Análisis Multivariado - Práctica 2, Pt. 2, Ej. 9

Gonzalo Barrera Borla

18/10/2019

Librerias

```
library(mvtnorm) # Distribuciones multivariadas
library(tidyverse) # manipulación de datos en general, graficos
```

2-2-9: Test de Hotelling para muestras T-multivariadas

Se quiere ahora testear $H_0 : \mu = \mu_0$ con $\mu_0 = \mathbf{0}$ cuando $\mathbf{x}_i \sim \mathcal{T}_{p,k}(\mu, \Sigma)$ usando el estadístico

$$T = n\left(\overline{\mathbf{x}} - \mu_0\right)^{\mathrm{T}} \mathbf{S}^{-1} \left(\overline{\mathbf{x}} - \mu_0\right)$$

a. Hacer una simulación para decidir qué resultado obtendría si se usara el percentil de T como si las observaciones fueran normales. Use $k = 1, 2, 4, 10$, $p = 2, 4$ y $n = 20$.
b. Armar un mecanismo bootstrap para testear H_0 en este caso.

A: Simulación

Sabemos que cuando $\mathbf{x}_i \sim \mathcal{N}_p(\mu, \Sigma) \Rightarrow \frac{n-p}{p(n-1)} T^2 \sim \mathcal{F}_{p,n-p}\left(\lambda^2\right)$, y por ende podemos calcular el p-valor correspondiente al T^2_{obs}, p_i en cada simulación. Como el estadístico de Hotelling es invariante bajo transformaciones no-singulares de las $\mathbf{x}_i$, consideramos sin pérdida de generalidad $\Sigma = \mathbf{I}_p$.

Además de simular los valores del estadístico T^2 cuando la muestra es $\mathcal{T}_{p,k}$ para cada tupla (n, p, k) propuesta, consideramos como caso testigo la normal multivariada $\mathcal{N}_p(0, \mathbf{I}_p)$. Como la normal multivariada no tiene grados de libertad como parámetro, nos referimos a ésta con $k = \mathrm{NA}$.

Comenzamos seteamos los parámetros de la simulación

```
nsims <- 1000
rango_k <- c(1, 2, 4, 10, NA)
rango_p <- c(2, 4)
rango_n <- c(20)
```

Genero un data frame vacio para guardar los resultados de la simulacion. Suele ser más eficiente "reservar" el espacio en disco con un data frame de las dimensiones necesarias y `NA` en todas las posiciones, e ir llenándolo, que usar `rbind(df, nuevaFila)` para ir anexando filas reiteradamente a un data frame.

```
vars_df <- c("p", "n", "k", "nsim", "T0sq", "F0", "p.value")
resultados <- data.frame(matrix(
  ncol = length(vars_df),
  nrow = nsims * (length(rango_k) * length(rango_p) * length(rango_n))
))
colnames(resultados) <- vars_df
```

Y ahora generamos `nsims` muestras para cada tupla de parámetros, y guardamos los estadísticos de interés en el data frame. Usamos un índice `i` para ir avanzando la posición en el mismo.

