Aprendizaje Estadístico - (84:04)

Doctorado en Ingeniería Civil

Alejandro Verri

1 Estimación no-paramétrica

1. En el archivo `buffalo.txt` se encuentran los datos que corresponden a la mediciones de cantidad de nieve caída (en pulgadas) en Buffalo en los inviernos de 1910/1911 a 1972/1973.Realizar un histograma para estos datos utilizando los parámetros por defecto.

```
DT <- fread("data/buffalo.txt", sep=" ") |> transpose()
X <- DT$V1 |> sort()
summary(X)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   25.00   64.50   79.60   80.30   97.65  126.40
```

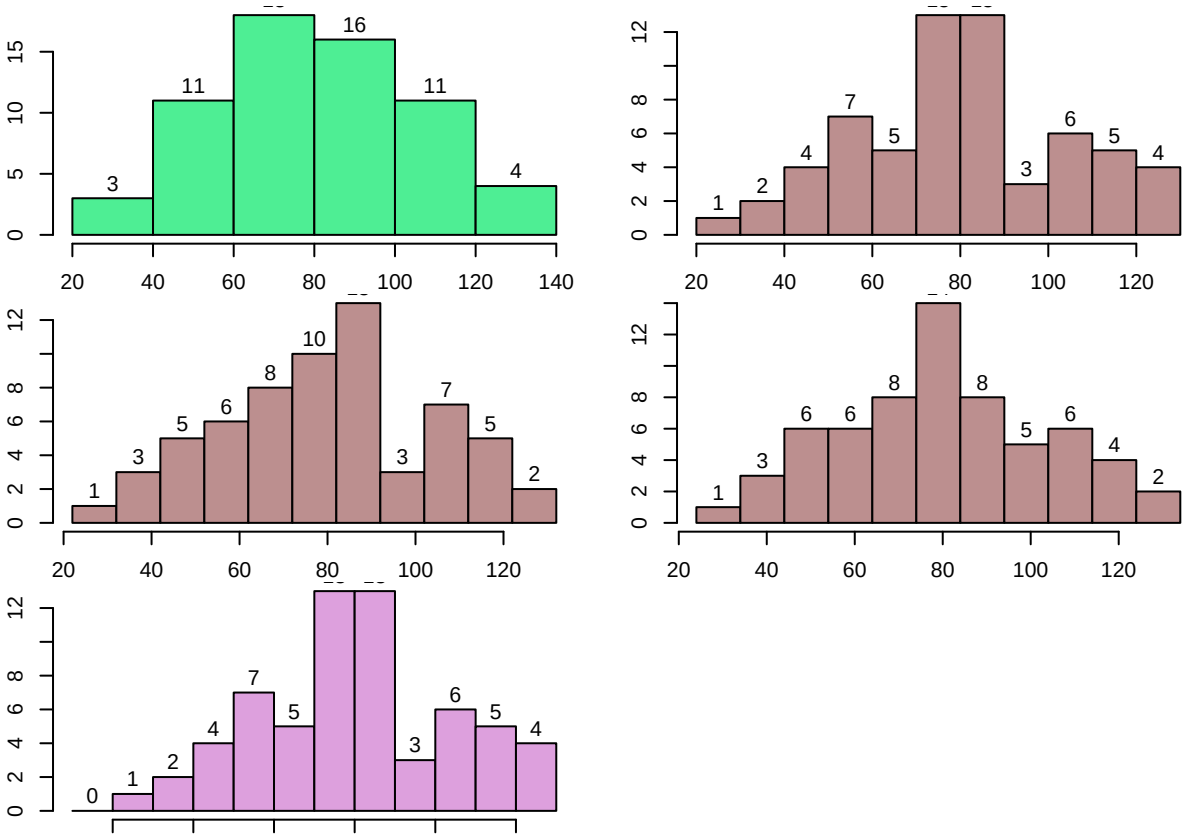
Repetir eligiendo como puntos de corte las siguientes secuencias

a. De 20 a 130 con un paso de 10
b. De 22 a 132 con un paso de 10
c. De 24 a 134 con un paso de 10
d. De 26 a 136 con un paso de 10 [1]

2. Realizar un histograma para estas observaciones utilizando puntos de corte (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130). Comparar todos los histogramas obtenidos. ¿Tiene algún efecto la elección del punto inicial para estos datos?

