for(i in 1: k)
  {
    n.empi[i] = sum(x >= intervals[i] & x <
intervals[i+1])
    n.theo[i] = nx*(pnorm(intervals[i+1], mx, sdx)-
pnorm(intervals[i], mx, sdx))
    n.theo2[i] = nx*(pnorm(intervals[i+1])-
pnorm(intervals[i]))
  }
  pvalue2 = chisq.test(n.empi, p = n.theo, rescale.p =
T)$p.value
  pvalue3 = chisq.test(n.empi, p = n.theo2, rescale.p =
T)$p.value
  if(pvalue1 < 0.05) rejet1 = rejet1+1
  if(pvalue2 < 0.05) rejet2 = rejet2+1
  if(pvalue3 < 0.05) rejet3 = rejet3+1
}
rejet1/nessai; rejet2/nessai; rejet3/nessai;

# ... tester
x = rnorm(100); intervals = c(-Inf,-1 : 1, Inf)
chi2.adequation(x,intervals)$p.value
ks.test(x,"pnorm")$p.value
shapiro.test(x)$p.value
#  Quel est le risque empirique ?
risque.empirique = function(nessai)
{
  rejet.chi2 = 0
  rejet.ks = 0
  rejet.shapiro = 0
  for(i in 1:nessai)
  {
    x = rnorm(100)
    pvalue.chi2 = chi2.adequation(x,c(-Inf,-1 : 1,
Inf))$p.value
    if(pvalue.chi2 < 0.01)
      rejet.chi2 = rejet.chi2+1
    pvalue.ks = ks.test(x,"pnorm")$p.value
    if(pvalue.ks < 0.01)
      rejet.ks = rejet.ks+1
    pvalue.shapiro = shapiro.test(x)$p.value
    if(pvalue.shapiro < 0.01)
      rejet.shapiro = rejet.shapiro+1
  }
```

```
list(alpha.chi2=rejet.chi2/nessai,alpha.ks=rejet.ks/nessai,

       alpha.shapiro=rejet.shapiro/nessai)
}
risque.empirique(2000)
#
# c) On considere maintenant un echantillon ...
# On utilise la meme function avec une loi uniforme que
l'on appele
# puissance.empirique ...
#
puissance.empirique = function(nessai)
{
  rejet.chi2=0
  rejet.ks=0
  rejet.shapiro=0

  for(i in 1:nessai)
  {
    x=runif(100,min=-2,max=2)
    pvalue.chi2=chi2.adequation(x,c(-Inf,-1 : 1,
Inf))$p.value
    if(pvalue.chi2 < 0.05)
      rejet.chi2=rejet.chi2+1
    pvalue.ks=ks.test(x,"pnorm",0,1)$p.value
    if(pvalue.ks < 0.05)
      rejet.ks=rejet.ks+1
    pvalue.shapiro=shapiro.test(x)$p.value
    if(pvalue.shapiro < 0.05)
      rejet.shapiro=rejet.shapiro+1
  }
  list(pi.chi2=rejet.chi2/nessai,pi.ks=rejet.ks/nessai,
       pi.shapiro=rejet.shapiro/nessai)
}
puissance.empirique(1000)
################################
# Exercice 11 : tests d'adequation à une loi à queue
lourde
#a) Créer les fonctions...
ppareto <- function(x, p){
  if(x < 1){0}else{1-1/(x^p)}}

ppareto <- function(x, p){
  y = 1:length(x)
  y[x<=1] = 0; y[x>1] = 1-1/(x[x>1]^p);
```

