IECD 2C2024 - Trabajo Práctico

En el trabajo siguiente, estudiaremos el test «de rango signado de Wilcoxon», un test no paramétrico para la mediana de una distribución simétrica. Tendrán que implementarlo de manera compatible con la implementación nativa de *R*, `wilcox.test`, y luego calcular su potencia para alternativas puntuales por el método de *botstrap*. Para ello, introduciremos primero

- `S3`, el paradigma OOP[1] más viejo de R (sí, hay varios) y
- las condiciones de validez y propiedades generales del test de Wilcoxon.

Condiciones de entrega y aprobación

El presente escrito es un apunte sobre OOP y el test de Wilcoxon, con preguntas prácticas diseminadas en medio. Cada pregunta correcta vale tantos puntos como se menciona entre paréntesis en cada una, y **para aprobar es necesario contar con 65 de 114 puntos**.

El TP ha de resolverse en grupos de 3 integrantes. El formato de entrega consistirá de dos archivos subidos a través del campus,

- `informe-<apellido1>-<apellido2>-<apellido3>.pdf`, un informe *en formato PDF* páginas contestando las preguntas teóricas, y
- `codigo-<apellido1>-<apellido2>-<apellido3>.R`, un *script* de R con las respuestas a las preguntas de código, siguiendo estrictamente el formato requerido en cada una.

Tanto el informe como el código se evaluarán con especial énfasis en la **claridad y concisión de exposición, y prolijidad en la presentación**. El código, además, se evaluará de manera automática, a través de una serie de casos de prueba secretos (pero análogos a los que se ofrecen en cada pregunta), que deberán ejecutarse con éxito.

Los gráficos que se piden, deben ser incluidos en el *informe*, y no es necesario incluir el código utilizado para realizarlos.

Hemos decidido eliminar la restricción de longitud, pero por favor, eviten usarlo como licencia de corso para la perorata.

Para el informe pueden usar el procesador de texto que deseen, aunque sugerimos utilizar formatos amigables a la expresión científica, como LaTeX[2], RMarkdown o Typst[3].

La idea del trabajo es que *aprendan unos temas poco comunes*, no que sufran: si no contestan todas las preguntas, está bien. Si quieren saltearse algunas en una primera pasada y volver más tarde, también.

¡Mucha serte!

[1] Programación Orientada a Objetos, por sus siglas en inglés

[2] Si no conocen un buen editor, Overleaf es una excelente primera opción

[3] ¡Este TP está escrito en Typst!

Bibliografía 📚

Lamentablemente la enorme mayoría de la bibliografía de calidad de estos temas está en inglés. A quien se le dificulte la lectura, le recomendamos acudir a cualquier buen traductor o *chatbot* respetable para asistirlo en el proceso.

OOP en R

Esta exposición está recortada arbitrariamente y traducida al castellano de «Advanced R», de Hadley Wickham y equipo, que recomiendo enfáticamente en su totalidad para quienes deseen profundizar sus conocimientos de R. En particular, les sugerimos leer:

- Cap. 12 - Tipos Base
- Cap. 13 - S3 hasta 13.5 «Object Styles» inclusive

Si no tienen ninguna noción de R más allá de «hice unas cositas sueltas para IECD», recomendamos además leer someramente los capítulos 2 («Nombres y Valores») y 3 («Vectores»).

Un recurso un poco más viejo pero repleto de amor y odio por las particularidades de R es R Inferno en el Séptimo Círculo, «Tripping on Object Orientation», cubre someramente y con perpsectiva histórica estos mismos temas.

Lectura de verano ☀️ 🏝️

Para quien desee lectura de verano, los siguientes libros disponibles *online* son de extrema utilidad para el cientista de datos profesional:

- Hands-on Programming With R, una excelente guía general al lenguaje, y luego
- R para Ciencia de Datos (¡en español!) y su original en inglés, más enfocados en las particularidades de la *data saiens*.

Test de Wilcoxon de Rango Signado

Una introducción somera se puede encontrar en Wikipedia: Wilcoxon signed-rank test. El recurso canónico para tests basados en rangos, es «Statistical Inference Based on Ranks», de Thomas P. Hettmansperger (§2, p. 29, 1984, pdf, 6MB). En este último está basada la exposición que sigue. Además,

- El «libro de recetas de cocina»[4] por excelencia es *Practical Nonparametric Statistics*, de W.J. Conover (p.352, 1999, djvu - 9MB, pdf, 124MB).
- Un recurso más moderno del mismo Thomas Hettmansperger, es *Robust Nonparametric Statistical Methods*, de T.P. Hettmansperger y J.W. McKean (p. 38, 2010, pdf, 5MB).

Notación

- $[n] = \{1, ..., n\}$ es el conjunto de los primeros n números naturales
- $C = A/B = \{x : x \in A \wedge x \notin B\}$ es la operación de sustracción de conjuntos,
- la **negrita** ($\underline{X}, \underline{a}, \underline{1}$) denota valores vectoriales, y fuente «normal» para escalares
- usamos mayúsculas ($\underline{Y}, Z$) para elementos aleatorios y minúsculas ($y, \underline{a}$) para no-aleatorios[5].
- $\mathbb{1}\{A\}$ es la función indicadora, que vale 0 cuando A es Falso y 1 cuando A es Verdadero
- Dado un conjunto A, $\#A$ denotará su cardinalidad
- $X \perp Y$ indica que las v.a. (potencialmente multivariadas) X e Y son independientes entre sí

[4] en inglés, *cookbook*, un texto de referencia para practicantes con buenos ejemplos y escasa teoría

[5] sean conocidas o no

OOP en R

Breve intro opinionada

Que los computólogos me juzguen por las barbaridades que estoy por decir. Muy resumidamente, en el paradigma de «Programación Orientada a Objetos», un programa se compone de una serie de interacciones entre *objetos*, que además de poseer ciertos *atributos*, tiene *métodos* que les permiten interactuar entre sí y con otras *clases* de objetos.

