Búsqueda de la media a través de muestras

Markel Ferro

Diciembre 2020

1. Explicación del programa

Cuando se trabaja con grandes cantidades de datos calcular la media a través de métodos convencionales es muy costoso operacionalmente; en estas ocasiones utilizamos una aproximación de la media a través de muestras. Para estar seguros de que la aproximación es relativamente correcta se establecen hipótesis y sus posibilidades de cumplirse se contrastan con un error permitido.

Para empezar se cargan los datos de los cuales se desea calcular la media.

```
7  datos <- read.table("datos.txt")
8  # Al solo tener una columna, lo hacemos mas sencillo de manejar asi
9  datos <- datos$V1
```

Las muestras previamente comentadas se toman en potencias de 2 usando k como exponente de la operación 2^k, empezando con $k = 2$, para dejar un número menor a la hipótesis. Para tomar estas muestras se utiliza la función *obtener_sample* la cual devuelve la media y la varianza de la muestra junto a la muestra.

```
2  obtener_sample <- function(datos, tamano) {
3    muestra <- sample(datos, tamano)
4    media <- mean(muestra)
5    varianza <- var(muestra)
6    return (c(media, varianza, muestra))
7  }
```

También hay que conseguir una hipótesis (*contrastar*) inicial, la cual se guarda en una variable, se utiliza $k - 1$ para obtener una hipótesis con muestra menor a la muestra a la que se contrasta, encima sabemos que en esa línea $k = 2$ con lo que se cogerá una muestra de tamaño 2 (la más pequeña posible):

```
14 contrastar <- obtener_sample(datos,2^(k-1))[1]
```

La primera hipótesis es bastante probable que no se pueda asegurar con la certeza que se desea, con lo que todo el código a partir de este momento deberá estar dentro de un bucle, en este caso se usa un *while* bajo la condición de que una boolean llamada $CONTINUE$ sea *true*.

Para empezar, hay que asegurarse que la muestra que se coge no sea mayor que la cantidad de datos que tenemos, ya que no tiene sentido seleccionar 1500 datos de una selección de 1000. Para ello se utiliza este fragmento de código:

```
19    tamano_muestra <- 2^k
20    if (tamano_muestra > length(datos)) {
21      print("La muestra mayor que el número de datos.")
22      break
23    }
```

A continuación se debe decidir si se quiere utilizar una distribución tipo T-Student o una distribución normal, para ello se utiliza la función *distribucion_correcta* que basándose en la cantidad de datos que tiene nuestra muestra decide la distribución adecuada en cada caso.

```
10  distribucion_correcta <- function(tamano_muestra) {
11    if (tamano_muestra > 30) {
12      return("NORMAL")
13    } else {
14      return("T-STUDENT")
15    }
16  }
```

Pero aunque el tamaño de la muestra es de gran utilidad, todavía tenemos que calcular la muestra, para ello se reúsa la función *obtener_sample* previamente mencionada; para guardar los datos los diferentes datos que devuelve, se debe utilizar el siguiente fragmento de código:

```
28    if (distribucion == "NORMAL") {
29      x <- seq(mean-3*dt, mean+3*dt, by=dt/50)
30      y <- dnorm(x,mean,dt)
31      plot(x,y, type="l", col="green")
```

Una vez obtenida la muestra, obtener la d.t. y la posición opuesta del contrastador es el siguiente paso:

```
33      # Selección de rango https://bit.ly/3hx2R0I
34      bordes <- qt(c(.003, .997), df=tamano_muestra-1) # Dibujar el 99.7% de la
        ↪  distribución
```

```
18  calcular_otro_lado <- function(media,contrastar) {
19    if (media > contrastar) {
20      return(media + abs(media - contrastar))
21    } else {
22      return(media - abs(media - contrastar))
```

```
23    }
24  }
```

