EST-24107: Simulación

Profesor: Alfredo Garbuno Iñigo — Otoño, 2022 — Intervalos *Bootstrap*.
Objetivo: En la sección anterior vimos cómo construir con remuestreo una distribución alrededor de un estimador basado en una muestra. En esta sección veremos cómo resumir dicha distribución en una estimación por intervalos.
Lectura recomendada: El libro de [2] es una buena introducción al tema.

1. ERROR ESTÁNDAR BOOTSTRAP E INTERVALOS NORMALES

Ahora podemos construir nuestra primera versión de intervalos de confianza basados en la distribución *bootstrap*.

- Supongamos que queremos estimar una cantidad poblacional θ con una estadística $\hat{\theta} = s(X_1, \ldots, X_N)$, donde $X_1, \ldots, X_N$ es una muestra independiente e idénticamente distribuida de la población.
- Suponemos además que la distribución muestral de $\hat{\theta}$ es aproximadamente normal (el teorema central del límite aplica), y está centrada en el verdadero valor poblacional θ.

Ahora queremos construir un intervalo que tenga probabilidad 95 % de cubrir al valor poblacional θ. Tenemos que

$$\mathsf{Prob}\left(-2\mathsf{ee}(\hat{\theta}) < \hat{\theta} - \theta < 2\mathsf{ee}(\hat{\theta})\right) \approx 0.95\,, \tag{1}$$

por las propiedades de la distribución normal ($\mathsf{Prob}(-2\sigma < X - \mu < 2\sigma) \approx 0.95$ si X es normal con media μ y desviación estándar σ).

Es decir, la probabilidad de que el verdadero valor poblacional θ esté en el intervalo

$$[\hat{\theta} - 2\mathsf{ee}(\hat{\theta}), \hat{\theta} + 2\mathsf{ee}(\hat{\theta})]$$

es cercano a 0.95. En este intervalo no conocemos el error estándar (es la desviación estándar de la distribución de muestreo de $\hat{\theta}$), y aquí es donde entre la distribución *bootstrap*, que aproxima la distribución de muestreo (en términos de varianza). Lo estimamos con

$$\hat{\mathsf{ee}}_{\mathsf{boot}}(\hat{\theta})\,, \tag{2}$$

que es la desviación estándar de la **distribución *bootstrap***.

1.0.1. ***Definición*** *[Intervalos* boostrap *normales]:* El **error estándar *bootstrap*** $\hat{\mathsf{ee}}_{\mathrm{boot}}(\hat{\theta})$ se define como la desviación estándar de la distribución bootstrap de θ. El **intervalo de confianza normal *bootstrap*** al 95 % está dado por

$$[\hat{\theta} - 2\hat{\mathsf{ee}}_{\mathsf{boot}}(\hat{\theta}), \hat{\theta} + 2\hat{\mathsf{ee}}_{\mathsf{boot}}(\hat{\theta})]\,. \tag{3}$$

Nótese que hay varias cosas qué checar aquí: que el teorema central del límite aplica y que la distribución de muestreo de nuestro estimador está centrado en el valor verdadero. Esto en algunos casos se puede demostrar usando la teoría, pero más abajo veremos comprobaciones empíricas.

1.1. Ejemplo

Consideremos la estimación que hicimos de el procentaje de tomadores de té que toma té negro:

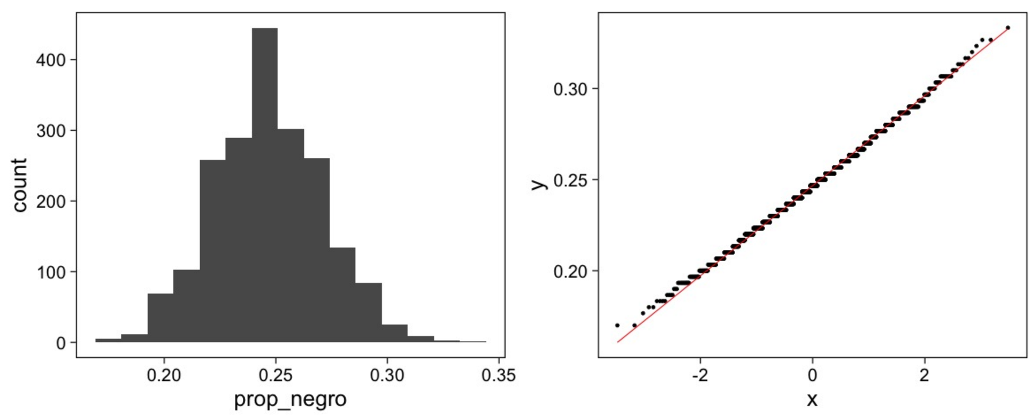
```
te ← read_csv("data/tea.csv") ▷
  rowid_to_column() ▷
  select(rowid, Tea, sugar)
```

```
## paso 1: define el estimador
calc_estimador ← function(datos){
  prop_negro ← datos ▷
    mutate(negro = ifelse(Tea == "black", 1, 0)) ▷
    summarise(prop_negro = mean(negro), n = length(negro)) ▷
    pull(prop_negro)
  prop_negro
}
```

```
## calcula el estimador
prop_hat ← calc_estimador(te)
prop_hat ▷ round(4)
```

```
[1] 0.2467
```

Podemos graficar su distribución bootstrap —la cual simulamos arriba—.