En R, hay no una sino al menos 3 «dialectos» para expresarse «en objetos»: S3, S4 y R6[6], de los cuales «S3» es el (a) el más viejo, y (b) el que está *por todas partes* en la implementación base de R que todos amamos y sufrimos por partes iguales. Por ello, **nos concentraremos en S3**.

En OOP, un *objeto* es una *instancia* de una *clase* general que se comporta de cierta manera. R tiene una implementación espectacularmente simple de esta convención:

Definición 1 (Objeto en S3): Un objeto es una *estructura* con un atributo de nombre `class` cuyo valor define la clase del objeto.

```
> objs <- list(mtcars, 1:5, sum, lm(mpg ~ cyl, mtcars), t.test)
> for (obj in objs) { print(class(obj)) }
[1] "data.frame"
[1] "integer"
[1] "function"
[1] "lm"
[1] "function"
```

Pregunta 1 (3 pts.): ¿Qué clase tienen los siguientes vectores: `c(T, F)`, `c(T, F, 1)` y `c(T, F, 1, "1")`? ¿Qué cree que está sucediendo?

Una estructura puede ser *cualquier cosa*, prácticamente, y el atributo se setea con la sintaxis clásica.

```
> lucas <- 1:5
> class(lucas)
[1] "integer"
> attr(lucas, "class") <- "pato"
> class(lucas)
[1] "pato"
```

Una manera más común de definir la clase de un objeto, es usar el constructor `structure`, que hace literalmente lo que necesitamos, asignarle atributos arbitrarios a un objeto cualquiera.

```
> donald <- structure(6:10, class="pato")
> class(donald)
[1] "pato"
```

[6] Wickham explica bien los orígenes de c/u en los capítulos antedichos.

¿Es peligrosa esta filosofía? Sí y no: *puede serlo*, pero sólo si insistimos en asignare clase «petunias» al método `mean` o clase «pato» a un vector de enteros, cosas por el estilo. Aquí, la filosofía de R es «mientras no te taladres los pies, un taladro es una herramienta y no un arma»: exponer las «entrañas» del lenguaje tan abiertamente, hace posible que la comunidad de desarrolladores implemente nuevas clases y métodos con un mínimo de conocimiento sobre sus convenciones.

Métodos «genéricos»

En R, casi todos los métodos de «base» son *genéricos*, que pueden adaptar su comportamiento según la *clase* del primer parámetro que reciben. La convención, *a grosso modo*, dice que cuando se llama un método genérico (como `print`[7]) con primer argumento `obj`, `print(obj)`, R averigua la clase del objeto, `cls <- class(obj)`, y chequea si está definido el método `print.cls`,

- Si lo está, devuelve `print.cls(obj)`, y
- si no, «sigue la cadena de herencia» hasta encontrar una coincidencia o salir por la versión `default`.

Luego, podemos tomar confusas decisiones de diseño, que funcionan de pelos:

```
> print.pato <- function(pato) { "cuac" }
> print(lucas)
[1] "cuac"
```

`sloop::s3_dispatch(llamada)`[8] devuelve una sinopsis de cómo S3 «despachó» la llamada `llamada` al de la clase correspondiente, según su cadena de herencia:

```
> sloop::s3_dispatch(print(lucas))
=> print.pato
 * print.default
> s3_dispatch(mean(1:5))
   mean.integer
   mean.numeric
=> mean.default
```

De igual manera se consigue que `plot(density(1:500))` «automágicamente» plotee la estimación de la densidad por núcleos, sin más:

```
> s3_dispatch(plot(density(1:500)))
=> plot.density
 * plot.default
```

Pregunta 2 (3 pts.): ¿Qué clase tiene `density`? ¿Y `density(1:500)`? ¿Dónde está la diferencia?

---

[7] De `help("print")`: «`print` prints its argument and returns it invisibly (via `invisible(x)`). **It is a generic function which means that new printing methods can be easily added for new classes.**»

[8] `sloop`, de «S Language OOP», es una librería desarrollada por Wickham et al para siplificar el trabajo con los sistemas de clases en R. Como a cualquier librería, a `sloop` se la instala con `install.packages("sloop")` y se la importa con `library(sloop)`.

Introspección: `methods`

Para conocer los métodos a los que sabe despachar cierto genérico `gen`, basta con llamar a `methods("gen")`. Si se quiere conocer todos los métodos asociados con la clase `"cls"`, se invoca `methods(class="cls")`:

```
> methods("plot")[1:8]
[1] "plot,ANY-method" "plot,color-method" "plot.acf" "plot.data.frame"
[5] "plot.decomposed.ts" "plot.default" "plot.dendrogram" "plot.density"
> methods(class="density")
[1] coerce      initialize  plot        print       show        slotsFromS3
see '?methods' for accessing help and source code
```

Las funciones `sloop::s3_methods_generic(gen)` y `s3_methods_class(cls)` retornan la misma información, ordenada más sistemáticamente.

Pregunta 3 (3 pts.): ¿A cuántas clases sabe despachar el genérico `print`? ¿Con cuántos métodos cuenta `density`, además de `plot`?

Fijando ideas: `mi.t.test`

Supongamos que contamos con una muestra $\underline{X} = (X_1, ..., X_n)$ de tamaño n con distribución $X_i \overset{\text{iid}}{\sim} \text{Normal}(\mu, \sigma^2)$ y deseamos testear

$$H_0 : \mu = \mu_0 \quad \text{versus} \quad H_1 : \mu \neq \mu_0$$

Para fijar ideas, supongamos que para nosotros tanto μ como σ^2 son desconocidos cuando en realidad, $X_i \overset{\text{iid}}{\sim} \text{Normal}(1, 1)$, contamos con $n = 30$ y nos interesa testear $\mu_0 = 0, \alpha = 0.05$. En estas circunstancias, el «test T» es el adecuado. En R,

```
# Genero la muestra
mu <- 1
sigma_sq <- 1
n <- 30
X <- rnorm(n, mean = mu, sd = sqrt(sigma_sq))
# Ejecuto el test
mu_0 <- 0
alfa <- 0.05
test_t <- t.test(
  X,
  alternative = "two.sided",
  mu = mu0,
  conf.level = 1 - alfa
)
```

Pregunta 4 (3pts.): Lea `help(unclass)` y conteste: ¿Qué devuelve `class(unclass(test_t))`?¿Por qué?

