EST-24107: Simulación

Profesor: Alfredo Garbuno Iñigo — Otoño, 2022 — *Bootstrap*.
Objetivo: Que veremos.
Lectura recomendada: Referencia.

1. INTRODUCCIÓN

El remuestreo se refiere a un conjunto de técnicas estadísticas, computacionalmente intensivas, que `estiman` la *distribución de una población* basadas en `muestreo aleatorio con reemplazo` de una muestra observada.

Se considera una muestra aleatoria $X_1, \ldots, X_n$ como si fuera una población finita y se generan muestras aleatorias de la misma muestra para estimar características poblacionales y hacer inferencia de la población muestreada.

Las técnicas de remuestreo permiten calcular medidas de ajuste (en términos de sesgo, varianza, intervalos de confianza, errores de predicción o de algunas otras medidas) a los estimados basados en muestras.

Estas técnicas son usualmente no paramétricas, y varias son tan antiguas como la estadística misma. Por ejemplo, las técnicas de permutación son de Fisher (1935) y Pitmann (1937); la validación cruzadas fue propuesta por Kurtz en 1948, y el *Jackknife* fue propuesto por Maurice Quenouille en 1949 aunque fue John Tukey en 1958 quién le dio el nombre a la técnica.

1.1. Contexto histórico

Bradley Efron introdujo el Bootstrap en 1979, y sus estudiantes Rob Tibshirani y Trevor Hastie han aportado mucho a la ciencia estadística. Ofrecen un curso en Statistical Learning en la plataforma MOOC de la Universidad de Stanford.

El término 'bootstrapping' se refiere al concepto de "pulling oneself up by one's bootstraps", frase que aparentemente se usó por primera vez en:

- Raspe, R. E. (1786). *Gulliver Revived: Or the Singular Travels, Campaigns, Voyages, and Adventures of Baron Munikhouson, Commonly Called Munchausen.*

1.2. Idea general

El objetivo del remuestreo es estimar alguna característica poblacional, representada por (tal como media, mediana, desviación estándar, coeficientes de regresión, matriz de covarianza, etc.) basado en los datos.

También interesan las propiedades de la distribución de estimador, sin hacer supuestos restrictivos sobre la forma de la distribución de los datos originales.

Para una muestra aleatoria $X_1, \ldots, X_n$, la distribución de remuestreo es la distribución empírica $\hat{\mathbb{P}}_n$, que asigna probabilidad $1/n$ a cada una de las observaciones de la muestra.

1.3. Ejemplo

Consideremos una muestra de 6 parejas. La variable de interés es la diferencia del ingreso de los miembros de cada pareja (en miles de pesos al mes).

i	$P_i^{(1)}$	$P_i^{(2)}$	$d_i = P_i^{(1)} - P_i^{(2)}$
1	24	18	6
2	14	17	-3
3	40	35	5
4	44	41	3
5	35	37	-2
6	45	45	0

Definamos θ como el promedio de las diferencias de ingreso poblacional. Podemos estimar θ con

$$\hat{\theta}_n = \frac{6 - 3 + 5 + 3 - 2 + 0}{6} = 1.5\,. \tag{1}$$

¿Cómo calculamos la variabilidad de nuestro estimador? Es decir, ¿cómo calculamos la variabilidad de $\hat{\theta}_n$?

1.3.1. ***Ejercicio:*** Escribe la fórmula del error estándar bajo los siguientes supuestos:

1. La diferencia tiene una distribución $d_i \sim \mathsf{N}(\theta, \sigma^2)$.
2. La varianza σ^2 es conocida.

1.4. `Observaciones`:

- Suponer que la diferencia de ingresos es d_i como una variable normal puede no estar *tan* errado. Pues con un número suficiente de muestras podríamos suponer que el resultado del TLC se cumple. Entonces, ¿qué hacemos si no conocemos la distribución de las observaciones?
- Si no conocemos σ^2 lo podemos estimar con la muestra. Por ejemplo, podemos utilizar intervalos de confianza derivados de una distribución t.
- Si nos interesa otro parámetro de la población podemos construir estimadores diferentes. Por ejemplo, nos podría interesar la `mediana` de una población $q_{0.5} = \mathbb{P}^{-1}(1/2)$. Para este caso, podemos estimar dicho parámetro por medio de

$$\hat{q}_{0.5} = \begin{cases} X_{(\frac{n+1}{2})} & \text{si } n \text{ es impar} \\ \frac{X_{(n/2)} + X_{(n/2+1)}}{2} & \text{si } n \text{ es par} \end{cases}\,. \tag{2}$$

En Fig. 1 la estimación de la mediana en distintos grupos acompañados de su estimación de incertidumbre.