Y notamos que la distribución *bootstrap* es aproximadamente normal.

Adicionalmente, vemos que el sesgo tiene un valor estimado de:

```
prop_negro_tbl ← read_rds("cache/prop_negro_tbl.rds")
media_boot ← prop_negro_tbl ▷ pull(prop_negro) ▷ mean()
media_boot - prop_hat
```

```
[1] -0.00021333
```

De esta forma, hemos verificado que:

- La distribución *bootstrap* es aproximadamente normal (ver gráfica de cuantiles normales);
- La distribución *bootstrap* es aproximadamente insesgada.

Lo cual nos lleva a construir intervalos de confianza basados en la distribución normal. Estimamos el error estándar con la desviación estándar de la distribución *bootstrap*

```
ee_boot <- prop_negro_tbl |> pull(prop_negro) |> sd()
ee_boot
```

```
[1] 0.024537
```

y construimos un intervalo de confianza del 95 %:

```
intervalo_95 <- c(inf = prop_hat - 2 * ee_boot,
                  centro = prop_hat,
                  sup = prop_hat + 2 * ee_boot)
intervalo_95 |> round(3)
```

```
  inf centro    sup
0.198  0.247  0.296
```

Este intervalo tiene probabilidad del 95 % de capturar al verdadero poblacional.

2. INVENTARIOS DE CASAS VENDIDAS

Ahora consideremos el problema de estimar el total del valor de las casas vendidas en un periodo. Igual que antes, tenemos una muestra de tamaño $n = 200$. Pero ahora utilizaremos el paquete `rsample` para realizar las estimaciones el método *bootstrap*.

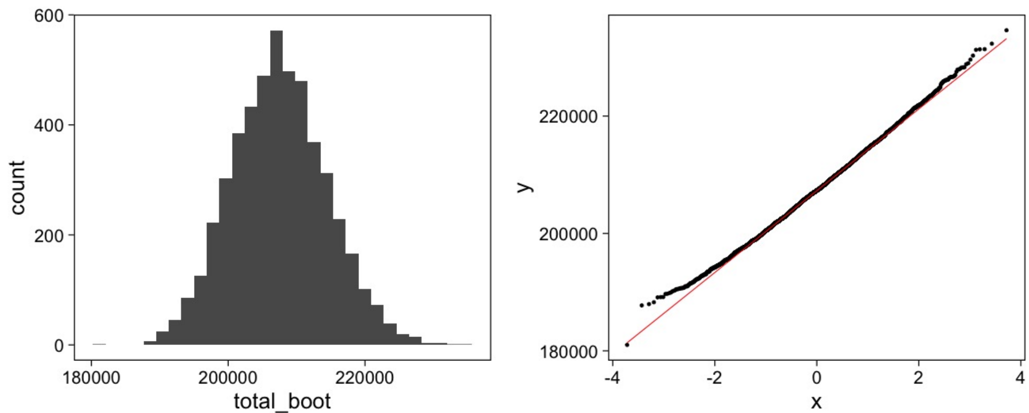
```
## muestra original
set.seed(121)
poblacion_casas <- read_csv("data/casas.csv")
muestra_casas   <- read_rds("cache/casas_muestra.rds")
## paso 1: define el estimador
estimador_lote <- function(split, ...){
  N <- nrow(poblacion_casas)
  muestra <- analysis(split)
  muestra |>
    summarise(estimate = (N / n()) * sum(precio_miles)) |>
    mutate(term = "Valor lote")
}
```

```
totales_boot <- bootstraps(muestra_casas,  5000) |>  ## paso 2 y 3
  mutate(res_boot = map(splits, estimador_lote))     ## paso 4
```

La función `rsample::bootstraps` utiliza la estructura del `tidyverse`. Esto es por que genera un `tibble` con celdas de distintos tipos de objetos.

```
totales_boot
```

```
# Bootstrap sampling
# A tibble: 5,000 × 3
   splits            id            res_boot
   <list>            <chr>         <list>
 1 <split [200/69]> Bootstrap0001 <tibble [1 × 2]>
 2 <split [200/68]> Bootstrap0002 <tibble [1 × 2]>
 3 <split [200/70]> Bootstrap0003 <tibble [1 × 2]>
 4 <split [200/73]> Bootstrap0004 <tibble [1 × 2]>
 5 <split [200/69]> Bootstrap0005 <tibble [1 × 2]>
 6 <split [200/77]> Bootstrap0006 <tibble [1 × 2]>
 7 <split [200/69]> Bootstrap0007 <tibble [1 × 2]>
 8 <split [200/68]> Bootstrap0008 <tibble [1 × 2]>
 9 <split [200/72]> Bootstrap0009 <tibble [1 × 2]>
10 <split [200/83]> Bootstrap0010 <tibble [1 × 2]>
# ... with 4,990 more rows
#  Use `print(n = ...)` to see more rows
```



En este caso, la distribución de muestreo presenta cierta asimetría, pero la desviación no es grande. En la parte central la aproximación normal es razonable. Procedemos a checar sesgo:

Primero necesitamos calcular el valor del estimador de la muestra original

```
estimador.obs <- muestra_casas |>
  summarise(estimador = (nrow(poblacion_casas)/n() * sum(precio_miles))) |>
  pull(estimador)
estimador.obs
```

```
[1] 207431
```