No es muy difícil reimplementar la lógica detrás de un test T a dos colas como ésta con los conocimientos adquiridos este cuatrimestre. Respetando la convención de nombres de la salida de `t.test`, nos queda:

```r
mi.t.test <- function(x, mu0 = 0, alfa = 0.05) {
  n <- length(x)
  parameter <- n - 1
  estimate <- mean(x)
  stderr <- sd(x) / sqrt(n)
  statistic <- (estimate - mu0) / stderr
  conf.int <- estimate + qt(c(alfa / 2, 1 - alfa / 2), df = parameter) * stderr
  p.value.izq <- pt(statistic, df = parameter)
  p.value <- 2 * min(p.value.izq, 1 - p.value.izq)
  list(
    parameter=parameter,
    estimate=estimate,
    stderr=stderr,
    statistic=statistic,
    conf.int=conf.int,
    p.value=p.value
  )
}
```

La desgracia, es que el `t.test` de R tiene una presentación por defecto bastante informativa:

```
> t.test(X)

  One Sample t-test

data:  X
t = 4.6001, df = 29, p-value = 7.697e-05
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.4391649 1.1422807
sample estimates:
mean of x
0.7907228
```

... y mucho mejor que la de nuestro test:

```
> (mi_test_t <- mi.t.test(X))
$parameter
[1] 29

$estimate
[1] 0.7907228

$stderr
[1] 0.1718916

$statistic
```

```
[1] 4.600124

$conf.int
[1] 0.4391649 1.1422807

$p.value
[1] 7.697055e-05
```

¿Será que `t.test` es instancia de una clase que `print` entiende? ¡Pues claro!

```
> help("s3_dispatch")
> s3_dispatch(print(R_test_t))
=> print.htest
 * print.default
> s3_dispatch(print(mi_test_t))
   print.list
=> print.default
```

En lugar de escribir de cero una función específica de `print` para `mi.t.test`, podemos pararnos en los hombros de gigantes. Le daremos a `mi.t.test` la clase de `t.test`, y veremos de respetar sus convenciones, de manera que podamos utilizar la ya bien pulida `print.htest`[9]. Vamos de nuevo:

```
is.scalar <- function(x) { is.numeric(x) && length(x) == 1 }
mi.t.test <- function(x, mu = 0, conf.level = 0.95) {
  stopifnot(is.numeric(x))
  stopifnot(is.scalar(mu))
  stopifnot(is.scalar(conf.level), (conf.level > 0), (conf.level < 1))
  alfa <- 1 - conf.level
  n <- length(x)
  rv <- list(
    parameter = c(df = n - 1),
    estimate = c(`mean of x` = mean(x)),
    stderr = sd(x) / sqrt(n),
    null.value = c(mean = mu),
    alternative = "two.sided",
    method = "One Sample t-test",
    # Stack Overflow: How to convert variable (object) name into String
    # https://stackoverflow.com/a/14577878
    data.name = deparse(substitute(x))
  )
  rv$statistic <- setNames((rv$estimate - mu) / rv$stderr, "t")
  rv$conf.int <- rv$estimate + qt(c(alfa / 2, 1 - alfa/2), df = rv$parameter)
* rv$stderr
  attr(rv$conf.int, "conf.level") <- conf.level
  pval_izq <- pt(rv$statistic, df = rv$parameter)
  rv$p.value <- 2 * min(pval_izq, 1 - pval_izq)
```

[9]que dicho sea de paso, considera *unos cuantos casos*: link al código

```
  structure(rv, class = "htest")
}
```

Y ahora resulta que:

```
> mi.t.test(X)

  One Sample t-test

data:  X
t = 4.6001, df = 29, p-value = 7.697e-05
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.4391649 1.1422807
sample estimates:
mean of x
0.7907228

> stopifnot(capture.output(t.test(X)) == capture.output(mi.t.test(X)))
```

Observación 1 (`stopifnot`): Cuando uno desea «testear»« condiciones en medio de un programa, `stopifnot` es sumamente útil: recibe varias expresiones, y si alguna *no* evalúa a `TRUE`, devolverá un error. Aquí arriba, nos dice que el output de `t.test` y `mi.t.test` son exactamente iguales.

Test de Wilcoxon de rango signado para una muestra

Introducción

Cuando «la distribución F de X pertenece a la familia normal $\mathcal{N}$»[10], el test T que vimos durante la cursada es uniformemente más potente para hipótesis de la forma:

$$H_0 : \mu \leq \mu_0 \quad \text{versus} \quad H_1 : \mu > \mu_0$$
$$H_0 : \mu \geq \mu_0 \quad \text{versus} \quad H_1 : \mu < \mu_0$$

cunando σ^2 es desconocido. En el «mundo real», el supuesto de normalidad es una hipótesis sumamente ceñida, en tanto consigna la distribución de X a un *modelo paramétrico* específico.

