Taller de Consultoria - TP3

Gonzalo Barrera Borla

23/09/2019

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

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

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. Consideraremos que el test que realizamos está *bien calibrado*, si $\hat{\alpha} \to \alpha$, donde $\hat{\alpha} = \#\{p_i < \alpha \,\forall\, i = 1, ..., \text{nsims}\}$.

Para cada una de las combinaciones sugeridas y algunas mas, realizamos `nsims` simulaciones con una familia t-multivariada, y otras `nsims` con una familia Normal. Si todo está bien programado, para la familia $\mathcal{N}_p$, $\hat{\alpha} \to \alpha$, y nos servirá de testigo para contrastar los resultados de la familia $\mathcal{T}_{p,k}$.

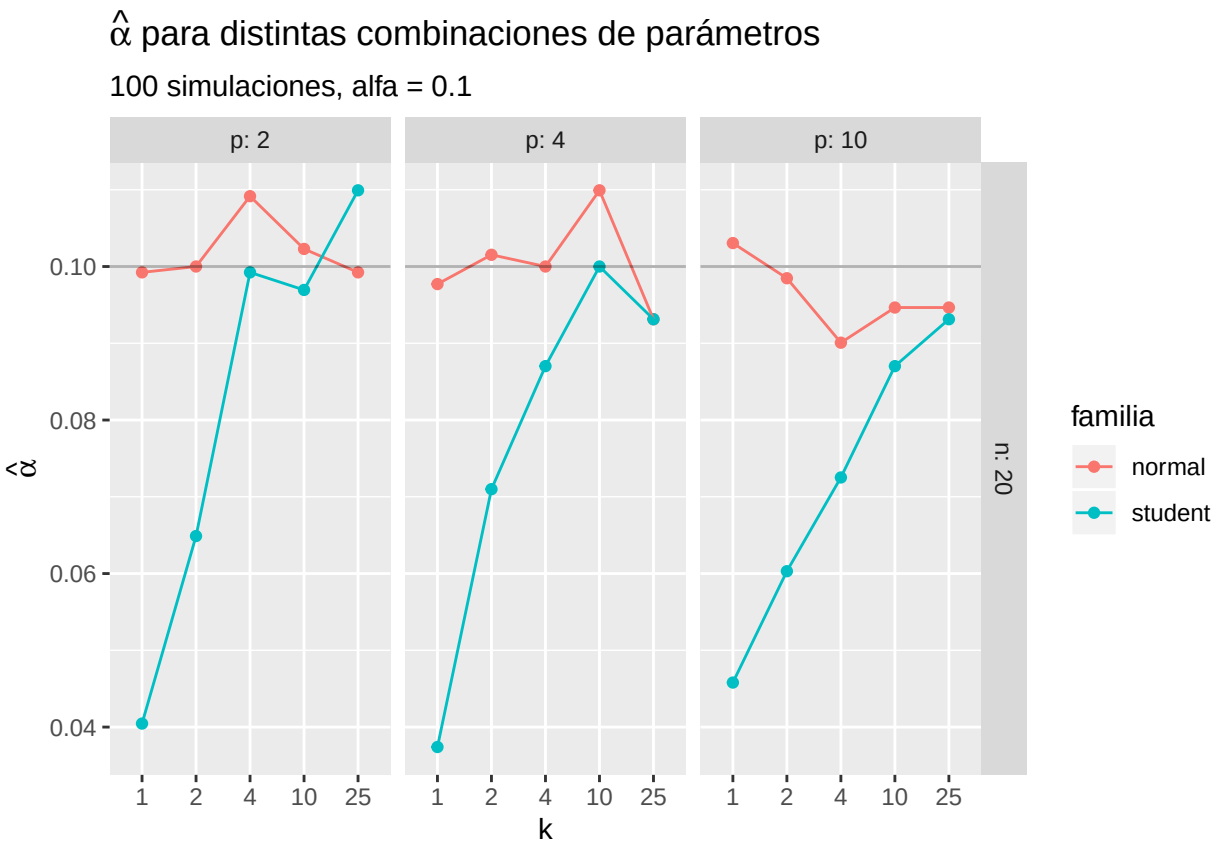
```
ayudante_muestra_normal <- function(n, p, mu=rep(0, p), Sigma=diag(p)) {
  rmvnorm(n, mu, Sigma)
}

ayudante_muestra_t <- function(n, p, k, mu=rep(0, p), Sigma=diag(p)) {
  rmvt(n, Sigma, k, mu, "shifted")
}

# Se computa sin la raíz, que nunca usamos.
# https://en.wikipedia.org/wiki/Mahalanobis_distance
distancia_mahalanobis <- function(x, mu, Sigma) {
  as.double(t(x - mu) %*% solve(Sigma) %*% (x - mu))
}

ayudante_MD_muestral <- function(X, mu0=rep(0,ncol(X))) {
  x_ <- apply(X, 2, mean)
  S <- cov(X)
  distancia_mahalanobis(x_, mu0, S)
}

simular <- function(
  rango_k, # conjunto de GL de T_{p,k} a probar
```

```
rango_p, # conjunto de imensiones posibles de T_{p,k} a probar
rango_nsamp, # conjunto de tamanios muestrales a probar
alfa,    # nivel de significacion del test de Hotelling
nsims=NULL, # numero de simulaciones a realizar. NULL hace infinitas sims
nbatch=min(nsims, 500), # cada cuantas simulaciones guardo los resultados?