Después necesitamos la media *bootstrap* para poder calcular el sesgo

```
resumen_boot <- totales_boot |>
  unnest(res_boot) |>
  summarise(media.boot = mean(estimate)) |>
  mutate(sesgo = media.boot - estimador.obs)
resumen_boot
```

```
# A tibble: 1 × 2
  media.boot sesgo
```

```
        <dbl> <dbl>
1    207461.  30.5
```

Este número puede parecer grande, pero si calculamos la diferencia relativa con respecto al estimador vemos que es chico en la escala de la distribución *bootstrap*:

```
resumen_boot |>
  mutate(sesgo_relativo = sesgo / estimador.obs)
```

```
# A tibble: 1 × 3
  media.boot sesgo sesgo_relativo
        <dbl> <dbl>          <dbl>
1    207461.  30.5       0.000147
```

De forma que procedemos a construir intervalos de confianza como sigue :

```
intervalos_normales <- totales_boot |>
  unnest(res_boot) |>
  summarise(media_boot = mean(estimate), ee_boot = sd(estimate)) |>
  mutate(inf = media_boot - 2 * ee_boot, sup = media_boot + 2 * ee_boot)
intervalos_normales
```

```
# A tibble: 1 × 4
  media_boot ee_boot     inf     sup
        <dbl>   <dbl>   <dbl>   <dbl>
1    207461.   6933. 193596. 221327.
```

Que está en miles de dólares. En millones de dólares, este intervalo es:

```
intervalos_normales / 1000
```

```
  media_boot ee_boot   inf    sup
1     207.46  6.9327 193.6 221.33
```

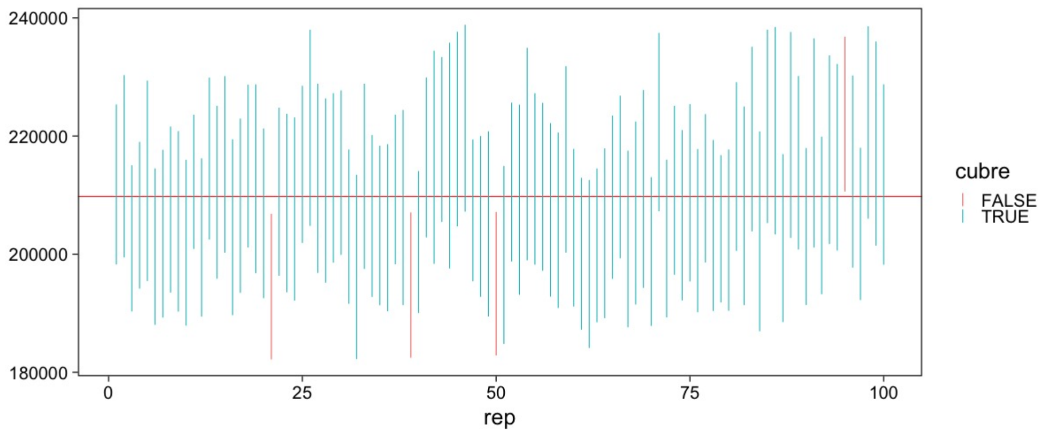
2.0.1. `Nota:` En el siguiente ejemplo mostraremos una alternativa de intervalos de confianza que es más apropiado cuando observamos asimetría. Sin embargo, primero tendremos que hablar de dos conceptos clave con respecto a intervalos de confianza: calibración e interpretación.

3. CALIBRACIÓN DE INTERVALOS DE CONFIANZA

¿Cómo sabemos que nuestros intervalos de confianza del 95 % nominal tienen cobertura real de 95 %? Es decir, tenemos que checar:

- El procedimiento para construir intervalos debe dar intervalos tales que el valor poblacional está en el intervalo de confianza para 95 % de las muestras.

Como solo tenemos una muestra, la calibración depende de argumentos teóricos o estudios de simulación previos. Para nuestro ejemplo de casas tenemos la población, así que podemos checar qué cobertura real tienen los intervalos normales:



La cobertura para estos 100 intervalos simulados da

```
total <- sum(poblacion_casas$precio_miles)
sims_tbl |>
  summarise(cobertura = mean(cubre))
```

```
# A tibble: 1 × 1
  cobertura
      <dbl>
1      0.96
```

que es **consistente** con una cobertura real del 95 % (¿qué significa "consistente"? ¿Cómo puedes checarlo con el *bootstrap*?)

3.0.1. ***Observación:*** En este caso teníamos la población real, y pudimos verificar la cobertura de nuestros intervalos. En general no la tenemos. Estos ejercicios de simulación se pueden hacer con poblaciones sintéticas que se generen con las características que creemos va a tener nuestra población (por ejemplo, sesgo, colas largas, etc.).

> En general, no importa qué tipo de estimadores o intervalos de confianza usemos, requerimos checar la calibración. Esto puede hacerse con ejercicios de simulación con poblaciones sintéticas y tanto los procedimientos de muestreo como los tamaños de muestra que nos interesa usar.

Verificar la cobertura de nuestros intervalos de confianza por medio simulación está bien estudiado para algunos casos. Por ejemplo, cuando trabajamos con estimaciones para poblaciones teóricas. En general sabemos que los procedimientos funcionan bien en casos:

- con distribuciones simétricas que tengan colas no muy largas;
- estimación de proporciones donde no tratamos con casos raros o casos seguros (probabilidades cercanas a 0 o 1).