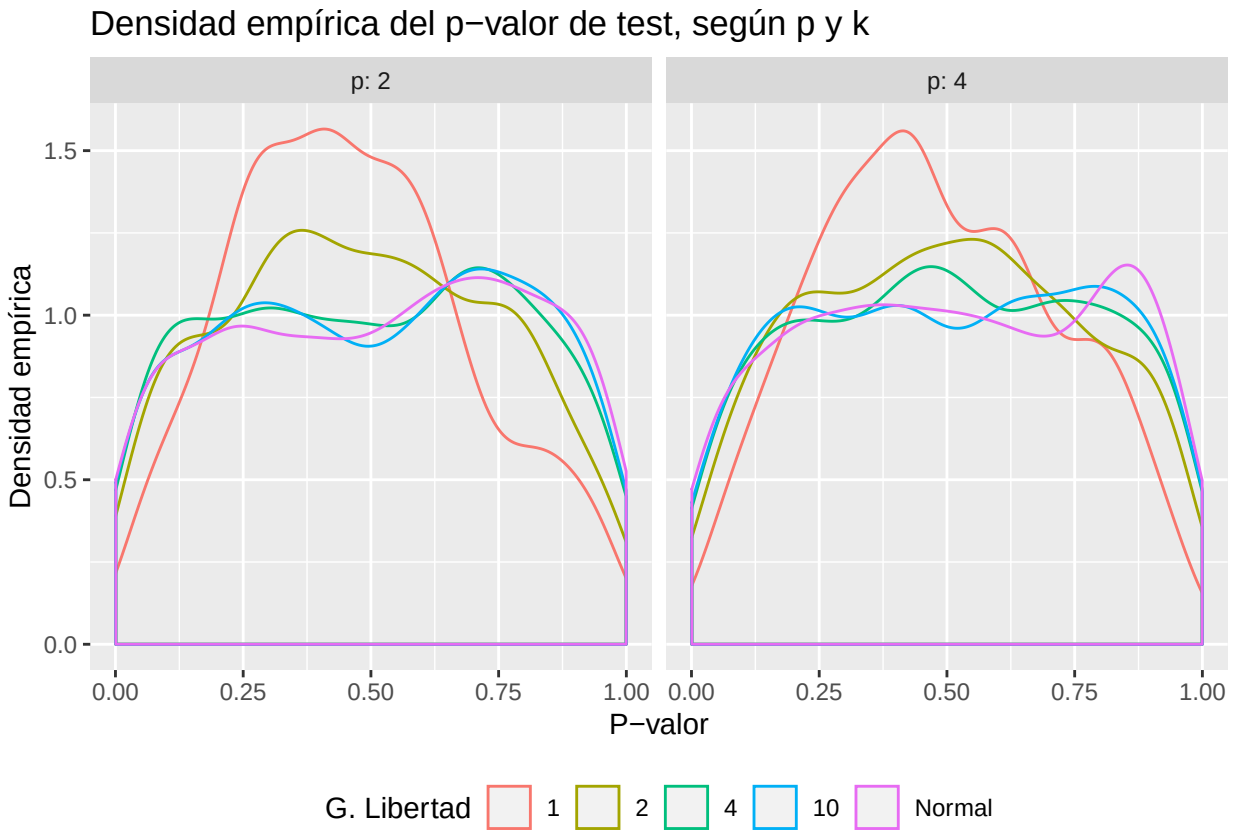
```r
i <- 1 # Índice de fila para los resultados
for (p in rango_p) {
  for (n in rango_n) {
    for (nsim in seq.int(nsims)) {
      for (k in rango_k) {
        set.seed(nsim) # Repito la semilla en cada "ronda" de simulación
        if (is.na(k)) { # muestra Normal de referencia
          X <- rmvnorm(n, sigma = diag(p))
        } else {
          X <- rmvt(n, sigma = diag(p), df = k)
        }
        y <- colMeans(X)
        W <- cov(X)
        T0sq <- n * as.numeric(t(y) %*% solve(W) %*% y)
        F0 <- T0sq * (n-p) / ((n-1) * p)
        p.value <- 1 - pf(F0, p, n - p)

        vals <- c(p, n, k, nsim, T0sq, F0, p.value)
        resultados[i,] <- vals
        i <- i+1
      }
    }
  }
}
```

La tabla resultante contiene los resultados de un simulación por fila. En particular, nos interesa estudiar la distribución de los p-valores.