memoria='out.csv' # archivo temporal para guardar los resultados
) {

# Funcion auxiliar para acumular los resultados parcales de cada batch
unir_resultados <- function(nuevo, memoria) {
  TIPO_COLS <- cols(
    k = col_integer(),
    p = col_integer(),
    n = col_integer(),
    familia = col_character(),
    q = col_integer(),
    rechazos = col_integer()
  )
  actual <- if (file.exists(memoria)) {
    read_csv(memoria, col_types=TIPO_COLS)
  } else { NULL }

  nuevo <- bind_rows(actual, nuevo) %>%
    group_by(k, p, n, familia) %>%
    summarise_at(c("q", "rechazos"), sum)

  write_csv(nuevo, memoria)
  return(nuevo)
}

sims_ <- 0 # simulaciones total hasta ahora
done <- FALSE # termine?
while (!done) {
  message(paste("Simulaciones totales:", sims_))
  nuevo <- cross_df(list(
    k = rango_k,
    p = rango_p,
    n = rango_nsamp,
    nsim = seq.int(nbatch))) %>%
    mutate(
      student = pmap(list(n, p, k), ayudante_muestra_t),
      normal = map2(n, p, ayudante_muestra_normal)) %>%
    gather("familia", "muestra", -k, -n, -p, -nsim) %>%
    mutate(
      md = map_dbl(muestra, ayudante_MD_muestral),
      Fobs = (n-p)*n/(p*(n-1))*md,
      p_valor = pf(Fobs, df1 = p, df2 = (n-p))) %>%
    group_by(k, p, n, familia) %>%
    summarise(
      q = n(),
      rechazos = sum(p_valor < alfa))

  res <- unir_resultados(nuevo, memoria)
```

```
    sims_ <- sims_ + nbatch
    if (!is.null(nsims) && (sims_ >= nsims)) { done <- TRUE }
  }
  return(res)
}
```

Definimos los parámetros casi tal cual los pide el ejercicio, realizamos la simulación y graficamos los resultados:

```
rango_k <- c(1, 2, 4, 10, 25)
rango_p <- c(2, 4, 10)
rango_nsamp <- 20
alfa <- 0.1
nsims <- 100
resumen <- simular(rango_k, rango_p, rango_nsamp, alfa, nsims)
```