```
  y
}
qpareto <- function(y, p){(1/(1-y))^(1/p)}
rpareto <- function(n, p){qpareto(runif(n), p)}
#b) Reecrire le test d'adequation ...
chi2.pareto = function(x, intervals, p)
{
  # taille echantillon
  nx = length(x)
  # nombre classes
  k = length(intervals)-1
  # initialise effectif empirique
  n.empi = 1:k
  # initialise effectif theorique
  n.theo = 1:k
  # calcul de la statistique
  for(i in 1: k)
  {
    n.empi[i] = sum(x >= intervals[i] & x <
intervals[i+1])
    n.theo[i] = nx*(ppareto(intervals[i+1], p)-
ppareto(intervals[i], p))
  }
  if(any(n.theo<5))
  {
    printf("WARNING : effectif theorique trop petit");
    break
  }
  # calcul de la statistique
  D = sum((n.empi-n.theo)^2/n.theo)
  # retourner la p-value
  list(p.value = 1 - pchisq(D, k-1))
}
# b) Tester l'hypothese d'adequation ...
# generer les variables normales et les intervalles ...
p = 3
x = rpareto(100, 3); intervals = c(-Inf,1.5,2, Inf)
# ... tester
chi2.pareto(x, intervals, 3)$p.value
#  Quel est le risque empirique ?
risque.empirique = function(nessai)
{
  rejet.chi2 = 0
  rejet.ks = 0
  for(i in 1:nessai)
```

```
  {
    x = rpareto(100, p )
    pvalue.chi2 = chi2.pareto(x,intervals, p)$p.value
    if(pvalue.chi2 < 0.05)
      rejet.chi2 = rejet.chi2+1
    pvalue.ks = ks.test(x,"ppareto", p)$p.value
    if(pvalue.ks < 0.05)
      rejet.ks = rejet.ks+1
  }
  list(alpha.chi2=rejet.chi2/nessai,
alpha.ks=rejet.ks/nessai)
}
risque.empirique(2000)
#
# c) On considere maintenant un echantillon ...
# On utilise la meme function avec une loi uniforme que
l'on appele
# puissance.empirique ...
#
puissance.empirique = function(nessai)
{
  rejet.chi2 = 0; rejet.ks = 0
  for(i in 1:nessai)
  {
    # x = runif(100,min=1,max=3)
    x = rpareto(100, 2)
    pvalue.chi2 = chi2.pareto(x,intervals, p)$p.value
    if(pvalue.chi2 < 0.05)
      rejet.chi2=rejet.chi2+1
    pvalue.ks = ks.test(x,"ppareto", p)$p.value
    if(pvalue.ks < 0.05)
      rejet.ks=rejet.ks+1
  }
  list(pi.chi2=rejet.chi2/nessai,pi.ks=rejet.ks/nessai)
}
puissance.empirique(1000)

#
# Exercice 12 : regression
#

#a)

# generation de l'echantillon et affichage du nuage de
points
```

```
x=1:30
y=10+x+5*sin(x)+2*rnorm(30)
plot(x,y)

# afficher la vraie courbe
xall=seq(0,30,0.01)
yall=10+xall+5*sin(xall)
lines(xall,yall)

#b)

# estimer les parametres du modele
reg.simple=lm(y~x)

# estimer la droite de regression
attach(reg.simple)
y.simple=coefficients["
(Intercept)"]+coefficients["x"]*xall
lines(xall,y.simple,lty=2)
summary(reg.simple)
# les residus on les voit avec ...
plot(reg.simple)

# estimer les parametres du modele
reg.complex=lm(y~x+sin(x)+I(x^2))

# estimer la droite de regression
detach(reg.simple)
attach(reg.complex)
y.complex=coefficients["
(Intercept)"]+coefficients["x"]*xall
+coefficients["sin(x)"]*sin(xall)
+coefficients["I(x^2)"]*xall^2
lines(xall,y.complex,lty=3)

summary(reg.complex)

# estimer les parametres du modele
reg.ok=lm(y~x+sin(x))

# estimer la droite de regression
detach(reg.complex)
attach(reg.ok)
y.ok=coefficients["(Intercept)"]+coefficients["x"]*xall
+coefficients["sin(x)"]*sin(xall)
```

```
lines(xall,y.ok,lty=4)

summary(reg.ok)
summary(reg.simple)
#
# ... les criteres AIC choisissent le modele correct,
alors que R^2 choisit
# le modele le plus complique ; R bar ^2 peut
eventuellement convenir ...
#

AIC(reg.simple)
AIC(reg.complex)
AIC(reg.ok)

# ... pour verifier les residus
plot(reg.ok)
```