Ahora queda la parte más difícil, calcular el p-valor de que la media de nuestra muestra k sea la hipótesis (en este caso llamada contrastador) derivada de la muestra $k-1$. Para realizar estos cálculos correctamente es necesario saber la distribución ya que dependiendo de si es T-Student o una gaussiana se debe proceder a calcular la probabilidad de distintas maneras. También podemos realizar el código de dos maneras, la primera es comparando cual es el valor menor entre los dos para determinar si calcular la cola inferior o no; mientras que la segunda es tener una precondición cuando se llama a la función; en mi caso he decidido tomar la segunda opción, resultando en una función así:

```
88  calcular_p_valor <- function(distribucion, mean, dt, tamano_muestra,
     ↪  contrastar) {
89    # La función siempre recibe una mean < contrastar
90    # Tipificamos (posiblemente no sea necesario)
91    a_tabla <- (mean-contrastar)/dt
92    if (distribucion == "T-STUDENT") {
93      p_valor <- pt(a_tabla, tamano_muestra-1)
94    } else { # Normal
95      p_valor <- pnorm(a_tabla, lower.tail = TRUE)
96    }
97    return(p_valor)
98  }
```

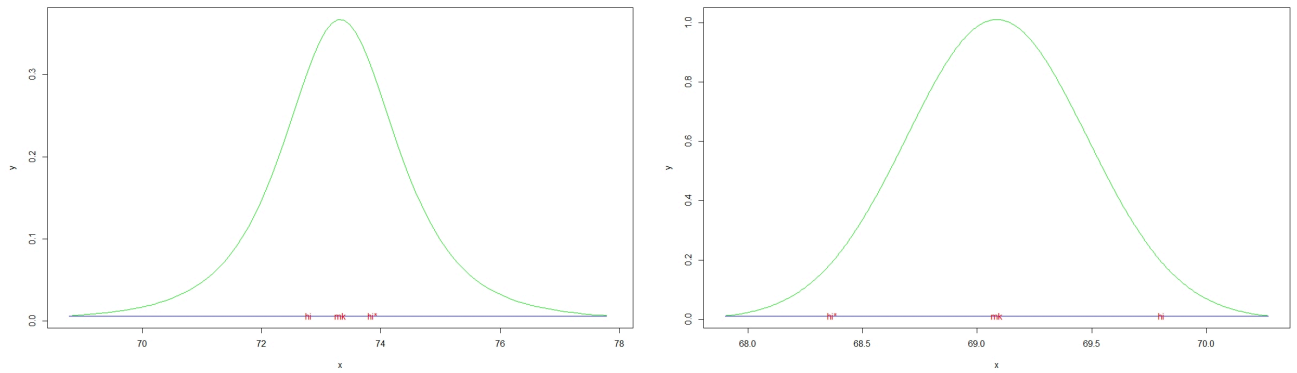
Pero mi decisión significa que siempre hay que pasar la hipótesis mayor entre la opuesta y la original, con lo que el código en el programa principal se complica:

```
38      # A este plot hay que ponerle límites o si no se va a 30 valores de
         ↪  distancia de la media (en n=2)
39      plot(x,y, type="l", col="green",
40           xlim=c(max(mean+bordes[1], mean-20), min(mean+bordes[2], mean+20)))
              ↪  # El límite solo afecta si se pasa
41    }
42    lines(c(min(x),max(x)),c(min(y),min(y)), col="blue")
43    text(media_muestra, min(y),"mk", col="red")
44    text(contrastar, min(y),"hi", col="red")
```

Para visualizar todo mejor, se dibuja la función de distribución de la muestra, esta puede ser tanto una distribución T-Student o normal gaussiana. La gaussiana se dibuja fácilmente debido a que sabemos que el rango $[mean - 3 * dt, mean + 3 * dt]$ incluye el 99,7 % de los datos y a que la función *dnorm* se puede ajustar a una media y d.t. Por otro lado, la T-Student (con grados de libertad $2^k - 1$) presenta más problemas, primero tenemos que calcular el rango que deseamos evaluar, para ello usamos la función *qt* (ver línea 34); una vez calculada la altura con media 0 debemos cambiar el eje x para que refleje la media de la distribución (ver línea 37). Todos estos cálculos están implementados en la función *dibujar_distribucion*:

```r
dibujar_distribucion <- function(distribucion, mean, dt, contrastar, contrastar_opuesto, tamano_muestra) {
  if (distribucion == "NORMAL") {
    x <- seq(mean-3*dt, mean+3*dt, by=dt/50)
    y <- dnorm(x,mean,dt)
    plot(x,y, type="l", col="green")
  } else { # T-Student
    # Selección de rango https://bit.ly/3hx2R0I
    bordes <- qt(c(.003, .997), df=tamano_muestra-1) # Dibujar el 99.7% de la distribución
    x <- seq(bordes[1], bordes[2], 0.1)
    y <- dt(x, df = tamano_muestra-1)
    x <- seq(mean+bordes[1], mean+bordes[2], 0.1) # Ajustamos el eje x a la media (bordes[1] es negativo con lo que sumamos)
    # A este plot hay que ponerle límites o si no se va a 30 valores de distancia de la media (en n=2)
    plot(x,y, type="l", col="green",
         xlim=c(max(mean+bordes[1], mean-20), min(mean+bordes[2], mean+20))) # El límite solo afecta si se pasa
  }
  lines(c(min(x),max(x)),c(min(y),min(y)), col="blue")
  text(media_muestra, min(y),"mk", col="red")
  text(contrastar, min(y),"hi", col="red")
  text(contrastar_opuesto, min(y), "hi*", col="red")
}
```