Se observa que el test está bien calibrado (id est, $\hat{\alpha}_n \approx \alpha = 0.1$) para la normal multivariada, pero para distribuciones t-multivariadas con pocos grados de libertad $\hat{\alpha}_n << \alpha$, con $\hat{\alpha}_n \overset{k\to\infty}{\longrightarrow} \alpha$. Esto es razonable, ya que del Ejercicio 2-2-7 sabemos que si

$$\mathbf{x} \sim \mathcal{T}_{p,k}(\mu, \Sigma) \Rightarrow \mathrm{Cov}(\mathbf{x}) = \mathbf{E}(v) \cdot \Sigma = \begin{cases} +\infty & \text{si } k \leq 2 \\ \frac{k}{k-2}\Sigma & \text{si } k > 2 \end{cases}$$

Es decir, la matrix de covarianza "explota" cuando $k \to 2$, y todo intervalo de confianza para μ será tan ancho que aún siendo cierta nunca rechazamos la hipótesis nula $H_0 : \mu = \mu_0$.

B: Bootstrap Paramétrico

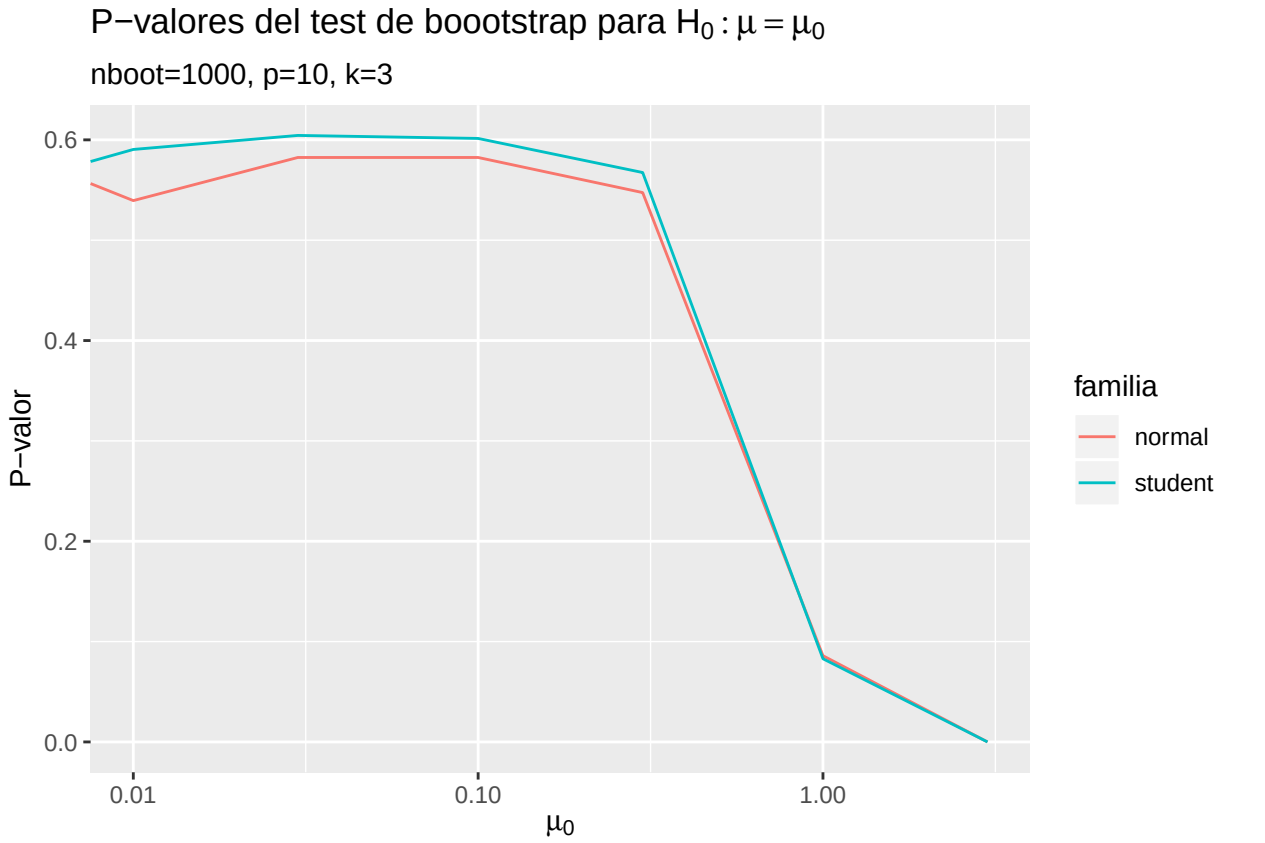
Utilizaremos un proceso de bootstrap paramétrico, pero como de la lectura del ejercicio, no es evidente qué familia de distribuciones utilizar para parametrizar las muestras *bootstrapeadas*, agregamos un parámetro `familia` que nos permite usar la misma función tanto para nuestras normales como T multivariadas, siempre y cuando pasemos un parámetro extra k en caso de usar la T.

```
bootstrap_test <- function(
  X, # matriz muestral de n*p, con n obs. de una distribución p-dimensional
  mu0=rep(0, ncol(X)), # define la hipotesis nula H_0: mu=mu0. 0 por defecto
  familia = "normal", # familia paramétrica: "n"/"normal" o "t"/"student"
  nboot=1000, # Cantidad de muestrar a generar
  ... # Parametros extra. Si familia=="t", debe incluir los `k` GL
  ) {
  extra_kwargs <- names(list(...))
  nsamp <- nrow(X)
  S <- cov(X)
  # No hace falta guardar el estadistico exacto, solo lo que cambia,
  # es decir, la distancia de Mahalanobis al cuadrado


  muestra_H0 <- function(familia, mu0, S, ...) {
    # Bajo H0, las observaciones tienen media mu0 y S estima su covarianza
    if (familia %in% c("normal", "n")) {
      # uso bootstrap parametrico normal multivariado para generar una muestra