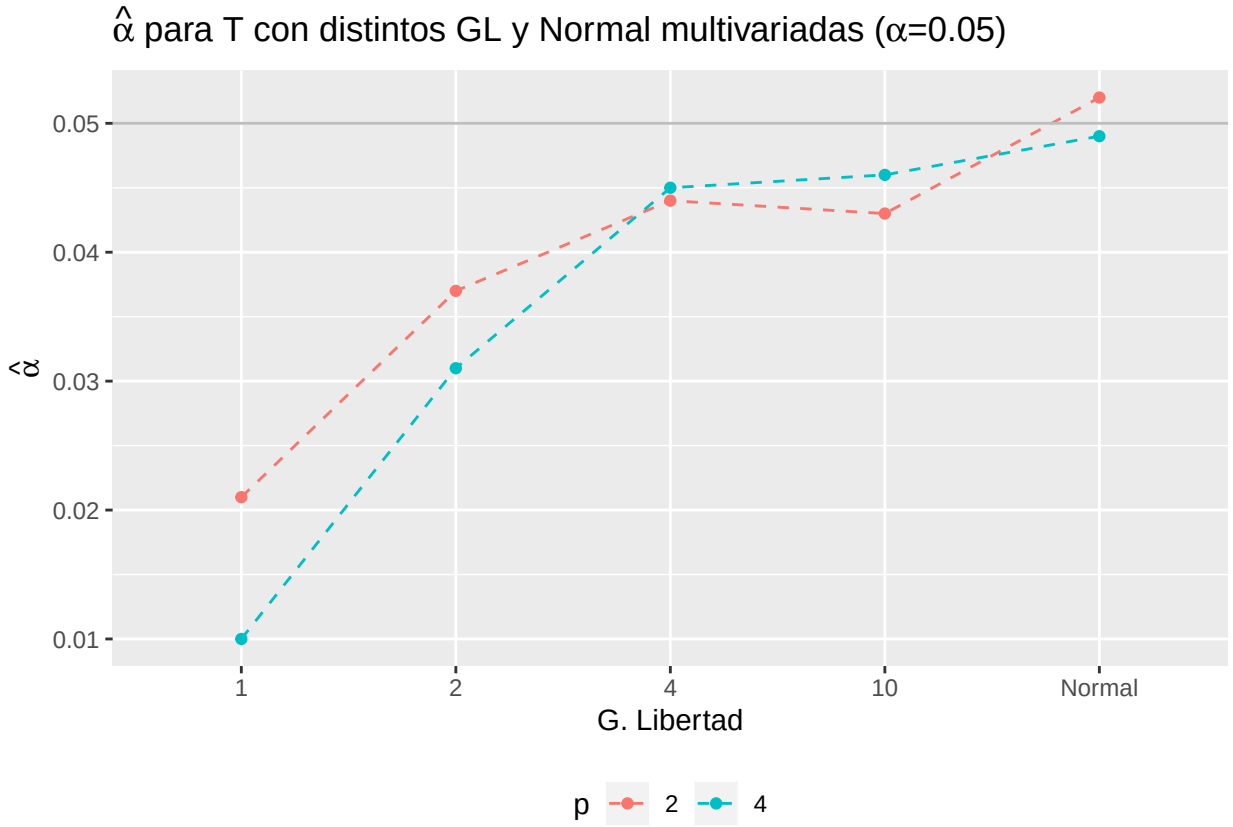
p	n	k	nsim	T0sq	F0	p.value
2	20	1	1	1.900	0.900	0.424
2	20	2	1	0.728	0.345	0.713
2	20	4	1	2.951	1.398	0.273
2	20	10	1	2.478	1.174	0.332
2	20	Normal	1	2.130	1.009	0.384
4	20	1	1	3.748	0.789	0.549
4	20	2	1	4.524	0.953	0.460
4	20	4	1	3.363	0.708	0.598
4	20	10	1	3.080	0.648	0.636
4	20	Normal	1	2.767	0.583	0.680
2	20	1	2	3.702	1.753	0.201
2	20	2	2	6.182	2.928	0.079

Si la distribución del estadísto T_0^2 (`T0sq`) fuese efectivamente Hotelling, los p-valores del test deberían tener una distribución uniforme entre 0 y 1. Recordemos que $k = \text{NA}$ se utilizó para guardar los resultados de la simulación normal de referencia, en cuyo caso *no se utiliza el parámetro k*.