El dibujo de una distribución ”vacía”queda así:



(a) T-Student (b) Gaussiana

Figura 1: Distribuciones vacías

Lo siguiente que hay que hacer es dibujar el área de p-valor, para ello se realiza el dibujo en dos partes, primero un lado del dibujo y luego el otro lado. Para ello se utiliza la función *rellenar_area* que dibuja desde un valor hacia arriba o hacia abajo dependiendo de los valores que se le pase y de la distribución que se le utilice.

```r
rellenar_area <- function(valor, hacia_arriba, distribucion, media, dt, tamano_muestra) {
  if (distribucion == "NORMAL") {
    # Comprobamos que la hipótesis esté dentro de gráfico
    if (media-3*dt<=valor & valor<=media+3*dt) {
      if (hacia_arriba) {
        # Utilizamos dt/5 como paso para que el espacio entre líneas sea uniforme en todos los gráficos
        ab <- seq(valor, media+3*dt, dt/5)
        r <- dnorm(ab, media, dt)
        lines(ab, r, type = "h", col ="purple")
      } else {
        ab <- seq(media-3*dt, valor, dt/5)
        r <- dnorm(ab, media, dt)
        lines(ab, r, type = "h", col ="purple")
      }
    }
  } else { # T_STUDENT
    bordes <- qt(c(.003, .997), df=tamano_muestra-1) + media
    # Comprobar que está dentro del gráfico
    if (bordes[1]<=valor & valor<=bordes[2]) {
      if (hacia_arriba) {
        ab <- seq(valor, bordes[2], (bordes[2]-media)/25)
        r <- dt(ab-media, df = tamano_muestra-1)
        lines(ab, r, type = "h", col="purple")
      } else {
        ab <- seq(bordes[1], valor, (bordes[2]-media)/25)
        r <- dt(ab-media, df = tamano_muestra-1)
        lines(ab, r, type = "h", col="purple")
      }
    }
  }
}
```

Esto requiere pasar los valores correctos a la función, con lo que necesitamos modificar un poco el llamamiento a la función, hay que llamarla dos veces y hay que llamarla de manera correcta:

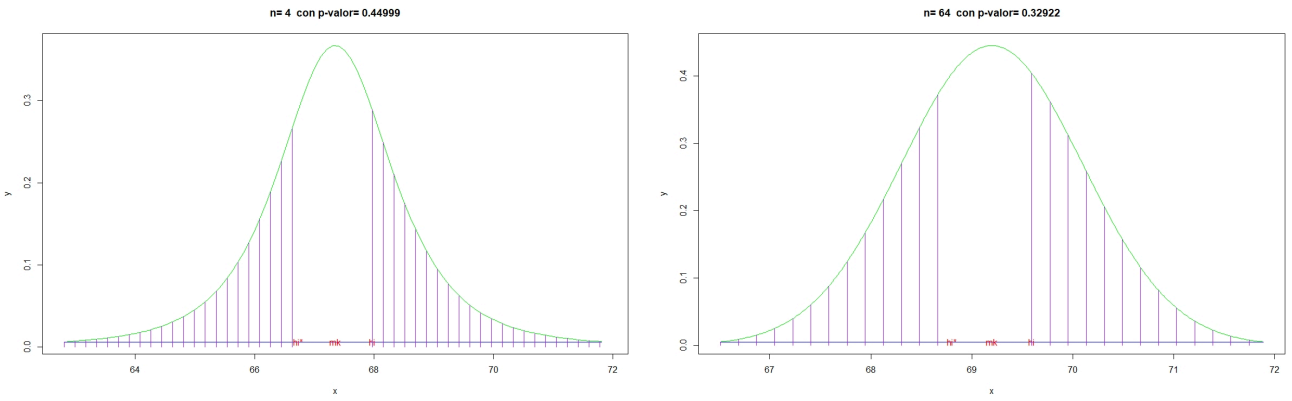
```r
# Escribe el título del gráfico (posiblemente se podría implementar en dibujar_distribucion)
escribir_titulo <- function(tamano_muestra, p_valor) {
  titulo <- paste("n=", tamano_muestra, " con p-valor=", round(p_valor, 5))
  title(titulo)
}

# Las barras verticales moradas que marcan el espacio del p-valor
```