1.5. La distribución de muestreo

Hasta ahora lo que hemos hecho es estimar $\hat{\pi}_N^{\mathsf{MC}}(f) \approx \pi(f) = \int f(x)\,\pi(x)\,\mathrm{d}x$ por medio de muestras de la densidad $\pi(\cdot)$. Es decir, por medio de

$$X_1, \ldots, X_N \overset{\mathsf{iid}}{\sim} \pi\,. \tag{3}$$

Hemos considerado la noción frecuentista de medir nuestra incertidumbre en nuestro estimador por medio del **error estándar** de nuestro estimador. Donde éste último está definido como

$$\mathsf{ee}\left(\hat{\pi}_N^{\mathsf{MC}}(f)\right) = \left(\mathbb{V}(\hat{\pi}_N^{\mathsf{MC}}(f))\right)^{1/2}\,, \tag{4}$$

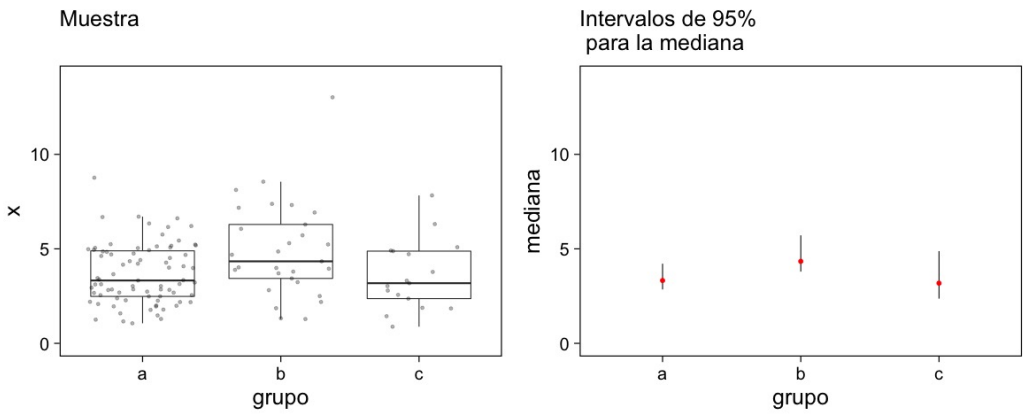


FIGURA 1. *Estimación de mediana (panel izquierdo) con intervalos de incertidumbre (panel derecho).*

y la varianza es con respecto a la variabilidad que *nace* por haber observado distintas muestras.

Es decir, estamos considerando la situación en que podemos replicar el proceso de muestreo tantas veces como queramos (o recursos computacionales tengamos). Denotemos por B el número de réplicas que podemos realizar y denotemos por

$$X_1^{(b)}, \ldots, X_N^{(b)} \overset{\text{iid}}{\sim} \pi\,, \qquad b = 1, \ldots, B\,, \tag{5}$$

la réplica que generamos.

Notemos que es a través de este proceso de crear réplicas podemos construir una distribución para $\hat{\pi}_N^{\mathsf{MC}}(f)$ y notemos, además, que nuestro estimador es el resultado de aplicar una función a la muestra dada

$$\hat{\pi}_N^{\mathsf{MC},(b)}(f) = t(X_1^{(b)}, \ldots, X_N^{(b)})\,, \qquad b = 1, \ldots, B\,. \tag{6}$$

La distribución de resultante de nuestro estimador $\hat{\pi}_N^{\mathsf{MC}}(f)$ derivada de haber observado un conjunto de datos distinto es lo que en sus cursos de estadística le llamamos `distribución de muestreo` del estimador.