4. INTERPRETACIÓN INTERVALOS DE CONFIANZA

Como hemos visto, "intervalo de confianza" (de 90 % de confianza, por ejemplo) es un término **frecuentista**, que significa:

- **Cada muestra produce un intervalo distinto**. Para el 90 % de las muestras posibles, el intervalo cubre al valor poblacional.
- La afirmación es **sobre el intervalo y el mecanismo para construirlo.**

- Así que con **alta probabilidad**, el intervalo contiene el valor poblacional.
- Intervalos más anchos nos dan más incertidumbre acerca de dónde está el verdadero valor poblacional (y al revés para intervalos más angostos).

Existen también "intervalos de credibilidad" (de 90 % de probabilidad, por ejemplo), que se interpetan de forma **bayesiana**:

- Con 90 % de probabilidad (relativamente alta), creemos que el valor poblacional está dentro del intervalo de credibilidad.

Esta última interpretación es más natural. Obsérvese que para hablar de intervalos de confianza frecuentista tenemos que decir:

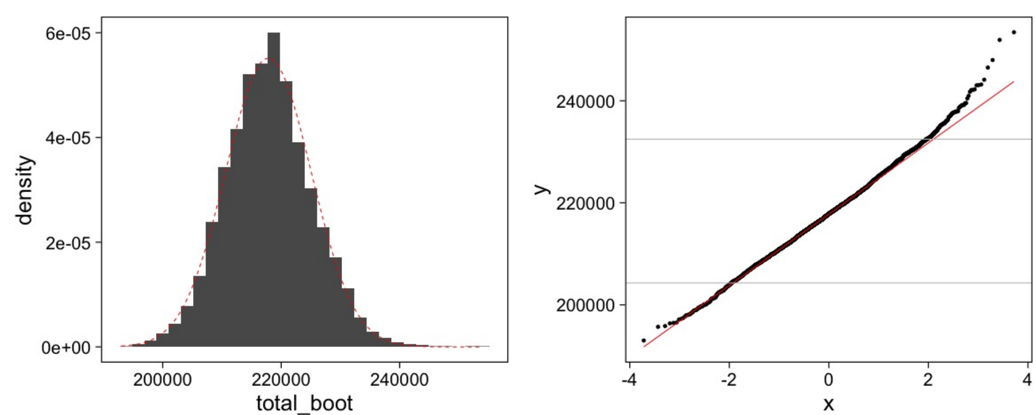
- Este intervalo particular cubre o no al verdadero valor, pero nuestro procedimiento produce intervalos que contiene el verdadero valor para el 90 % de las muestras.
- Esta es una interpretación relativamente débil, y muchos intervalos poco útiles pueden satisfacerla.
- La interpretación bayesiana es más natural porque expresa más claramente incertidumbre acerca del valor poblacional.

Sin embargo, la interpretación frecuentista nos da maneras empíricas de probar si los intervalos de confianza están bien calibrados o no: es un mínimo que "intervalos del 90 %" deberían satisfacer.

Así que tomamos el punto de vista bayesiano en la intepretación, pero buscamos que nuestros intervalos cumplan o aproximen bien garantías frecuentistas (discutimos esto más adelante). Los intervalos que producimos en esta sección pueden interpretarse de las dos maneras.

5. INTERVALOS BOOTSTRAP DE PERCENTILES

Retomemos nuestro ejemplo del valor total del precio de las casas. A través de remuestras bootstrap hemos verificado gráficamente que la distribución de remuestreo es **ligeramente** asimétrica (ver la figura de abajo).



Anteriormente hemos calculado intervalos de confianza basados en supuestos normales por medio del error éstandar. Este intervalo está dado por

```
intervalos_normales / 1000
```

```
  media_boot ee_boot   inf     sup
1     207.46  6.9327 193.6  221.33
```

y por construcción sabemos que es simétrico con respecto al valor estimado, pero como podemos ver la distribución de muestreo no es simétrica, lo cual podemos confirmar por ejemplo calculando el porcentaje de muestras bootstrap que caen por arriba y por debajo del intervalo construido:

```
totales_boot ▷ unnest(res_boot) ▷
  mutate(upper = estimate ≥ max(intervalos_normales$sup),
         lower = estimate ≤ min(intervalos_normales$inf)) ▷
  summarise(prop_inf = mean(lower),
            prop_sup = mean(upper))
```

```
# A tibble: 1 × 2
  prop_inf prop_sup
     <dbl>    <dbl>
1   0.0178   0.0284
```

los cuales se han calculado como el porcentaje de medias *bootstrap* por debajo (arriba) de la cota inferior (superior), y vemos que no coinciden con el nivel de confianza preestablecido (2.5 % para cada extremo).