Una familia bastante amplia de distribuciones, está dada por «el conjunto de distribuciones absolutamente continuas, con mediana igual a 0»

$$\Omega_0 = \left\{ F : F \text{ absolutamente continua}, F(0) = \frac{1}{2} \right\}$$

Cuando $X \sim G(x) = F(x - \theta), F \in \Omega_0$, el «test del signo»[11], resulta ser uniformemente más potente para testear la mediana ($G(\theta) = F(\theta - \theta) = F(0) = \frac{1}{2}$) según

[10] es decir, que $X_i \sim F \in \mathcal{N} = \{F_X : X \sim \text{Normal}(\mu, \sigma^2), \mu \in \mathbb{R}, \sigma^2 \in (0, \infty)\}$. Diremos indistintamente que X ó F pertenecen a la familia $\mathcal{N}$

[11] confer Práctica 5, ejercicio 22

$$H_0 : \theta \leq 0 \text{ versus } H_1 : \theta > 0$$

, aunque es cierto que no existen *muchos* tests de nivel dado para esta familia tan amplia. Esta clase de tests, se considera «no paramétricos»[12], en tanto la familia de distribuciones en la que funcionan *no admiten obvias parametrizaciones.*

Una familia «a medio camino» entre $\mathcal{N}$ y Ω_0, es el de las distribuciones simétricas:

Definición 2 (distribución simétrica): una v.a. X con densidad f se dice «simétrica alrededor de θ» si $f(\theta+\delta) = f(\theta-\delta)\ \forall\delta > 0$

Esto nos permite definir $\Omega_s \subset \Omega_0$, el conjunto de las distribuciones simétricas alrededor del 0:

$$\Omega_s = \{F : F \in \Omega_0, F(t) = 1 - F(-t) \forall t \in \mathbb{R}\}$$

Cuando $X \sim F \in \Omega_0$, decimos que «X (o F) es simétrica alrededor del cero». Ahora, $\underline{X} = (X_1, ..., X_n)$ será una muestra aleatoria tomada de $G(x) = F(x - \theta), F \in \Omega_s$, donde la mediana es única, está bien definida y es igual a θ. Sin pérdida de generalidad, nos interesarán, entonces, tests de la forma

$$\begin{aligned} H_0 : \theta \leq 0 \quad &\text{versus} \quad \theta > 0 \\ H_0 : \theta \geq 0 \quad &\text{versus} \quad \theta < 0 \end{aligned}$$

Observación 2: Para testear $H_0 : \theta \leq \theta_0$, basta con definir $Y_i = X_i - \theta_0$ y realizar los test definidos aquí arriba sobre $\underline{Y}$

Observación 3: Si definimos $\mathcal{N} = \{F_X : X \sim \text{Normal}(0, \sigma^2), \sigma^2 > 0\} \Rightarrow \mathcal{N} \subset \Omega_s \subset \Omega_0$.

Wilcoxon (1945, link) planteó un test bastante ingenioso para estas situaciones, que (aunque no lo probaremos), resulta ser uniformemente más potente para distribuciones en Ω_s.

Motivación: diseños experimentales apareados

Existen casos completamente válidos en los lo único que sabemos acerca de una distribución es que es simétrica, y nos interesa testear su mediana. Dicho esto, existe un escenario muy común donde la distribución bajo la hipótesis nula pertenece a Ω_s.

Consideremos la situación en la que tenemos dos tratamientos de interés, A y B[13], que se pueden aplicar a sujetos de una población de interés, y estamos interesados en una respuesta particular después de que se hayan aplicado estos tratamientos.

Sea X la respuesta de un sujeto después de que se le haya aplicado el tratamiento A y sea Y la medida correspondiente para un sujeto después de que se le haya aplicado el tratamiento B. La hipótesis nula natural será

$$\begin{aligned} &H_0 : \text{No hay diferencia en la distribución de } X \text{ e } Y. \\ \text{es decir,}\ \ &H_0 : F_X = F_Y \end{aligned}$$

[12] *nopa*, para los amigos

[13] Entendido de forma amplia, un placebo - o cualquier otro procedimiento de referencia o «control» - también es un tratamiento, y este mismo *setup* permite describir diseños del tipo «tratamiento / control».

Supongamos que tenemos una manera de aparear los sujetos de un estudio. Por ejemplo, disponemos de gemelos idénticos para un estudio en sujetos humanos, compañeros de camada para un estudio en sujetos animales o las dos mitades de una misma pared exterior de una casa para un estudio sobre la durabilidad de pinturas de exterior. En el *diseño por pares* (o apareado), se seleccionan aleatoriamente n *pares* de sujetos de la población de interés. Dentro de cada par, un miembro se asigna aleatoriamente al tratamiento A mientras que el otro recibe el tratamiento B.

Este diseño experimental da como resultado una muestra de pares $(X_1, Y_1), ..., (X_n, Y_n)$. A pesar de que este experimento tiene un vector de respuestas de dimensión dos, el interés está puesto en las diferencias obtenidas: $D_1 = X_1 - Y_1, ..., D_n = X_n - Y_n$, y las $D_1, ..., D_n$ se convierten en la *única* muestra de interés para decidir si los tratamientos se diferencian significativamente.

Bajo la hipótesis nula de que no hay diferencia en el tratamiento (es decir, que la distribución de X es la misma que la distribución de Y) junto con la asignación aleatoria dentro de cada par a recibir el tratamiento A o el B obtenemos una distribución simétrica de las diferencias.

Pregunta 5 (5 pts.): Bajo $H_0 : F_X = F_Y$ (los tratamientos son indistinguibles) y asumiendo que la asignación de cada individuo al tratamiento se realiza de forma aleatoria, la distribución conjunta $F_{X,Y}$ del vector aleatorio (X, Y) es la misma que la del vector (Y, X); es decir, $F_{X,Y} = F_{Y,X}$. Probar que entonces la distribución de $D = X - Y$ es simétrica alrededor del cero.

Habiendo expuesto razonablemente la relevancia de la familia Ω_s, en particular en el contexto de evaluación empírica de «tratamientos» en diseños muestrales «apareados», pasemos a describir el test de Wilcoxon en sí.

Descripción del test

De aquí en más, consideraremos únicamente el escenario de una sola muestra X, con distribución G simétrica.