Para $k = 1$, $k = 2$, cuando la varianza de la T multivariada no es finita, la densidad no está uniformemente repartida entre los p-valores. Para k más grande, o cuando la distribución es normal multivariada, la densidad empírica sí pareciera asemejarse a una densidad uniforme en $(0, 1)$.

Para hacer una comparación más directa, podemos elegir un nivel de significación $\alpha = 0.05$, y ver cómo se comparan los niveles de significación empíricos $\hat{\alpha}$ (id est, la cantidad de simulaciones con un p-valor del test menor a α).



Por la distribución poco uniforme de los p-valores para valores bajos de k, el test basado en el estadístico de Hotelling rechaza muy por debajo de la proporción nominal α. Esto es razonable: cuando $k \leq 2 \Rightarrow \text{Var}(\mathbf{x}) = +\infty$, y los intervalos de confianza alrededor de la media muestral serán tan amplios que prácticamente siempre incluirán al cero. A medida que k crece, $\mathcal{T}_{p,k} \xrightarrow{D} \mathcal{N}_p$ y la significación empírica se asemeja a la nominal. Aún así, existe cierta variabilidad alrededor del verdadero valor $\alpha = 0.05$.

B: Bootstrap Paramétrico

Dada una muestra aleatoria observada X_0 donde las observaciones tienen distribución $\mathcal{T}_{p,k}(\mu, \Sigma)$, y una hipótesis nula a testear $H_0 : \mu = \mu_0$ con Σ desconocida, el proceso de bootstrap será el siguiente:

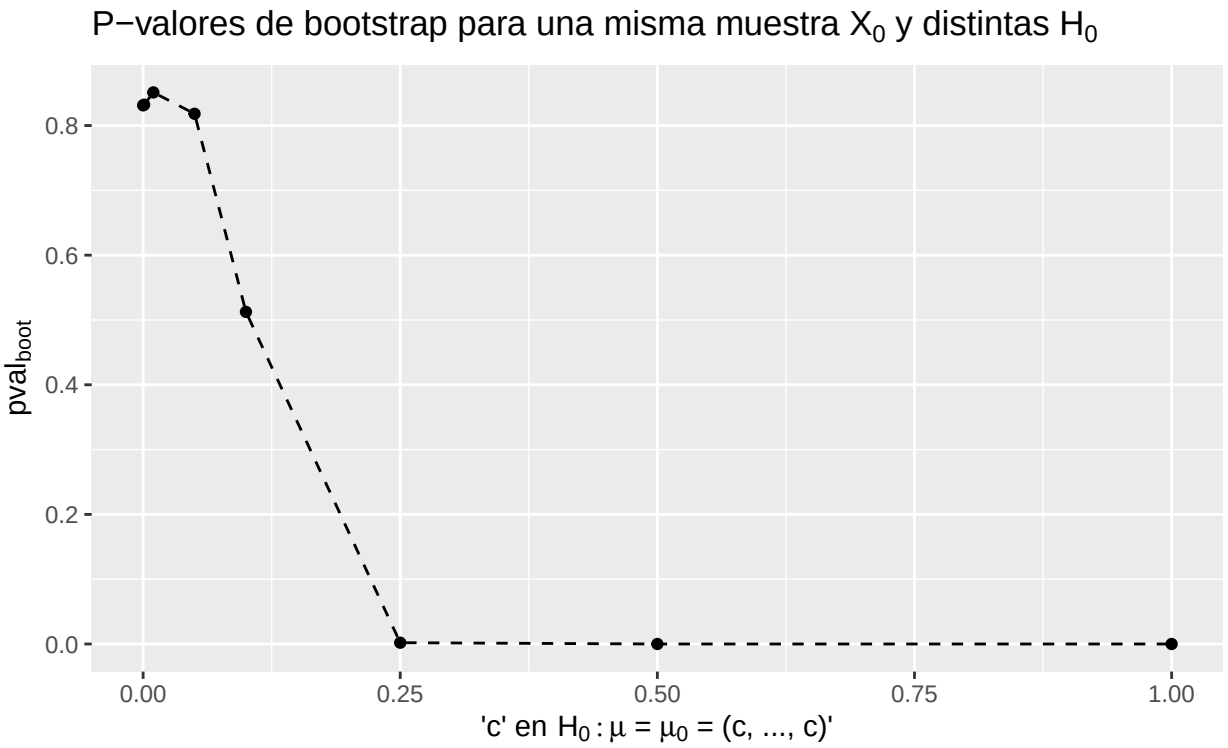
1. Fijamos la semilla, y generamos n_{boot} muestras de tamaño n de una $\mathcal{T}_{p,k}(\mu_0, I_p)$ (como T^2 es invariante bajo transformaciones no-singulares de las observaciones, suponer que la matriz de covarianza es I_p no quita generalidad).
2. Computamos el estadístico T_0^2 para la muestra original y los n_{boot} estadísticos $T_{boot,i}^2$ sobre las muestras sintéticas, y el p-valor asociado al test sobre la muestra original, será