Nota que en esta situación asumimos que podemos generar tantas muestras como queramos de la distribución de interés π. En esta sección del curso estudiaremos un mecanismo para cuando no podemos hacer eso (generar muestras de una población) y sólo tenemos acceso a una muestra—que asumimos aleatoria—de la población que nos interesa.

2. LA IDEA DEL BOOTSTRAP

Como explicamos, el problema que tenemos ahora es que normalmente sólo tenemos una muestra, así que no es posible calcular las distribuciones de muestreo como hicimos arriba y evaluar qué tan preciso es nuestro estimador. Sin embargo, podemos hacer lo siguiente:

Supongamos que tenemos una muestra $X_1, X_2, \ldots, X_n$ independientes de alguna población desconocida y un estimador $T_n = t(X_1, \ldots, X_n)$

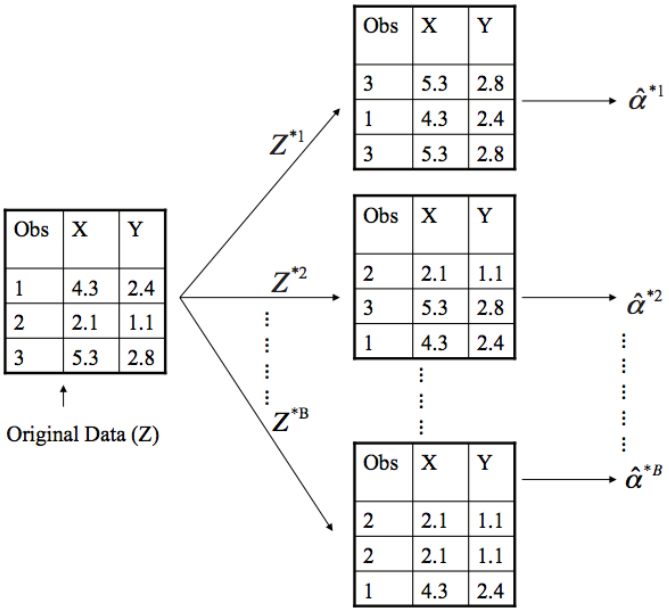
Mundo poblacional

1. Si tuviéramos la distribución poblacional, simulamos muestras iid para aproximar la distribución de muestreo de nuestro estimador, y así entender su variabilidad.
2. Pero **no** tenemos la distribución poblacional.
3. **Sin embargo, podemos estimar la distribución poblacional con nuestros valores muestrales**.

Mundo *bootstrap*

1. Si usamos la estimación del inciso 3, entonces usando el inciso 1 podríamos tomar muestras de nuestros datos muestrales, como si fueran de la población, y usando el mismo tamaño de muestra. El muestreo lo hacemos con reemplazo de manera que produzcamos muestras independientes de la misma "población estimada", que es la muestra.
2. Evaluamos nuestra estadística en cada una de estas remuestras.
3. A la distribución resultante le llamamos **distribución *bootstrap*** o **distribución de remuestreo** del estimador.
4. Usamos la distribución *bootstrap* de la muestra para estimar la variabilidad en nuestra estimación con **la muestra original**.

El esquema de esta estrategia lo podemos representar con la figura siguiente



Veamos que sucede para un ejemplo concreto, donde nos interesa estimar la media de los precios de venta de una población de casas. Tenemos nuestra muestra:

```
set.seed(2112)
poblacion_casas ← read_csv("data/casas.csv")
muestra ← sample_n(poblacion_casas, 200, replace = TRUE) ▷
  select(id, nombre_zona, area_habitable_sup_m2, precio_miles)
```