Finalmente se añade un título que identifique al gráfico con la función *escribir_titulo*:

```
49 escribir_titulo <- function(tamano_muestra, p_valor) {
50   titulo <- paste("n=", tamano_muestra, " con p-valor=", round(p_valor, 5))
51   title(titulo)
52 }
```

Y así resultan los gráficos una vez aplicadas las funciones *rellenar_area* y *escribir_titulo*:



(a) T-Student (b) Gaussiana

Figura 2: Distribuciones rellenadas

Una vez dibujado y calculado todo solo queda rechazar o aceptar la hipótesis, para ello se utilizan las siguientes condiciones:

- Si $p - valor > 0{,}01$ se acepta la hipótesis y se termina el proceso.
- Si $0{,}001 < p - valor < 0{,}01$, se mantiene al hipótesis y se aumenta el tamaño de la muestra.
- Si $p - valor < 0{,}001$, se rechaza la hipótesis, se actualiza la media muestral actual y se aumenta el tamaño de la muestra.

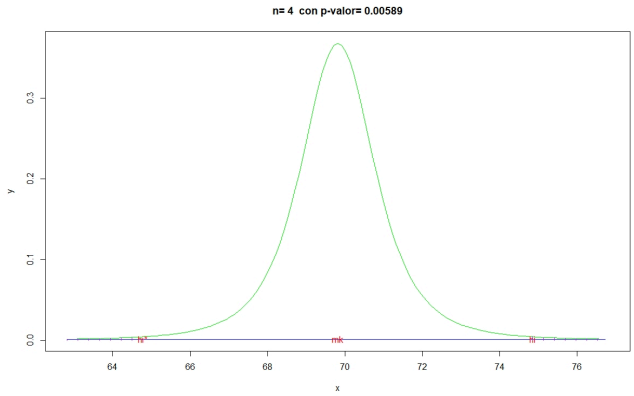
Esto se traduce en el siguiente código:

```
101 decidir_hipotesis <- function(p_valor, mean, hipotesis) {
102   if (p_valor > 0.01) {
103     cat(paste("Aceptamos la hipotesis", round(hipotesis,5), "(media:",
      ↪  round(mean, 5) ,") debido a que ", round(p_valor, 5) ,
      ↪  "es mayor que 0.01.\n"))
104     return(c(FALSE, contrastar))
105   }
106   if (p_valor > 0.001) { # Ya sabemos que tiene que ser < 0.01
107     cat(paste("Mantenemos la hipotesis aumentando la muestra debido a que p-valor es",
      ↪  round(p_valor, 5), ".\n"))
108     return(c(TRUE, contrastar))
109   }
```

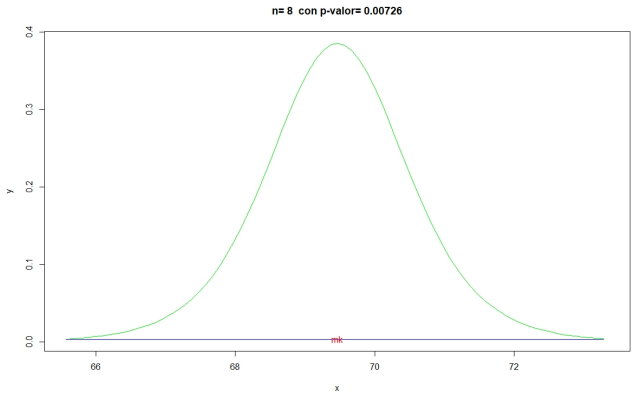
```
110   cat("Rechazamos la hipotesis (",
      ↪  round(hipotesis,5),") y la cambiamos mientras aumentamos la muestra debido a que p
      ↪  round(p_valor, 5), ".\n")
111   return(c(TRUE, mean))
112 }
```