Consideremos una muestra aleatoria $\underline{X} = (X_1, ..., X_n)$, $X_i \overset{\text{iid}}{\sim} G(t) = F(t - \theta)$. Deseamos encontrar un test para la «locación»[14] θ.

$$H_0 : \theta = 0 \quad \text{versus} \quad H_1 : \theta > 0$$

, con $F \in \Omega_s$.

Antes de introducir el estadístico a emplear, definiremos algunas funciones:

Definición 3 (Función signo):

$$\text{signo}(x) : \mathbb{R} \to \{-1, 0, +1\}, \text{signo}(x) = \begin{cases} -1 \text{ si } x < 0 \\ 0 \quad \text{si } x = 0 \\ +1 \text{ si } x > 0 \end{cases}$$

[14] En distribuciones simétricas, las dos locaciones clásicas, media y mediana, coinciden.

Definición 4 (Función Rango): Sea $\underline{X} = (X_1, ..., X_n)$ una muestra aleatoria, y $X^{(1)}, ..., X^{(n)}$ la misma muestra, ordenada en forma no decreciente. Llamaremos el *rango*[15] de X_i, al índice de la posición que ocupa en la muestra ordenada:

$$R_i = \text{Rango}(X_i \mid \underline{X}) = j \Leftrightarrow X_i = X^{(j)}$$

Observación 4: Una manera de calcular el rango de una observación, es

$$R_i = \#\{X : X \leq X_i, X \in \underline{X}\} = \sum_{j=1}^{n} \mathbb{1}\{X_j \leq X_i\}$$

Es decir, la función Rango toma como input un vector $\underline{x}$, y devuelve otro vector $\underline{r}$ que es una permutación de $[n] = \{1, ..., n\}$, donde el i-ésimo elemento indica la contidad de elementos de $\underline{x}$ menores o iguales a x_i, o lo que es lo mismo, su posición ordinal.

Llamemos $|\underline{X}| = (|X_1|, |X_2|, ..., |X_n|)$ al vector de valores absolutos de $\underline{x}$, y $R_i = \text{Rango}(|X_i|)$ el rango de $|X_i|$ en $|\underline{X}|$. Ahora sí, podemos presentar una primera versión del estadístico del test de rangos signados de Wilcoxon:

$$T(\underline{X}) = \sum_{i=1}^{n} \text{signo}(X_i) R_i$$

donde por ser X_i v.a. absolutamente continuas e independientes entre sí,

$$\Pr(X_i = 0) = 0 \ \forall i \in [n]$$
$$\Pr(X_i = X_j) = 0 \ \forall \ i, j \in [n], i \neq j$$

de manera que no hay empates ni $\text{signo}(X_i) = 0$.

Pregunta 6 (3 pts.): Muestre que los siguientes estadísticos:

$$T^+ = \sum_{i=1}^{n} \mathbb{1}\{X_i > 0\} R_i$$
$$T^- = \sum_{i=1}^{n} \mathbb{1}\{X_i < 0\} R_i$$

son equivalentes a T (i.e., muestre que a partir de cualquiera de los 3 y conociendo n, se pueden computar exactamente los otros dos). *Sugerencia: calcule* $T^+ + T^-$

De las tres formas, la que más comúnmente se usa para definir el test, es T^+ será la que consideremos de aquí en más. Nuestro test será de la forma

$$\phi(\underline{X}) = \mathbb{1}\{T^+ > k\}$$

, de manera rechazaremos la hipótesis nula cuando la suma de los rangos de los $X_i > 0$ sea lo suficientemente grande, dándole peso a la hipótesis alternativa de que $\theta > 0$.

[15] como en el *rango* militar, donde «general» está por encima de «capitán», que está por encima de «oficial», etc.

Para facilitar el estudio de la distribución de T^+ bajo H_0, introduciremos una última - lo juro - variante en la notación.

Definición 5 (Antirrango): Sean $R_1, ..., R_n$ los rangos correspondientes a un vector $\underline{m} = m_1, ..., m_n$, o sea que $R_i = j \Leftrightarrow m_i = m^{(j)}$. Diremos entonces que el j-ésimo *antirrango* es igual a i. En otras palabras, el antirrango es «la inversa» del rango: $D_j = i$ si el j-ésimo elemento de la muestra ordenada es el i-ésimo en la muestra original:

$$D_j = i \Leftrightarrow R_i = j$$

Por ejemplo, si $\underline{m} = (m_1, m_2, m_3)$, con $m_2 < m_3 < m_1$, resulta que

$$R_1 = R(m_1 \mid \underline{m}) = 3 \qquad D_1 = 2 \quad \text{pues} \quad m^{(1)} = m_2$$
$$R_2 = R(m_2 \mid \underline{m}) = 1 \qquad D_2 = 3 \quad \text{pues} \quad m^{(2)} = m_3$$
$$R_3 = R(m_3 \mid \underline{m}) = 2 \qquad D_3 = 1 \quad \text{pues} \quad m^{(3)} = m_1$$

Habiendo definido los antirrangos, podemos reescribir el estadístico T^+ de la siguiente manera:

$$T^+ = \sum_{i \in [n]} \mathbb{1}\{X_i > 0\} R_i = \sum_{D_j \in [n]} \mathbb{1}\left\{X_{D_j} > 0\right\} R_{D_j} = \sum_{j \in [n]} W_j \times j$$

donde $W_j = \mathbb{1}\left\{X_{D_j} > 0\right\} = \frac{\text{signo}\left(X_{D_j}\right)+1}{2}$ y por definición, $R_{D_j} = R\left(|X_{D_j}|\right) = j$.

Distribución de T^+ bajo la hipótesis nula

Ahora sí estamos en condiciones de estudiar la distribución de T^+ bajo H_0.

Pregunta 7 (5 pts.): Demuestre que bajo H_0, $|X_i|$ es independiente de $\text{signo}(X_i)$.
Sugerencia: utilice sus conocimientos sobre F_X bajo H_0.