Otra opción común que se usa específicamente cuando la distribución bootstrap no es muy cercana a la normal son los intervalos de percentiles *bootstrap*:

5.0.1. `Definición` *[intervalos* bootstrap *de percentiles]:* El **intervalo de percentiles *bootstrap*** al 95 % de confianza está dado por

$$[q_{0.025}, q_{0.975}]\,, \tag{4}$$

donde q_f es el percentil f de la distribución *bootstrap*. Es decir el intervalo de $1 - 2\alpha$ está dado por

$$[\hat{\theta}_{\mathsf{inf}}, \hat{\theta}_{\mathsf{sup}}] = [\hat{\theta}^{*(\alpha)}, \hat{\theta}^{*(1-\alpha)}]\,, \tag{5}$$

donde $\hat{\theta}^{*(\alpha)}$ es el `estadístico de orden` de nuestras estimaciones *bootstrap* $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$.

> Nota que estamos aproximando los percentiles utilizando nuestra muestra *bootstrap* observada $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$ pues en teoría deberíamos de utilizar la distribución *bootstrap* ideal (aquella con $B \to \infty$) y que hemos denotado por $\hat{\theta}^{*}$. En este sentido, seguimos utilizando el principio de *plug-in* para construir nuestros estimadores.

Otros intervalos comunes son el de 80 % o 90 % de confianza, por ejemplo, que corresponden a $[q_{0.10}, q_{0.90}]$ y $[q_{0.05}, q_{0.95}]$. **Ojo**: intervalos de confianza muy alta (por ejemplo 99.5 %) pueden tener mala calibración o ser muy variables en su longitud pues dependen del comportamiento en las colas de la distribución.

Para el ejemplo de las casas, calcularíamos simplemente

```
intervalo_95 ← totales_boot ▷ unnest(res_boot) ▷
  pull(estimate) ▷
  quantile(probs = c(0.025, 0.50, 0.975))
intervalo_95 / 1000
```

```
  2.5%    50%  97.5%
194.46 207.32 221.53
```

que está en millones de dólares. Nótese que es similar al intervalo de error estándar.

Otro punto interesante sobre los intervalos *bootstrap* de percentiles es que lidian naturalmente con la asímetría de la distribución bootstrap. Ilustramos esto con la distancia de las extremos del intervalo con respecto a la media:

```
abs(intervalo_95 - estimador.obs)/1000
```

```
     2.5%       50%    97.5%
12.97571  0.11143 14.10036
```

Los intervalos de confianza nos permiten presentar un rango de valores posibles para el parámetro de interés. Esto es una notable diferencia con respecto a presentar sólo un candidato como estimador. Nuestra fuente de información son los datos. Es por esto que si vemos valores muy chicos (grandes) en nuestra muestra, el intervalo se tiene que extender a la izquierda (derecha) para compensar dichas observaciones.

5.0.2. ***Ejercicio:*** Explica por qué cuando la aproximación normal es apropiada, el intervalo de percentiles al 95 % es muy similar al intervalo normal de 2 errores estándar.

5.1. Ejemplo

Consideramos los datos de propinas. Queremos estimar la media de cuentas totales para la comida y la cena. Podemos hacer bootstrap de cada grupo por separado:

```
## en este ejemplo usamos rsample, pero puedes escribir tu propio ócdigo
library(rsample)
propinas <- read_csv("data/propinas.csv",
                     progress = FALSE,
                     show_col_types = FALSE) |>
  mutate(id = 1:244)
propinas
```

```
# A tibble: 244 × 7
   cuenta_total propina fumador dia   momento num_personas    id
          <dbl>   <dbl> <chr>   <chr> <chr>          <dbl> <int>
 1         17.0    1.01 No      Dom   Cena               2     1
 2         10.3    1.66 No      Dom   Cena               3     2
 3         21.0    3.5  No      Dom   Cena               3     3
 4         23.7    3.31 No      Dom   Cena               2     4
 5         24.6    3.61 No      Dom   Cena               4     5
 6         25.3    4.71 No      Dom   Cena               4     6
 7          8.77   2    No      Dom   Cena               2     7
 8         26.9    3.12 No      Dom   Cena               4     8
 9         15.0    1.96 No      Dom   Cena               2     9
10         14.8    3.23 No      Dom   Cena               2    10
# ... with 234 more rows
#  Use `print(n = ...)` to see more rows
```