Para un mismo ancho (bin), la elección del punto inicial (y los subsiguientes puntos de corte) cambia completamente la distribución de frecuencias del histograma.

[1] Este histograma no puede visualizarse con las propiedades por defecto de la función de `R hist()`. El valor mínimo de `X` 25 queda fuera del límite inferior de los puntos de corte.



3. Sea X la cantidad de nieve caída en un invierno en Buffalo. Implementar una función que dados los valores x, h y un conjunto de datos X, permita estimar a $P[X \in [x-h, x+h]] = P[x-h \leq X \leq x+h]$ para cada valor x.

2 Apéndices.

2.1 Funciones `fh.est()`, `hist.est()`

Para el cálculo de histogramas, se definió una función `hist.est(X,h)` siguiendo la definición de Härdle y Siman. La función requiere como argumento el vector aleatoria `X` (la muestra) y el ancho del intervalo (bin) `h`. Alternativamente, los intervalos pueden definirse a través del vector de puntos de corte `hist.est(X,breaks)`. El algoritmo primeramente construye la tabla de intervalos `B` y evalúa la función indicadora `II(X[i],B)` en cada elemento del vector aleatorio `X[i]` y verifica que dicho elemento se encuentre en algún intervalo de la tabla `B`. El código se detalla a continuación

```
# Función histograma.
hist.est <- function(X,h=NULL,breaks=NULL, plot=FALSE){

  if(!is.null(h)){
    # Intervalos definidos por tamaño de paso.
    x0 <- min(X)
    xn <- max(X)
    n <- round((xn-x0)/h,digits = 0)

  } else if(!is.null(breaks) & length(breaks)>1){
    # Intervalos definidos por puntos de corte
    n <- length(breaks)
    H <- breaks[2:n] - breaks[1:(n-1)]
    stopifnot(all(H[1]==H))
    x0 <- min(breaks)
    xn <- max(breaks)
    h <- (xn-x0)/n

  } else if(!is.null(breaks) & length(breaks)==1 & breaks>1){
    # Intervalos definidos por cantidad de puntos de corte
    n <- breaks
    h <- (xn-x0)/n
    x0 <- min(X)
  }
  # Construir intervalos
  B <- data.table(j=seq(1,n))[,.(a=x0+(j-1)*h,b=x0+j*h)]

  # construir columnas de conteos para cada xi
  C <- matrix(0,nrow = n,ncol = 1)
  for(xi in X){
    C <- cbind(C,as.numeric(II(xi,B)))}
  #
  C <- rowSums(C)
  # Construir Tabla de intervalos, counts y densidades
  H <- cbind(B,nx=C,fx=C/(n*h))
  return(H)
}
```

La función indicadora `II(x,B)` requiere una tabla de intervalos `B[a,b]` y un número real `x`,y devuelve un vector con en valor lógico `TRUE` en la posición del intervalo (fila) donde se ubica el valor de `x`

```r
# Función Indicadora
II <- function(x,B){B[,.(ifelse(x>=a & x<=b,TRUE,FALSE))]}
```

La función de estimación de densidad `fh.est(x)` evalúa el histograma en el punto `x` a partir de localizar el valor de `x` en el intervalo `B` mediante la función indicadora `II(x,B)`

```r
# Función de estimación de densidad
fh.est <- function(H=NULL,x,X=NULL,h=NULL,breaks=NULL){
  # Construir el histograma
  if(is.null(H)){
    H <- hist.est(X,h=h,breaks=breaks)
  }
  stopifnot(is.data.table(H),c("a","b","fx") %in% colnames(H))
  # Localizar x en los puntos de corte a,b
  idx <- II(x,H)
  fx <- H$fx[idx]
  return(fx)
}
```