Calcular Estadisticos Notas

Imanol

1/3/2021

Análisis de datos cuantitativos

Creamos el vector con los datos

```
set.seed(4)
notas = sample(0:10,100, replace = TRUE)
set.seed(NULL)
notas
```

```
##   [1]  7 10  2  2  6  2  5  4  9  2  7  5  1  7  0  3 10  2 10  4  1  4  5  4  0
##  [26]  5 10  4  3  0  7  5 10  3  4  8  1  9  3  7  9  1  9 10  5 10 10  9  5  0
##  [51]  3  1  3  2  0  6  6  4  7  4  7  3  9  0  7  0  3  0  3  3  1  4 10  9  1
##  [76]  4  0  6 10  0 10  1  0  2  6  4  8  2  3  7  7  3  3  8  2  6  6  2  8  9
```

Definir vector de los extremos

En este caso los intervalos seran: [0,5), [5,7),[7,9),[9,10].

```
L = c(0,5,7,9,10)
# Definimos notas1 como el resultado de la codificación en intervalos utilizando como etiquetas los pro
notas1 = cut(notas, breaks = L, right = FALSE, include.lowest = TRUE)
notas1
```

```
##   [1] [7,9)   [9,10] [0,5)   [0,5)   [5,7)   [0,5)   [5,7)   [0,5)   [9,10] [0,5)
##  [11] [7,9)   [5,7)   [0,5)   [7,9)   [0,5)   [0,5)   [9,10] [0,5)   [9,10] [0,5)
##  [21] [0,5)   [0,5)   [5,7)   [0,5)   [0,5)   [5,7)   [9,10] [0,5)   [0,5)   [0,5)
##  [31] [7,9)   [5,7)   [9,10] [0,5)   [0,5)   [7,9)   [0,5)   [9,10] [0,5)   [7,9)
##  [41] [9,10] [0,5)   [9,10] [9,10] [5,7)   [9,10] [9,10] [9,10] [5,7)   [0,5)
##  [51] [0,5)   [0,5)   [0,5)   [0,5)   [0,5)   [5,7)   [5,7)   [0,5)   [7,9)   [0,5)
##  [61] [7,9)   [0,5)   [9,10] [0,5)   [7,9)   [0,5)   [0,5)   [0,5)   [0,5)   [0,5)
##  [71] [0,5)   [0,5)   [9,10] [9,10] [0,5)   [0,5)   [0,5)   [5,7)   [9,10] [0,5)
##  [81] [9,10] [0,5)   [0,5)   [0,5)   [5,7)   [0,5)   [7,9)   [0,5)   [0,5)   [7,9)
##  [91] [7,9)   [0,5)   [0,5)   [7,9)   [0,5)   [5,7)   [5,7)   [0,5)   [7,9)   [9,10]
## Levels: [0,5) [5,7) [7,9) [9,10]
```

Definir marcas de clase

En este caso los intervalos seran: [0,5), [5,7),[7,9),[9,10].

```r
#Definimos las marcas de clase
MC = (L[1:length(L)-1] + L[2:length(L)])/2
# Definimos notas2 como el resultado de la codificación en intervalos utilizando como etiquetas las mar
notas2 = cut(notas, breaks = L, labels = MC,right = FALSE, include.lowest = TRUE)
notas2
```

```
##   [1] 8   9.5 2.5 2.5 6   2.5 6   2.5 9.5 2.5 8   6   2.5 8   2.5 2.5 9.5 2.5
##  [19] 9.5 2.5 2.5 2.5 6   2.5 2.5 6   9.5 2.5 2.5 2.5 8   6   9.5 2.5 2.5 8
##  [37] 2.5 9.5 2.5 8   9.5 2.5 9.5 9.5 6   9.5 9.5 9.5 6   2.5 2.5 2.5 2.5 2.5
##  [55] 2.5 6   6   2.5 8   2.5 8   2.5 9.5 2.5 8   2.5 2.5 2.5 2.5 2.5 2.5 2.5
##  [73] 9.5 9.5 2.5 2.5 2.5 6   9.5 2.5 9.5 2.5 2.5 2.5 6   2.5 8   2.5 2.5 8
##  [91] 8   2.5 2.5 8   2.5 6   6   2.5 8   9.5
## Levels: 2.5 6 8 9.5
```

Definir intervalos por su orden

```r
# Definimos notas3 como el resultado de la codificación en intervalos utilizando como etiquetas la posi
notas3 = cut(notas, breaks = L, labels = FALSE,right = FALSE, include.lowest = TRUE)
notas3
```

```
##   [1] 3 4 1 1 2 1 2 1 4 1 3 2 1 3 1 1 4 1 4 1 1 1 2 1 1 2 4 1 1 1 3 2 4 1 1 3 1
##  [38] 4 1 3 4 1 4 4 2 4 4 4 2 1 1 1 1 1 1 2 2 1 3 1 3 1 4 1 3 1 1 1 1 1 1 1 4 4
##  [75] 1 1 1 2 4 1 4 1 1 1 2 1 3 1 1 3 3 1 1 3 1 2 2 1 3 4
```