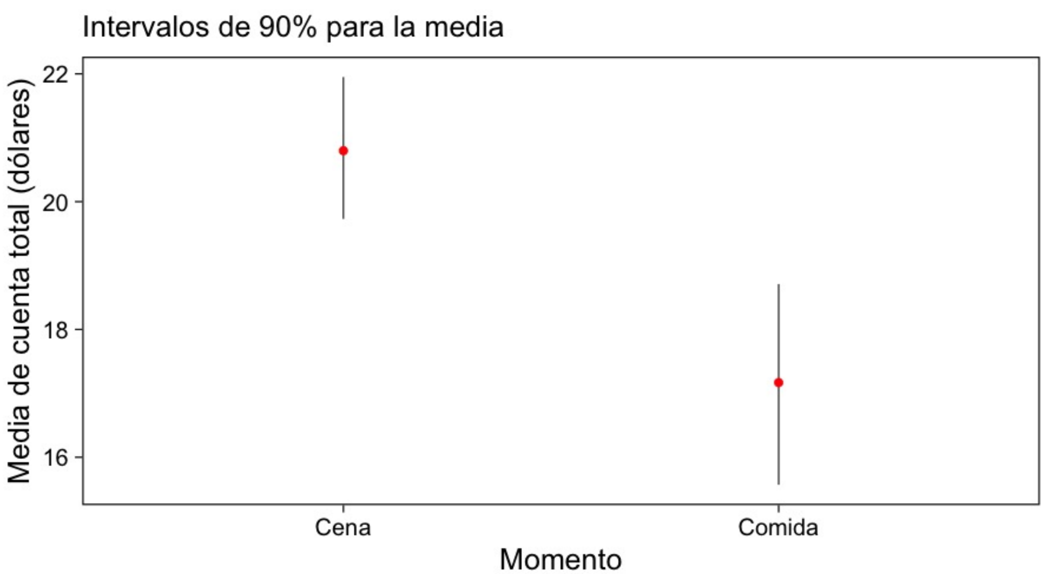
```
## paso 1: define el estimador
estimador <- function(split, ...){
  muestra <- analysis(split) |> group_by(momento)
  muestra |>
    summarise(estimate = mean(cuenta_total), .groups = 'drop') |>
    mutate(term = momento)
}
```

```
## paso 2 y 3: remuestrea y calcula estimador
boot_samples ← bootstraps(propinas, strata = momento, 1000) ▷
  mutate(res_boot = map(splits, estimador))
## paso 4: construye intervalos de confianza
intervalo_propinas_90 ← boot_samples ▷
  int_pctl(res_boot, alpha = 0.10) ▷
  mutate(across(where(is.numeric), round, 2))
intervalo_propinas_90
```

```
# A tibble: 2 × 6
  term    .lower .estimate .upper .alpha .method
  <chr>    <dbl>     <dbl>  <dbl>  <dbl> <chr>
1 Cena      19.7      20.8   22.0    0.1 percentile
2 Comida    15.6      17.1   18.7    0.1 percentile
```

Nota: `.estimate` es la media de los valores de la estadística sobre las remuestras, **no** es el estimador original.

De la tabla anterior inferimos que la media en la cuenta en la cena es más grande que la de la comida. Podemos graficar agregando los estimadores *plug-in*:



Nótese que el *bootstrap* lo hicimos por separado en cada momento del día (por eso el argumento `strata` en la llamada a **bootstraps**):

6. FUNCIONES DE CÓMPUTO:

Es común crear nuestras propias funciones cuando usamos *bootstrap*, sin embargo, en R también hay alternativas que pueden resultar convenientes:

1. El paquete `rsample` (forma parte de la colección tidymodels y tiene una función para realizar el remuestreo: `bootsrtraps()` que regresa un arreglo cuadrangular (`tibble`, `data.frame`) que incluye una columna con las muestras bootstrap y un identificador del número y tipo de muestra.

```
boot_samples
```

```
# Bootstrap sampling using stratification
# A tibble: 1,000 × 3
   splits            id            res_boot
   <list>            <chr>         <list>
 1 <split [244/100]> Bootstrap0001 <tibble [2 × 3]>
 2 <split [244/85]>  Bootstrap0002 <tibble [2 × 3]>
 3 <split [244/94]>  Bootstrap0003 <tibble [2 × 3]>
 4 <split [244/88]>  Bootstrap0004 <tibble [2 × 3]>
 5 <split [244/94]>  Bootstrap0005 <tibble [2 × 3]>
 6 <split [244/92]>  Bootstrap0006 <tibble [2 × 3]>
 7 <split [244/88]>  Bootstrap0007 <tibble [2 × 3]>
 8 <split [244/84]>  Bootstrap0008 <tibble [2 × 3]>
 9 <split [244/94]>  Bootstrap0009 <tibble [2 × 3]>
10 <split [244/90]>  Bootstrap0010 <tibble [2 × 3]>
# ... with 990 more rows
#  Use `print(n = ...)` to see more rows
```

Los objetos `splits` tienen muestras de tamaño 244. Sin embargo, utilizan (por el muestreo aleatorio con reemplazo) una fracción de los datos.

```
boot_samples$splits[[1]]
```

```
<Analysis/Assess/Total>
<244/100/244>
```

```
analysis(boot_samples$splits[[1]]) ▷
  group_by(id)
```

```
# A tibble: 244 × 7
# Groups:   id [144]
   cuenta_total propina fumador dia   momento num_personas    id
          <dbl>   <dbl> <chr>   <chr> <chr>          <dbl> <int>
 1         17.0    1.01 No      Dom   Cena               2     1
 2         17.0    1.01 No      Dom   Cena               2     1
 3         21.0    3.5  No      Dom   Cena               3     3
 4         23.7    3.31 No      Dom   Cena               2     4
 5         23.7    3.31 No      Dom   Cena               2     4
 6         23.7    3.31 No      Dom   Cena               2     4
 7         23.7    3.31 No      Dom   Cena               2     4
 8         25.3    4.71 No      Dom   Cena               4     6
 9         25.3    4.71 No      Dom   Cena               4     6
10          8.77   2    No      Dom   Cena               2     7
# ... with 234 more rows
#  Use `print(n = ...)` to see more rows
```

El paquete de `rsample` es un paquete muy eficiente para la creación de los conjunto de remuestreo y es una de sus principales ventajas.

```
library(pryr)
c(objeto_boot = object_size(boot_samples),
  original    = object_size(propinas),
  remuestra   = object_size(boot_samples)/nrow(boot_samples),
  incremento  = object_size(boot_samples)/object_size(propinas))
```

```
objeto_boot:   2.39 MB
original    : 15.43 kB
remuestra   :  2.39 kB
incremento  : 155.13 B
```