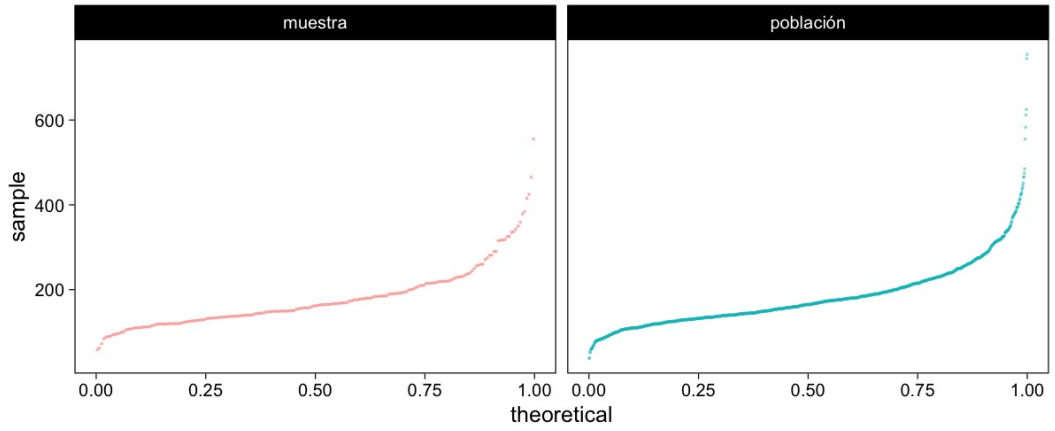
```
# A tibble: 6 × 4
     id nombre_zona area_habitable_sup_m2 precio_miles
  <dbl> <chr>                       <dbl>        <dbl>
1   502 Somerst                     164.          227.
2    79 Sawyer                      164.          136.
3   440 Edwards                     111.          110
4   524 Edwards                     434.          185.
5  1442 CollgCr                      78.8         149.
6   769 CollgCr                     171.          217.
```

```
[1] "Hay 1144 casas en total, tomamos muestra de 200"
```

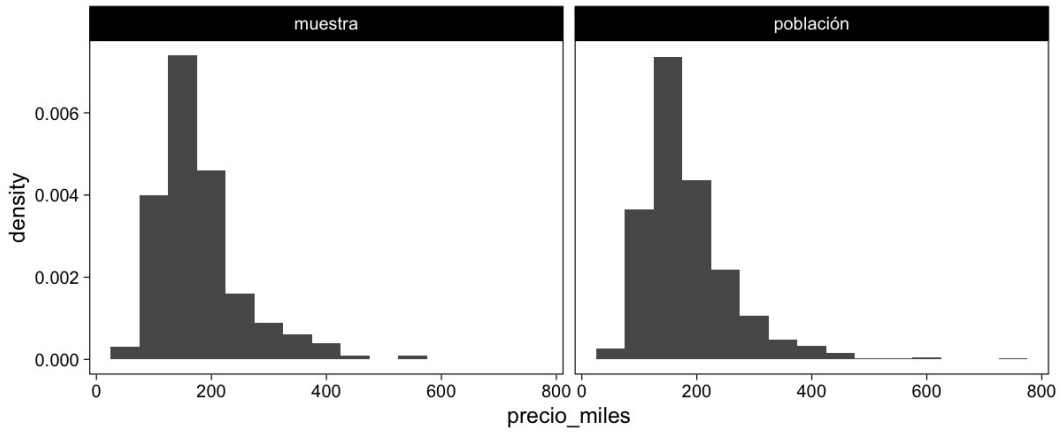
```
mean(muestra$precio_miles)
```

```
[1] 179.96
```

Esta muestra nos da nuestro estimador de la distribución poblacional. Por ejemplo, podemos fijarnos en un gráfico de cuantiles:



o en histogramas:



Y vemos que la aproximación es razonable en las partes centrales de la distribución.

Ahora supongamos que nos interesa cuantificar la precisión de nuestra estimación de la media poblacional de precios de casas, y usaremos la media muestral para hacer esto. Para nuestra muestra, nuestra estimación puntual es:

```
media ← mean(muestra$precio_miles)
media
```

```
[1] 179.96
```

Y recordamos que para aproximar la distribución de muestreo podíamos muestrear repetidamente la población y calcular el valor del estimador en cada una de estas muestras. Aquí no tenemos la población, **pero tenemos una estimación de la población**: la muestra obtenida.

Así que para evaluar la variabilidad de nuestro estimador, entramos en el mundo boostrap, y consideramos que la población es nuestra muestra.