Pregunta 8 (3 pts.): A partir del resultado anterior, pruebe que bajo H_0 los vectores de rangos $\underline{R} = (R_1, ..., R_n)$ y antirrangos $\underline{D} = (D_1, ..., D_n)$ correspondientes a $|\underline{X}|$ son independientes del vector de signos $\underline{S} = (\text{signo}(X_1), ..., \text{signo}(X_n))$ de la muestra original $\underline{X}$.

Pregunta 9 (5 pts.): Pruebe que bajo $H_0 : \theta = 0, F \in \Omega_s$, las v.a. $W_j = \mathbb{1}\left\{X_{D_j} > 0\right\}$ distribuyen según

$$W_1, ..., W_n \overset{\text{iid}}{\sim} \text{Bernoulli}\left(\frac{1}{2}\right)$$

Sugerencia: Utilice la <u>Ley de la Probabilidad Total</u> de $\underline{W}$ sobre los posibles valores de $\underline{D}$, y utilice los resultados previos para operar y factorizar $\Pr(\underline{W} = \underline{w}) = \prod_{j \in [n]} \Pr\left(W_j = w_j\right)$.

De los ítems anteriores, se desprende que bajo H_0, **la distribución de T^+ es una suma de variables aleatorias independientes, aunque no idénticamente distribuidas**.

Observación 5: Lo único que utilizamos para hallar la distribución bajo H_0 de T^+ fue que $X_i \overset{\text{iid}}{\sim} F(x-\theta), F \in \Omega_s$. Como la distribución de T^+ será la misma para *cualquier* $F \in \Omega_s$, se dice que T^+ es de *distribución libre* («distribution-free») bajo H_0

Distribución exacta de T^+

Aunque la distribución de T^+ no tiene forma cerrada, su cómputo exhaustivo no es particularmente difícil. Para simplificar la notación, omitiremos la dependencia de la probabilidad a H_0 hasta próximo aviso.

Sabemos que T^+ es una v.a. discreta con soporte en los enteros desde 0 hasta $\frac{n(n+1)}{2}$. Queremos hallar

$$p_n(t) = \Pr(T^+ = t) = \Pr\left(\sum_{j\in[n]} \left(W_j \times j\right) = t\right)$$

Llamemos $A_{n,t} = \{\underline{w} : T^+ = t\}$ al conjunto posibles valores de $\underline{w} = (w_1, ..., w_n) \in \{0,1\}^n$ tal que

$$T^+ = \sum_{j\in[n]} \left(w_j \times j\right) = \sum_{j:w_j=1} j = t$$

. Ya sabemos de la respuesta a Pregunta 9 que $\Pr(\underline{W} = \underline{w}) = \frac{1}{2^n}\ \forall\ \underline{w} \in \{0,1\}^n$, y luego

$$\Pr(T^+ = t) = \Pr\left(\underline{w} \in A_{n,t}\right) = \frac{\# A_{n,t}}{2^n}$$

donde según, $\# A_t$ es la cardinalidad de A «conjunto potencia» de todas las 2^n combinaciones posibles de signos.

Observación 6: Existe una equivalencia natural entre los vectores $\underline{w} = (w_1, ..., w_n)$ y los conjuntos $s(\underline{w}) = \{i : w_i = 1\}$ que conservan únicamente los índices no-nulos de $\underline{w}$, de manera que $A_{n,t}$ y $S_{n,t} = \left\{s(\underline{w}) : \underline{w} \in A_{n,t}\right\}$ tienen la misma cardinalidad. Luego, podemos escribir

$$p_n(t) = \frac{\#S_{n,t}}{2^n}$$

Cuando $n = 4$ por ejemplo, resulta que

t	$S_{4,t}$	$\#S_{4,t}$	$p_4(t)$
0	$\{\emptyset\}$	1	1/16
1	$\{\{1\}\}$	1	1/16
2	$\{\{2\}\}$	1	1/16
3	$\{\{3\},\{1,2\}\}$	2	2/16
4	$\{\{4\},\{1,3\}\}$	2	2/16
5	$\{\{1,4\},\{2,3\}\}$	2	2/16
6	$\{\{2,4\},\{1,2,3\}\}$	2	2/16
7	$\{\{3,4\},\{1,2,4\}\}$	2	2/16
8	$\{\{1,3,4\}\}$	1	1/16
9	$\{\{2,3,4\}\}$	1	1/16
10	$\{\{1,2,3,4\}\}$	1	1/16

Pregunta 10 (5pts.): Reproduzca la tabla de Observación 6 para $n = 5$

]

Pregunta 11 (5 pts.): Muestre que T^+ es simétrica alrededor de $\frac{n(n+1)}{4}$

Fórmula recursiva para n «grande»

Ya para un valor moderado como $n = 20$, $2^n = 1.048.576$: el crecimiento exponencial de los valores posibles de $\underline{W}$ vuelve inconcecible el cómputo «exhaustivo» de p_{T^+}. El hecho de que T^+ sea simétrica simplificaría algo las cuentas, pero no demasiado. Sin embargo, existe una fórmula recursiva particularmente interesante:

Supongamos que $s \in S_{n,t}$, de manera que $\sum_{x \in s} x = t$. O bien $n \in s$, o bien $n \notin s$:

- si $n \notin s$, entonces $s \subseteq [n-1]$ y suma t; luego $s \in S_{n-1,t}$,
- si $n \in s$, entonces $s - \{n\} \subseteq [n-1]$ y suma $t - n$; luego $s - \{n\} \in S_{n-1,t-n}$.