1. El paquete `boot` está asociado al libro *Bootstrap Methods and Their Applications* [1] y tiene, entre otras, funciones para calcular replicaciones *bootstrap* y para construir intervalos de confianza usando *bootstrap*:
   a) calculo de replicaciones *bootstrap* con la función `boot()`,
   b) intervalos normales, de percentiles y BC_a con la función `boot.ci()`,
   c) intervalos ABC con la función `abc.ci()`.
1. El paquete `bootstrap` contiene datos usados en [2], y la implementación de funciones para calcular replicaciones y construir intervalos de confianza:
   a) calculo de replicaciones *bootstrap* con la función `bootstrap()`,
   b) intervalos BC_a con la función `bcanon()`,
   c) intervalos ABC con la función `abcnon()`.

6.0.1. ***Ejercicio:*** Justifica el procedimiento de hacer el *bootstrap* separado para cada grupo. ¿Qué supuestos acerca del muestreo se deben satisfacer? ¿Deben ser muestras aleatorias simples de cada momento del día, por ejemplo? ¿Qué harías si no fuera así, por ejemplo, si se escogieron al azar tickets de todos los disponibles en un periodo?

7. CORRECCIÓN DE INTERVALOS

- Los intervalos basados en percentiles pueden ser mejorados con ciertos métodos de ajuste. El más popular, es el método acelerado con corrección de sesgo BC_a (*bias-corrected accelerated*).
- Los intervalos BC_a tienen mejores propiedades teóricas y mejor desempeño en la práctica.
- Para un intervalo de confianza los cuantiles $\alpha/2$ y $1-\alpha/2$ se ajustan por sesgo (*bias*) y por asimetría (*skewness*).
- Denotaremos por z_0 la corrección por sesgo y por a el ajuste por asimetría.
- La aceleración se obtiene de estimar la tasa de cambio del error estándar de $\hat{\theta}$ con respecto a θ en una escala normalizada.

7.0.1. ***Definición*** *[intervalos* boostrap *corregidos]:* El **intervalo de confianza** BC_a se construye como

$$[\hat{\theta}_{\mathsf{inf}}, \hat{\theta}_{\mathsf{sup}}] = [\hat{\theta}^{*(\alpha_1)}, \hat{\theta}^{*(\alpha_2)}]\,, \tag{6}$$

donde

$$\alpha_1 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z^{(\alpha)}}{1-\hat{a}\left(\hat{z}_0 + z^{(\alpha)}\right)}\right), \tag{7}$$

$$\alpha_2 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z^{(1-\alpha)}}{1-\hat{a}\left(\hat{z}_0 + z^{(1-\alpha)}\right)}\right), \tag{8}$$

donde $\Phi(\cdot)$ denota la función de acumulación de una normal estándar y $z^{(\alpha)}$ es el percentil α de una distribución normal estándar.

7.0.2. ***Ejercicio:*** Si no hay sesgo ni modificación por asimetría entonces tenemos los intervalos percentiles basados en un aproximación Gaussiana.

7.1. Cómputo del ajuste

El ajuste por sesgo se calcula por medio de la réplicas *bootstrap* y el estimador observado de nuestra muestra original

$$\hat{z}_0 = \Phi^{-1}\left(\frac{|\{\hat{\theta}^{(b)} < \hat{\theta}\}|}{B}\right) . \tag{9}$$

Obtenemos $\hat{z}_0 = 0$ si la mitad de las muestras *bootstrap* son menores a $\hat{\theta}$.

La aceleración $\hat{a}$ se calcula a través del método *jackknife* por medio de

$$\hat{a} = \frac{\sum_{i=1}^{n}\left(\widehat{\theta}_{(\cdot)} - \widehat{\theta}_{(i)}\right)^3}{6\left\{\sum_{i=1}^{n}\left(\widehat{\theta}_{(\cdot)} - \widehat{\theta}_{(i)}\right)^2\right\}^{3/2}} . \tag{10}$$

7.2. Ejemplo: Valor de un lote de casas

Recordemos nuestro problema de estimación para el precio total de casas en un lote. Para poder construir los intervalos necesitamos agregas la muestra original.

```
totales_boot ← bootstraps(muestra_casas,  2000, apparent = TRUE) ▷
  mutate(res_boot = map(splits, estimador_lote))
totales_boot ▷ tail()
```

```
# A tibble: 6 × 3
  splits            id            res_boot
  <list>            <chr>         <list>
1 <split [200/81]>  Bootstrap1996 <tibble [1 × 2]>
2 <split [200/74]>  Bootstrap1997 <tibble [1 × 2]>
3 <split [200/79]>  Bootstrap1998 <tibble [1 × 2]>
4 <split [200/67]>  Bootstrap1999 <tibble [1 × 2]>
5 <split [200/73]>  Bootstrap2000 <tibble [1 × 2]>
6 <split [200/200]> Apparent      <tibble [1 × 2]>
```

Los intervalos por el método de percentiles son:

```
totales_boot ▷
  int_pctl(res_boot) ▷
  select(- .alpha ) ▷
  mutate_if(is.numeric, function(x) {x/1000}) ▷
  mutate(length = .upper - .lower)
```

```
# A tibble: 1 × 6
  term       .lower .estimate .upper .method    length
  <chr>       <dbl>     <dbl>  <dbl> <chr>       <dbl>
1 Valor lote   195.      208.   222. percentile   26.8
```