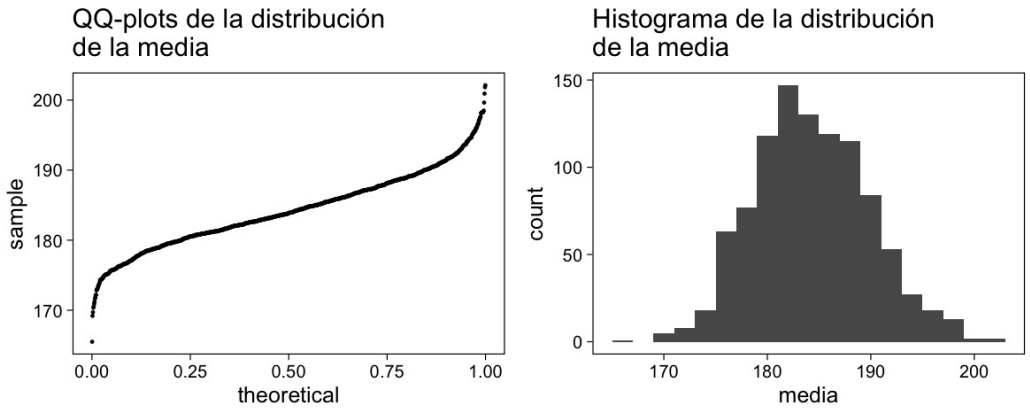
Podemos entonces extraer un número grande de muestras con reemplazo de tamaño 200 **de la muestra**: el muestreo debe ser análogo al que se tomó para nuestra muestra original. Evaluamos nuestra estadística (en este caso la media) en cada una de estas remuestras:

```
genera_remuestras ← function(data, n = 200){
  data ▷
    sample_n(200, replace = TRUE)
}
```

```
calcula_estimador ← function(data){
  data ▷
    summarise(media_precio = mean(precio_miles), .groups = "drop")
}
```

```
media_muestras ← map_dbl(1:1000, function(id){
  genera_remuestras(muestras) ▷
    calcula_estimador() ▷
    pull(media_precio)})
media_muestras[1:10]
```

Y nuestra estimación de la distribución de muestreo para la media es entonces:



A esta le llamamos la distribución de remuestreo de la media, que definimos más abajo. Ahora podemos calcular un intervalo de confianza del 90 % simplemente calculando los cuantiles de esta distribución (no son los cuantiles de la muestra original!):

```
limites_ic ← quantile(media_muestras, c(0.05,  0.95)) ▷ round(4)
limites_ic
```

```
    5%    95%
175.71 193.70
```

Otra cosa que podríamos hacer para describir la dispersión de nuestro estimador es calcular el error estándar de remuestreo, que estima el error estándar de la distribución de muestreo:

```
ee_boot ← sd(media_muestras)
round(ee_boot, 2)
```

```
[1] 5.55
```

2.0.1. ***Definición:*** Sea $X_1, X_2, \ldots, X_n$ una muestra independiente y idénticamente distribuida (iid), y $T_n = t(X_1, X_2, \ldots, X_n)$ una estadística. Supongamos que los valores que obervamos son $x_1, x_2, \ldots, x_n$.

La **distribución de remuestreo** de T_n es la distribución de $T^* = t(X_1^*, X_2^*, \ldots X_n^*)$, donde cada X_i^* se obtiene tomando al azar uno de los valores de $x_1, x_2, \ldots, x_n$.

Otra manera de decir esto es que la remuestra $X_1^*, X_2^*, \ldots, X_n^*$ es una muestra con reemplazo de los valores observados $x_1, x_2, \ldots, x_n$

2.0.2. ***Ejemplo:*** Si observamos la muestra

```
muestra ← sample(1:20, 5)
muestra
```

```
[1] 10 13 17  3  8
```

Una remuestra se obtiene:

```
sample(muestra, size = 5, replace = TRUE)
```

```
[1] 13  3 13 17 10
```

Nótese que algunos valores de la muestra original pueden aparecer varias veces, y otros no aparecen del todo.

2.1. Nota

La muestra original es una aproximación de la población de donde fue extraída. Así que remuestrear la muestra aproxima lo que pasaría si tomáramos muestras de la población. La **distribución de remuestreo** de una estadística, que se construye tomando muchas remuestras, aproxima la distribución de muestreo de la estadística.