      rmvnorm(nsamp, sigma = S, mean = mu0)
    } else if (familia %in% c("t", "student")) {
      if (!"k" %in% extra_kwargs) {
        stop("La familia t requiere un parametro `k` con sus GL")}
      # uso t multivariada con k grados de libertad
      rmvt(nsamp, S,  delta = mu0, df = k, type = "shifted")
    } else {
      stop("No reconozco esa familia de distribuciones")
    }
  }

  # Calculo lo unico que varía entre T^2 y los T^2_{boot}: el cuadrado de la
  # dist. Mahalanobis de la media muestral x_ a mu0
  MD <- ayudante_MD_muestral(X, mu0)
  MDboot <- vector("numeric", nboot)
  for (i in seq.int(nboot)) {
    Xboot <- muestra_H0(familia, mu0, S, ...)
    MDboot[i] <- ayudante_MD_muestral(Xboot, mu0)
  }

  return(list(
    # repito parametros usados
    nboot = nboot,
    mu0 = mu0,
    familia = familia,
    ...,
    S = S,
    x_ = apply(X, 2, mean),
    # estadistico original
    T.sq = nsamp * MD,
```

```
    # Estimo el p-valor como la tasa de Tboot's mayores al T observado
    p_valor = sum(MDboot > MD)/(nboot+1)))
}
```

Con esta función, elegimos unos parametros y generamos una "muestra original" X_0 de t-multivariadas con media en 0, y luego usamos el test bootstrap paramétrico normal y t, para distintas hipótesis nulas. Los valores de μ_0 se construyen tal que $||\mu_0|| = d$:

```
nboot <- 1000
nsamp <- 30
k <- 3
p <- 10
mu <- rep(0, p)
Sigma <- diag(p)
X0 <- ayudante_muestra_t(nsamp, p, k)

# Genera un vector de dimension `p` y norma `d`
alejar <- function(d, p) {
  rep(d/sqrt(p), p)
}
```



El test se porta como uno esperaría, con una probabilidad de rechazo que disminuye a medida que nos alejamos de la verdadera posición central $\mu = \vec{0}$. Resulta interesante notar que la parametrización del test bootstrap con la t multivariada no aporta precisión extra, a costa de agregar un parámetro extra. Luego, parece razonable concluir que de desconocer k, podemos usar un bootstrap con parametrización normal sin mayores inconvenientes.