Como ambas son mutuamente excluyentes, y se cumplen para cada $s \in S_{n,t}$, si llamamos $u_n(t)$ a la cardinalidad de $S_{n,t}$,

$$u_n(t) = u_{n-1}(t) + u_{n-1}(t-n)$$

Con los límites de la recursión en

$$u_0(t) = \begin{cases} 1 \text{ si } t = 0 \\ 0 \text{ caso contrario} \end{cases}$$

$$u_n(t) = 0 \quad \text{si} \quad t < 0 \quad \text{ó} \quad t > \frac{n(n+1)}{2}$$

Luego,

$$p_n(t) = \frac{\#S_{n,t}}{2^n} = \frac{u_n(t)}{2^n}$$

$$p_n(t) = \frac{1}{2}\left[\frac{u_{n-1}(t)}{2^{n-1}} + \frac{u_{n-1}(t-n)}{2^{n-1}}\right]$$

$$p_n(t) = \frac{1}{2}[p_{n-1}(t) + p_{n-1}(t-n)]$$

Pregunta 12 (8 pts.): Programe la rescursión $u_n(t)$ en R. Llámela `particiones`, y dele dos argumentos, t, n, ambos enteros.

```
particiones <- function(t, n) {
  # su código aquí
}
```

La función debe pasar al menos los siguientes tests:

```
stopifnot(
  particiones(t=3, n=4) == 2,
  particiones(t=24, n=12) == 67,
  particiones(t=55, n=10) == 1,
  particiones(t=45, n=30) == 1938
)
```

Pregunta 13 (8 pts.): Usando `particiones`, implemente `dTmas(x, n)` y `pTmas(x, n)` que toman un vector de enteros `x` y un escalar `n`, y den, respectivamente, la función de probabilidad puntual y la función de distribución de T^+ bajo H_0 en cada valor de `x`.

```
dTmas <- function(x, n) {
  ret <- vector(mode = "numeric", length = length(x))
  for (i in seq_along(x)) {
    # Su código aquí
  }
  return(???)
}
pTmas <- function(x, n) { "repita el patrón de dTmas" }
```

Al menos los siguientes casos de test deben pasar:

```
n <- 15
t <- 34
stopifnot(
  dTmas(24, 12) == 67 / 2 ^ 12,
  dTmas(0:10, 4) == c(1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1) / 16,
  sum(dTmas(0:21, 6)) == 1,
  dTmas(0:2, 55) == 2 ^ -55,
  dTmas(t, n) == dTmas(n * (n + 1) / 2 - t, n),
  pTmas(t, n) == 1 - pTmas(n * (n + 1) / 2 - (t + 1), n)
)
```

Nota: ¡Ojo! `particiones` *espera un escalar como primer argumento* `t`*, mientras que* `pTmas` *y* `dTmas` *esperan vectores.*

`mi.wilcox.test`

Ahora sí, estamos en condiciones de implementar el test de Wilcoxon de rango signado. Repasemos: Para una muestra aleatoria

$$\underline{X} = (X_1, ..., X_n),\ X_i \overset{\text{iid}}{\sim} F(x - \theta)\ \forall i \in [n],\ F \in \Omega_s$$

, deseamos testear:

$$\begin{aligned}
\text{('two.sided') igual contra distinto:} \quad & H_0 : \theta = \theta_0 \quad \text{versus} \quad H_0 : \theta \neq \theta_0 \\
\text{('greater') menor o igual contra mayor:} \quad & H_0 : \theta \leq \theta_0 \quad \text{versus} \quad H_0 : \theta > \theta_0 \\
\text{('less') mayor o igual contra menor:} \quad & H_0 : \theta \geq \theta_0 \quad \text{versus} \quad H_0 : \theta < \theta_0
\end{aligned}$$

Pregunta 14 (15 pts.): Programe `mi.wilcox.test`, una función con la misma clase que `wilcox.test`, que toma los siguientes parámetros (siguiendo la firma de `wilcox.test`):

- `x`, un vector numérico con la muestra $\underline{X}$,
- `alternative`, un escalar de tipo `"character"` representando las hipótesis a testear (una de `c("two.sided", "greater", "less")`),
- `mu`, un escalar numérico representando θ_0, el valor de la mediana bajo la hipótesis nula.

y devuelve un objeto de clase `"htest"`, con (al menos) atributos `statistic`, `p.value` y `alternative` equivalentes a los salida de `wilcox.test`.

No hace falta reportar una región de rechazo, basta con el p-valor. Al menos el siguiente caso de prueba debe pasar:

```r
set.seed(1234)
n <- 20
X <- rnorm(n)
theta0 <- -1
alternative <- "greater"

R_wilcox <- wilcox.test(X, alternative=alternative, mu = theta0)
mi_wilcox <- mi.wilcox.test(X, alternative=alternative, mu = theta0)
stopifnot(
  identical(mi_wilcox$statistic, R_wilcox$statistic),
  identical(mi_wilcox$alternative, R_wilcox$alternative),
  isTRUE(all.equal(mi_wilcox$p.value, R_wilcox$p.value)),
  identical(class(R_wilcox), class(mi_wilcox))
)
```

Sugerencia: Consulte la ayuda de `match.arg` para manipular el valor de `alternative`.

Distribución asintótica

Aunque T^+ es una combinación lineal de variables independientes e idénticamente distribuidas entre sí (las W_j), cada una está pesada por un coeficiente distinto (los $j \in [n]$), por lo cual la versión del Teorema Central del Límite que conocemos no nos servirá.

Pregunta 15 (5 pts.): Bajo H_0, ¿Cuánto vale $\mathbb{E}(T^+)$? ¿Y $\mathrm{Var}(T^+)$?

La «condición de Lindeberg)» nos dota de una forma un poco más general del T.C.L., que ahí adaptamos del Apéndice «A9» de Hettmansperger (1984)[16]

[16]cf. página 301 del libro o p. 317 del PDF para la prueba

Teorema 1 (TCL de Lindeberg): Sean $W_1, ..., W_n$ v.a. i.i.d. con $\mathbb{E}(W_1) = 0$, $\text{Var}(W_1) = \sigma^2$, $0 < \sigma^2 < \infty$. Defínase $S = \sum_{i=1}^n a_i W_i / \sqrt{n}$. Si

$$\frac{\max_i |a_i|}{\sqrt{\sum_{i=1}^n a_i^2}} \to 0$$

entonce $\frac{S}{\sqrt{\text{Var}(S)}} \xrightarrow{\mathcal{D}} \text{Normal}(0, 1)$, con $\text{Var}(S) = \sigma^2 \left(\sum_{i=1}^n a_i^2\right) / n$.