Y el proceso que hacemos es:

2.1.1. ***Remuestreo para una población:*** Dada una muestra de tamaño n de una población,

1. Obtenemos una remuestra de tamaño n con reemplazo de la muestra original
2. Repetimos este remuestreo muchas veces (por ejemplo $10,000$).
3. Construímos la distribución *bootstrap*, y examinamos sus características (dónde está centrada, dispersión y forma).

3. EL PRINCIPIO DE PLUG-IN

La idea básica detrás del *bootstrap* es el principio de *plug-in* para estimar parámetros poblacionales: si queremos estimar una cantidad poblacional, calculamos esa cantidad poblacional con la muestra obtenida. Es un principio común en estadística.

Por ejemplo, si queremos estimar la media o desviación estándar poblacional, usamos la media muestral o la desviación estándar muestral. Si queremos estimar un cuantil de la población usamos el cuantil correspondiente de la muestra, y así sucesivamente.

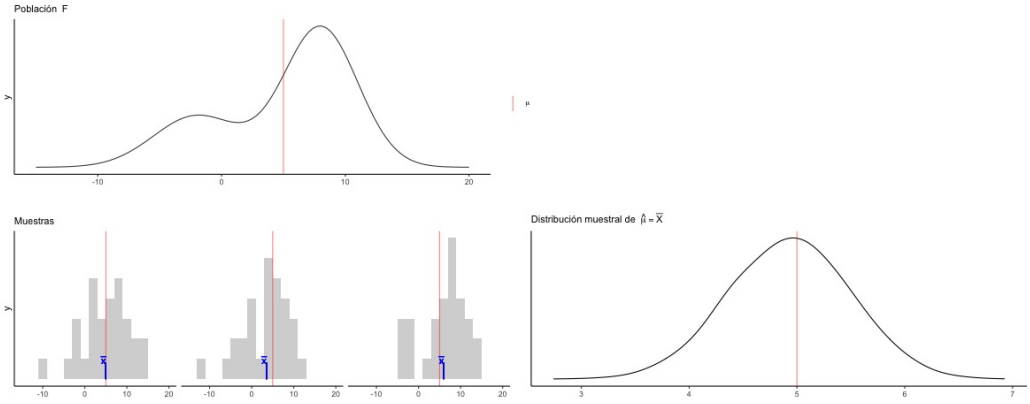
En todos estos casos, lo que estamos haciendo es:

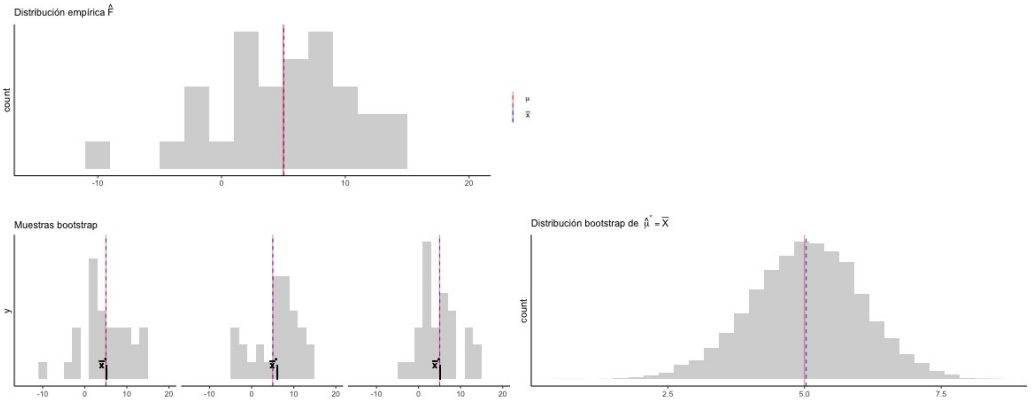
- Tenemos una fórmula para la cantidad poblacional de interés en términos de la distribución poblacional.
- Tenemos una muestra, que usamos para estimar la cantidad poblacional. La distribución que da una muestra se llama distribución **empírica**.
- Contruimos nuestro estimador "enchufando" la distribución empírica de la muestra en la fórmula del estimador.

En el *bootstrap* aplicamos este principio simple a la **distribución de muestreo**:

- **Si tenemos la población**, podemos **calcular** la distribución de muestreo de nuestro estimador tomando muchas muestras de la **población**.
- Estimamos la **poblacion** con la **muestra** y enchufamos en la frase anterior:
- Podemos **estimar** la distribucion de muestreo de nuestro estimador tomando muchas muestras de la **muestra** (*bootstrap*).