$$\text{p-valor}_{boot} = \frac{\#\{i : T_{boot,i}^2 > T_0^2\}}{n_{boot} + 1}$$

```r
bootstrap.test <- function(X0, k, mu0 = rep(0, dim(X0)[2]), nboot=1000) {
  # X0: matriz muestral (n*p), de VAIID T multivariadas con k GL
  n <- dim(X0)[1]
  p <- dim(X0)[2]
  x_0 <- colMeans(X0)
  S0 <- cov(X0)
  T0 <- as.numeric(n * t(x_0 - mu0) %*% solve(S0) %*% (x_0 - mu0))
  Tboot <- vector("numeric", nboot)
  for (i in seq.int(nboot)) {
    set.seed(i)
    # Bajo H0, las observaciones tienen media mu0
    X <- rmvt(n, delta = mu0, sigma = diag(p))
    x_ <- colMeans(X)
    S <- cov(X)
    Tboot[i] <- as.numeric(n * t(x_ - mu0) %*% solve(S) %*% (x_ - mu0))
  }
  p.value <- sum(Tboot > T0)/(nboot+1)
  return(p.value)
}
```

Para probar esta función, generaremos una muestra $X_0 \sim \mathcal{T}_{2,5}(0, I_2)$ de $n = 100$ observaciones, y a partir de ella testearemos distintas hipótesis nulas $H_{0i} : \mu = \mu_{0i}$. Si el proceso se comporta como uno espera, deberíamos obtener p-valores cercanos a 1 cuando $\mu_{0i} = 0$, y tender a cero a medida que $\|\mu_{0i} - \mu\|$ crece. En otras palabras, esperamos que la potencia del test para distinguir alternativas crezca a medida que éstas se alejan del verdadero valor. Consideraremos en particular la serie de $\mu_{0i} = c_i \times 1_p : c_i = \{0, 0.001, 0.01, 0.05, 0.1, 0.25, 0.5, 1\}$:

```r
set.seed(3)
X0 <- rmvt(n = 100, sigma = diag(2), df = 5, delta = rep(0, 2))
rango_c <- c(0, 0.001, 0.01, 0.05, 0.1, 0.25, 0.5, 1)
pval <- vector("numeric", length(rango_c))
for (i in seq_along(rango_c)) {
  pval[i] <- bootstrap.test(X0, k = 5, mu0 = rep(rango_c[i], 2)) }
```



El test se comporta como uno esperaría, con una potencia creciente (p-valor que disminuye) a medida que nos alejamos del verdadero parámetro $\mu = 0$. Vale aclarar que el máximo p-valor no se alcanza cuando $\mu_0 = \mu = 0$, sino cuando $\mu_0 = \overline{x}$, que en nuestra muestra particular, es 0.091, -0.051.