2. Ejemplo de la ejecución

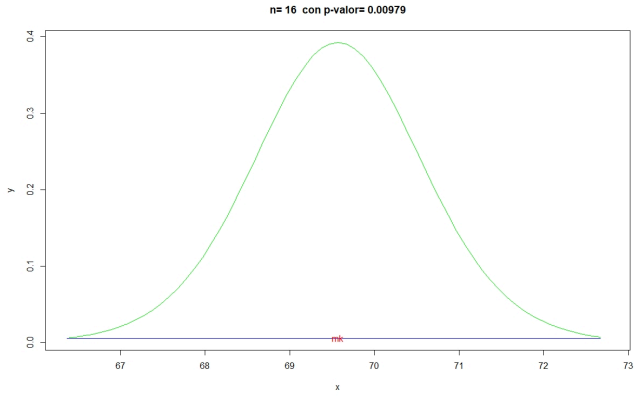
Cuando el programa se ejecuta se siguen los pasos descritos en *Explicación del programa* en el mismo orden que lo expuesto. Se empieza con distribuciones de tipo T-Student y cuando se llega a $k = 5$ ya pasamos a utilizar distribuciones normales. El programa se ejecuta hasta que $k = 6$ (inclusive) ya que en ese momento el p-valor es igual a 0,03858 el cual es superior a 0,01. Da como resultado una hipótesis de 70,23559 con media 68,64111. Los gráficos resultantes y mensajes de consola están en la siguiente página.



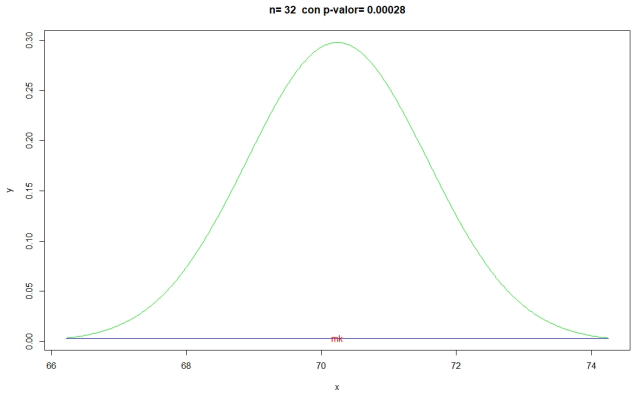
(a) Mantenemos la hipótesis aumentando la muestra.



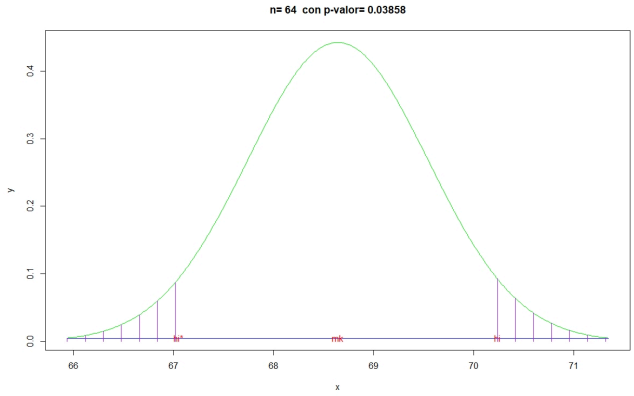
(b) Mantenemos la hipótesis aumentando la muestra.



(c) Mantenemos la hipótesis aumentando la muestra.



(d) Rechazamos la hipótesis y la cambiamos mientras aumentamos la muestra.



(e) Aceptamos la hipótesis 70.23559 (media: 68.64111) debido a que 0.03858 es mayor que 0.01.

Figura 3: Resultados de la ejecución del programa