Nótese que el proceso de muestreo en el último paso **debe ser el mismo** que se usó para tomar la muestra original. Estas dos imágenes simuladas con base en un ejemplo de [1] muestran lo que acabamos de describir:





3.1. Observación

Veremos ejemplos más complejos, pero nótese que si la muestra original son observaciones independientes obtenidas de la distribución poblacional, entonces logramos esto en las remuestras tomando aleatoriamente observaciones con reemplazo de la muestra. Igualmente, las remuestras deben ser del mismo tamaño que la muestra original.

3.1.1. Ejercicio:

- ¿Porqué no funcionaría tomar muestras sin reemplazo? Piensa si hay independencia entre las observaciones de la remuestra, y cómo serían las remuestras sin reemplazo.
- ¿Por qué no se puede hacer bootstrap si no conocemos cómo se obtuvo la muestra original?

3.2. Observación

Estos argumentos se pueden escribir con fórmulas usando por ejemplo la función de distribución acumulada $\mathbb{P}$ de la población y su estimador, que es la función empírica $\hat{\mathbb{P}}_n$, como en [2]. Si $\theta = t(\mathbb{P})$ es una cantidad poblacional que queremos estimar, su estimador *plug-in* es $\hat{\theta} = t(\hat{\mathbb{P}}_n)$.

3.3. Observación

La distribución empírica $\hat{\mathbb{P}}_n$ es un estimador *razonable* de la distribución poblacional $\mathbb{P}$ pues por el teorema de Glivenko-Cantelli ([3], o aquí), $\hat{\mathbb{P}}_n$ converge a $\mathbb{P}$ cuando el tamaño de muestra $n \to \infty$, lo cual es intuitivamente claro.

3.4. Ejemplo

En el siguiente ejemplo (tomadores de té), podemos estimar la proporción de tomadores de té que prefiere el té negro usando nuestra muestra:

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

```
te ▷
  mutate(negro = ifelse(Tea == "black", 1, 0)) ▷
  summarise(prop_negro = mean(negro), n = length(negro), .groups = "drop")
```

```
# A tibble: 1 × 2
  prop_negro     n
       <dbl> <int>
1      0.247   300
```

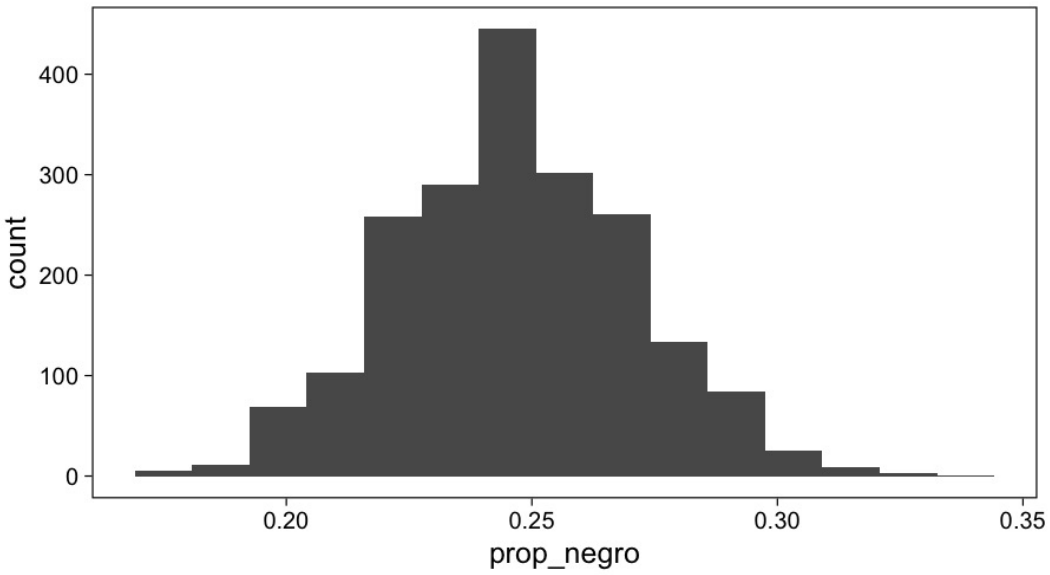
¿Cómo evaluamos la precisión de este estimador? Supondremos que el estudio se hizo tomando una muestra aleatoria simple de tamaño 300 de la población de tomadores de té que nos interesa. Podemos entonces usar el boostrap:

```
## 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), .groups = "drop") ▷
    pull(prop_negro)
  prop_negro
}
```

```
## paso 2: define el proceso de remuestreo
muestra_boot ← function(datos){
  ## tomar muestra con reemplazo del mismo ñtamao
  sample_n(datos, size = nrow(datos), replace = TRUE)
}
```

```
## paso 3: define el proceso de bootstrap
aplica_bootstrap ← function(id){
  muestra_boot(datos = te) ▷
    calc_estimador()
}
```

```
# paso 4: aplica el proceso de bootstrap
prop_negro_tbl ← map_dbl(1:2000, aplica_bootstrap ) ▷
  as_tibble() ▷
  rename( prop_negro = value)
```