Definir intervalos por calificación

```r
# Definimos notas3 como el resultado de la codificación en intervalos utilizando como etiquetas Susp, A
notas4 = cut(notas, breaks = L, labels = c("Susp","Aprob", "Not", "Exc"),right = FALSE, include.lowest =
notas4
```

```
##   [1] Not   Exc   Susp  Susp  Aprob Susp  Aprob Susp  Exc   Susp  Not   Aprob
##  [13] Susp  Not   Susp  Susp  Exc   Susp  Exc   Susp  Susp  Susp  Aprob Susp
##  [25] Susp  Aprob Exc   Susp  Susp  Susp  Not   Aprob Exc   Susp  Susp  Not
##  [37] Susp  Exc   Susp  Not   Exc   Susp  Exc   Exc   Aprob Exc   Exc   Exc
##  [49] Aprob Susp  Susp  Susp  Susp  Susp  Susp  Aprob Aprob Susp  Not   Susp
##  [61] Not   Susp  Exc   Susp  Not   Susp  Susp  Susp  Susp  Susp  Susp  Susp
##  [73] Exc   Exc   Susp  Susp  Susp  Aprob Exc   Susp  Exc   Susp  Susp  Susp
##  [85] Aprob Susp  Not   Susp  Susp  Not   Not   Susp  Susp  Not   Susp  Aprob
##  [97] Aprob Susp  Not   Exc
## Levels: Susp Aprob Not Exc
```

Calculo de las frecuencias

Absolutas

```r
table(notas4)
```

```
## notas4
##  Susp Aprob   Not   Exc
##    53    14    14    19
```

Relativas

```
prop.table(table(notas4))
```

```
## notas4
##  Susp Aprob   Not   Exc
##  0.53  0.14  0.14  0.19
```

Absolutas acumuladas

```
cumsum(table(notas4))
```

```
##  Susp Aprob   Not   Exc
##    53    67    81   100
```

Relativas acumuladas

```
cumsum(prop.table(table(notas4)))
```

```
##  Susp Aprob   Not   Exc
##  0.53  0.67  0.81  1.00
```

Todo lo anterior pero usando la funcion hist

```
notasHist = hist(notas, breaks = L, right = FALSE, include.lowest = TRUE, plot = FALSE)
FAbs = notasHist$count
FRel = prop.table(FAbs)
FAbsCum = cumsum(FAbs)
FRelCum = cumsum(FRel)
```

Crear data frame con todas las frecuencias

```
intervalos = c("[0,5)","[5,7)","[7,9)","[9,10]")
clasificacion = c("Suspenso","Aprobado","Notable","Excelente")
marcas = notasHist$mids #Mids nos da las marcas de clase del histograma
tabla.Fr = data.frame(intervalos,clasificacion,marcas,FAbs,FAbsCum,FRel,FRelCum)
tabla.Fr
```

```
##   intervalos clasificacion marcas FAbs FAbsCum FRel FRelCum
## 1      [0,5)      Suspenso    2.5   53      53 0.53    0.53
## 2      [5,7)      Aprobado    6.0   14      67 0.14    0.67
## 3      [7,9)       Notable    8.0   14      81 0.14    0.81
## 4     [9,10]     Excelente    9.5   19     100 0.19    1.00
```

Otra opción es usar una de las dos funciones que hemos creado anteriormente (script tabla de frecuencias)

```
TablaFrecs.L = function(x,L,V){
  x_cut = cut(x, breaks=L, right=FALSE, include.lowest=V)
  intervals = levels(x_cut)
  mc = (L[1:(length(L)-1)]+L[2:length(L)])/2
  Fr.abs = as.vector(table(x_cut))
  Fr.rel = round(Fr.abs/length(x),4)
  Fr.cum.abs = cumsum(Fr.abs)
  Fr.cum.rel = cumsum(Fr.rel)
  tabla = data.frame(intervals, mc, Fr.abs, Fr.cum.abs, Fr.rel, Fr.cum.rel)
  tabla
}
TablaFrecs.L(notas, L, TRUE)
```

```
##   intervals  mc Fr.abs Fr.cum.abs Fr.rel Fr.cum.rel
## 1     [0,5) 2.5     53         53   0.53       0.53
## 2     [5,7) 6.0     14         67   0.14       0.67
## 3     [7,9) 8.0     14         81   0.14       0.81
## 4    [9,10] 9.5     19        100   0.19       1.00
```