Los intervalos corregidos por sesgo y asimetría son:

```
intervalos_bca <- totales_boot |>
  int_bca(res_boot, .fn = estimador_lote)
```

```
# A tibble: 1 × 6
  term        .lower .estimate .upper .method length
  <chr>        <dbl>     <dbl>  <dbl> <chr>    <dbl>
1 Valor lote    196.      208.   223. BCa       26.9
```

7.3. Ejemplo: area habitable

Recordemos nuestro ejemplo de calcular el porcentaje del area habitable en las viviendas. Un problema con problemas mas severos de asimetría.

```
estimador_razon <- function(split, ...){
  muestra <- analysis(split)
  muestra |>
    summarise(estimate = sum(area_habitable_sup_m2) / sum(area_lote_m2),
              .groups = "drop") |>
    mutate(term = "area del lote construida")
}
```

```
razon_boot <- bootstraps(muestra_casas, 2000, apparent = TRUE) |>
  mutate(res_boot = map(splits, estimador_razon))
```

```
razon_boot |>
  int_pctl(res_boot) |>
  select(- .alpha ) |>
  mutate_if(is.numeric, function(x) {x*100}) |>
  mutate(length = .upper - .lower)
```

```
# A tibble: 1 × 6
  term                     .lower .estimate .upper .method    length
  <chr>                     <dbl>     <dbl>  <dbl> <chr>       <dbl>
1 area del lote construida   12.1      14.1   15.8 percentile   3.76
```

```
intervalos_bca <- razon_boot |>
  int_bca(res_boot, .fn = estimador_razon)
```

```
# A tibble: 1 × 6
  term                     .lower .estimate .upper .method length
  <chr>                     <dbl>     <dbl>  <dbl> <chr>    <dbl>
1 area del lote construida   11.0      14.1   15.5 BCa       4.44
```

Podemos comparar con los intervalos obtenidos de la distribución de muestreo del estimador.

```
resample_data <- poblacion_casas |>
  mc_cv(prop = 200/1144, 2000) |>
  mutate(results = map(splits, estimador_razon))
resample_data
```

```
# Monte Carlo cross-validation (0.17/0.83) with 2000 resamples
# A tibble: 2,000 × 3
   splits            id           results
   <list>            <chr>        <list>
 1 <split [200/944]> Resample0001 <tibble [1 × 2]>
 2 <split [200/944]> Resample0002 <tibble [1 × 2]>
 3 <split [200/944]> Resample0003 <tibble [1 × 2]>
 4 <split [200/944]> Resample0004 <tibble [1 × 2]>
 5 <split [200/944]> Resample0005 <tibble [1 × 2]>
 6 <split [200/944]> Resample0006 <tibble [1 × 2]>
 7 <split [200/944]> Resample0007 <tibble [1 × 2]>
 8 <split [200/944]> Resample0008 <tibble [1 × 2]>
 9 <split [200/944]> Resample0009 <tibble [1 × 2]>
10 <split [200/944]> Resample0010 <tibble [1 × 2]>
# ... with 1,990 more rows
#  Use `print(n = ...)` to see more rows
```

```
resample_data ▷
  unnest(results) ▷
  summarise(inf = quantile(estimate, probs = c(0.025)) * 100,
            sup = quantile(estimate, probs = c(0.975)) * 100) ▷
  mutate(length = sup - inf)
```

```
# A tibble: 1 × 3
    inf   sup length
  <dbl> <dbl>  <dbl>
1  12.9  15.6   2.73
```

8. CONCLUSIONES Y OBSERVACIONES

- El principio fundamental del *bootstrap* es que podemos estimar la distribución poblacional con la distribución empírica. Por tanto para hacer inferencia tomamos muestras con reemplazo de la distribución empírica y analizamos la variación de la estadística de interés a lo largo de las muestras.
- El bootstrap nos da la posibilidad de crear intervalos de confianza cuando no contamos con fórmulas para hacerlo de manera analítica y sin supuestos distribucionales de la población.
- Hay muchas opciones para construir intervalos bootstrap, los que tienen mejores propiedades son los intervalos BC_a, sin embargo los más comunes son los intervalos normales con error estándar *bootstrap* y los intervalos de percentiles de la distribución *bootstrap*.
- Antes de hacer intervalos normales (o con percentiles de una t) vale la pena graficar la distribución *bootstrap* y evaluar si el supuesto de normalidad es razonable.
- En cuanto al número de muestras bootstrap se recomienda al menos $1,000$ al hacer pruebas, y $10,000$ o $15,000$ para los resultados finales, sobre todo cuando se hacen intervalos de confianza de percentiles.
- La función de distribución empírica es una mala estimación en las colas de las distribuciones, por lo que es difícil construir intervalos de confianza (usando bootstrap no paramétrico) para estadísticas que dependen mucho de las colas.

REFERENCIAS

[1] A. Davison and D. Hinkley. *Bootstrap Methods and Their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997. ISBN 978-1-107-26853-1. 12

[2] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Springer US, Boston, MA, 1993. ISBN 978-0-412-04231-7 978-1-4899-4541-9. . 1, 12