Pregunta 16 (7 pts.): Dé la distribución asintótica de T^+

Pregunta 17 (8 pts.): Fije $n_1 = 4, n_2 = 10, n_3 = 20$. Para cada n, realice un gráfico de barras con la probabilidad puntual *exacta* de T^+ (válgase de `dTmas`) y superpóngale una línea con la densidad asintótica esperada. ¿Coinciden razonablemente? ¿En toda la distribución, en el centro, en las colas? ¿A partir de qué n? ¿Se le ocurre alguna corrección sencilla para los n pequeños?

Distribución bajo la alternativa vía *bootstrap*

Bajo H_1, F no es simétrica alrededor de 0 sino de algún otro valor, con lo cual los rangos no serán independientes de los signos, y la distribución del estadístico T^+ no cuenta con forma cerrada.

Sin embargo, conociendo el proceso generador de los datos (o DGP[17]), es posible calcular la potencia para una alternativa puntual, con el procedimiento de *bootstrap* paramétrico. Asuma el ambiente de test ya habitual, $X_i \overset{\text{iid}}{\sim} F(x - \theta), F \in \Omega_s$, y queremos testear a nivel menor o igual a α[18]:

$$H_0 : \theta = 0 \text{ versus } H_1 : \theta = \theta_1 > 0$$

El test resultará

$$\phi(\underline{X}) = \mathbb{1}\{T^+ > k^*\}, \mathbb{E}_0(\phi) = \alpha^* \leq \alpha$$

donde elegimos k^* para maximizar la potencia del test, respetando el nivel $\leq \alpha$.

Sea ahora $H(x) = F(x - \theta_1)$ la verdadera distribución del DGP, simétrica y con mediana igual a θ_1. Si el DGP es conocido, el siguiente procedimiento nos da un método para estimar la potencia $\pi_\phi(\theta_1)$:

1. Genere m muestras de tamaño n de la distribución H; llamémoslas $\underline{Y}_1, ..., \underline{Y}_m$
2. Compute $T^+(\underline{Y}_i)\ \forall i \in [m]$; guarde los resultados en un vector.
3. El estimador por bootstrap de $\pi_\phi(\theta_1)$ está dado por

$$\widehat{\pi_\phi}(\theta_1) = \widehat{\mathbb{E}_{\theta_1}}(\phi) = m^{-1} \sum_{i=1}^m \mathbb{1}\{T^+(\underline{Y}_i) > k^*\}$$

[17] *Data Generating Process*, por sus siglas en inglés

[18] Recuerden que como el estadístico T^+ es discreto, el nivel que alcance nuestro test no será exactamente α, sino el mayor $\alpha^* \leq \alpha$ que la distribución permita.

Observación 7: En este *setup*, el procedimiento de bootstrap es «paramétrico», en tanto la distribución del DGP está parametrizada y por ende podemos *samplear* de ella directamente. A diferencia del procedimiento visto en clase en el que había que estimar $\hat{\theta}_1$ y luego samplear de $F_{\hat{\theta}_1} = F\left(x - \hat{\theta}_1\right)$, aquí el enunciado provee el θ_1 real, así que su estimación es innecesaria y podemos samplear directamente de $H(x) = F(x - \theta_1)$. Por lo demás, el procedimiento es el mismo.

Pregunta 18 (13 pts.): Los siguientes datos fueron generados por $D = \text{Normal}(1, 1)$, $n = 12$:

```r
set.seed(1984)
n <- 12
theta1 <- 1
sigma_sq <- 1
X <- rnorm(n, mean=theta1, sd=sqrt(sigma_sq))
```

Compute ϕ_w, el test de Wilcoxon de rango signado de nivel menor o igual a $\alpha = 0.05$ para las hipótesis:

$$H_0 : \theta = 0 \quad \text{versus} \quad H_1 : \theta > 0$$

A continuación, fije $\theta_1 = 1, m = 10.000$ y estime por bootstrap la potencia $\widehat{\pi_{\phi_w}}(\theta_1)$.

Pregunta 19 (8 pts.): Compute para las mismas hipótesis y condiciones de Pregunta 18, ϕ_n, un test para el valor de la media (y mediana) de v.a.i.i.d. con varianza conocida, según D.

Calcule (analíticamente, sin estimar) la potencia $\pi_{\phi_n}(\theta_1)$. Compute además ϕ_s, el test del signo para las mismas hipótesis y estime por bootstrap, $\widehat{\pi_{\phi_s}}(\theta_1)$. Compare y contraste los resultados obtenidos. ¿Es el test t efectivamente el más potente? ¿Por cuánto?

(0pts., sólo para valientes) Sin asumir varianza conocida, hay que recurrir al «test t». Describa ϕ_t, el «test t» correspondiente a esta situación, calcule su potencia para la alternativa θ_1 e inclúyalo en la comparación con (ϕ_w, ϕ_n, ϕ_s).

Sugerencia: considere la distribución «t de Student no-central». Para el cálculo de potencia, de necesitarlo, sí puede utilizar el verdadero σ.

Nota: Al igual que con ϕ_w, tenga cuidado de proveer un test del signo ϕ_s exacto[19] *de nivel tan cercano a $\alpha = 0.05$ como pueda, pero sin pasarse.*

[19]En la Práctica 5 Ej. 22 se da un test del signo asintótico. Si se le complica deducir el equivalente exacto, en Hettmansperger (1984) la sección §1.2 describe «El test del Signo y Su Distribución».