Calculo de estadisticos en datos agrupados

Calcular el total

```
TOT = tabla.Fr$FAbsCum[4] # Accedo a la ultima fila de la suma acumulada que es dodne sale el total
TOT
```

```
## [1] 100
```

Calcular la media

```
notas.media = round(sum(tabla.Fr$FAbs*tabla.Fr$marcas)/TOT,3)
# suma del producto de las frecuencias absolutas por las marcas de clase dividido por el total
notas.media
```

```
## [1] 5.09
```

Calcular la varianza

```
notas.var = round(sum(tabla.Fr$FAbs*tabla.Fr$marcas^2)/TOT-notas.media^2,3)
# Suma producto de la frecuencia absoluta por las marcas al cuadrado dividido por el total y restarle l
notas.var
```

```
## [1] 8.552
```

Desviación típica

```
notas.dt = round(sqrt(notas.var),3)
notas.dt
```

```
## [1] 2.924
```

Intervalo Modal

```
I.modal = tabla.Fr$intervalos[which(tabla.Fr$FAbs == max(tabla.Fr$FAbs))]
# Los intervalos para los que la frecuencia absoluta es maxima
I.modal
```

```
## [1] [0,5)
## Levels: [0,5) [5,7) [7,9) [9,10]
```

Intervalo crítico para la mediana

```
I.critic = tabla.Fr$intervalos[which(tabla.Fr$FRelCum >= 0.5)]
I.critic[1] # I.critic me saca todos los intervalos por encima de 0.5, por eso le digo que me de la pos
```

```
## [1] [0,5)
## Levels: [0,5) [5,7) [7,9) [9,10]
```

Calcular estimación de la mediana de los datos "Reales"

```
n = TOT
Lc = L[1] # Limite inferior del intervalo
Lc.pos = L[2] # Limite superior
Ac = L[2]-L[1] # Amplitud del intervalo
Nc.ant = 0 # # Frecuencia absoluta acumulada del anterior en este caso 0 por ser la primera posición pe
nc = tabla.Fr$FAbs[1] # Frecuencia absoluta del intervalo actual
M = Lc + Ac * ((n/2) - Nc.ant) / nc
M # Aproximación de la mediana de los datos "reales"
```

```
## [1] 4.716981
```

Mediana real

```
median(notas)
```

```
## [1] 4
```

Lo creo en una función por si me sirve para calcular mas rapido

```r
med.est = function(P,TOT){
  n = TOT
  Lc = L[P] # Limite inferior del intervalo
  Lc.pos = L[P+1] # Limite superior
  Ac = L[P+1]-L[P] # Amplitud del intervalo
  if(P == 1){
    Nc.ant = 0
  }
  else {
    Nc.ant =  tabla.Fr$FAbsCum[P-1]
  }
  nc = tabla.Fr$FAbs[P] # Frecuencia absoluta del intervalo actual
  M = Lc + Ac * ((n/2) - Nc.ant) / nc
  M # Aproximación de la mediana de los datos "reales"
}

med.est(which(tabla.Fr$intervalos == I.critic)[1],TOT)
```

```
## [1] 4.716981
```

Función para calcular la aproximación a los cuantiles

```r
aprox.quantile.p = function(Lcrit,Acrit,n,p,Ncrit.ant,ncrit){
  round(Lcrit+Acrit*(p*n-Ncrit.ant)/ncrit,3)
}

aprox.quantile.p(Lc,Ac,n,0.25,Nc.ant,nc) #Primer cuartil
```

```
## [1] 2.358
```

```r
aprox.quantile.p(Lc,Ac,n,0.75,Nc.ant,nc) #Tercer cuartil
```

```
## [1] 7.075
```

Versión que hace todo automatico solo pide la posicion del I.crit el total y el quantil

```r
aprox.quantile.p_2 = function(P,TOT,quan){
  n = TOT
  Lc = L[P] # Limite inferior del intervalo
  Lc.pos = L[P+1] # Limite superior
  Ac = L[P+1]-L[P] # Amplitud del intervalo
  if(P == 1){
    Nc.ant = 0
  }
  else {
    Nc.ant =  tabla.Fr$FAbsCum[P-1]
  }
  nc = tabla.Fr$FAbs[P] # Frecuencia absoluta del intervalo actual
```

```
  round(Lc+Ac*(quan*n-Nc.ant)/nc,3)
}

aprox.quantile.p_2(1,n,0.25) #Primer cuartil
```

```
## [1] 2.358
```

```
aprox.quantile.p_2(1,n,0.75) #Tercer cuartil
```

```
## [1] 7.075
```

Cuantiles reales

```
quantile(notas,0.25)
```

```
## 25%
##   2
```

```
quantile(notas,0.75)
```

```
## 75%
##   7
```