Y podemos evaluar varios aspectos, por ejemplo dónde está centrada y qué tan dispersa es la distribución *bootstrap*:

```
prop_negro_tbl ▷
  summarise(media = mean(prop_negro),
            sesgo = mean(prop_negro) - 0.2499,
            ee = sd(prop_negro),
            cuantil_75 = quantile(prop_negro, 0.75),
            cuantil_25 = quantile(prop_negro, 0.25),
            .groups = "drop") ▷
  mutate(across(where(is.numeric), round, 3)) ▷
  pivot_longer(cols = everything())
```

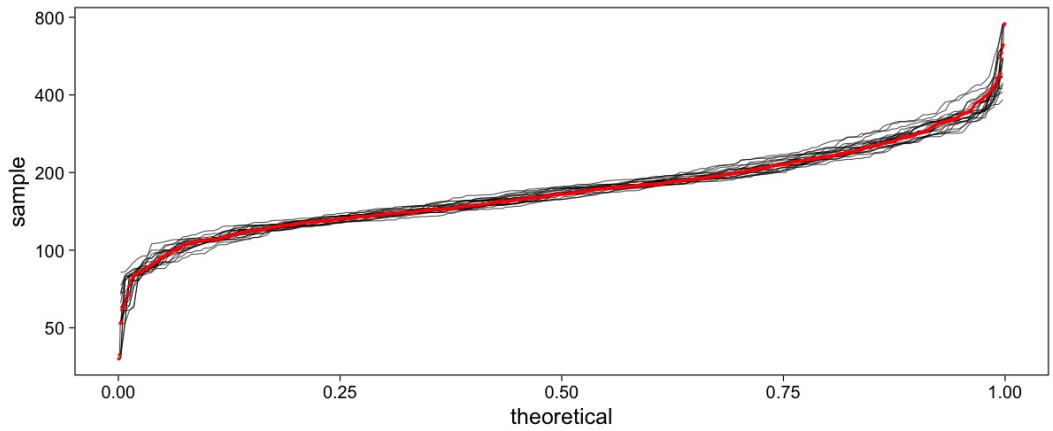
```
# A tibble: 5 × 2
  name         value
  <chr>        <dbl>
1 media        0.247
2 sesgo       -0.003
3 ee           0.024
4 cuantil_75   0.263
5 cuantil_25   0.23
```

4. PROPIEDADES DISTRIBUCIÓN BOOTSTRAP

Uasremos la distribución *bootstrap* principalmente para evaluar la variabilidad de nuestros estimadores (y también otros aspectos como sesgo) estimando la dispersión de la distribución de muestreo. Sin embargo, es importante notar que **no** la usamos, por ejemplo, para saber dónde está centrada la distribución de muestreo, o para "mejorar" la estimación remuestreando.

4.1. Ejemplo

En nuestro ejemplo, podemos ver varias muestras (por ejemplo 20) de tamaño 200, y vemos cómo se ve la aproximación a la distribución de la población:



Podemos calcular las distribuciones de remuestreo para cada muestra bootstrap, y compararlas con la distribución de muestreo real.

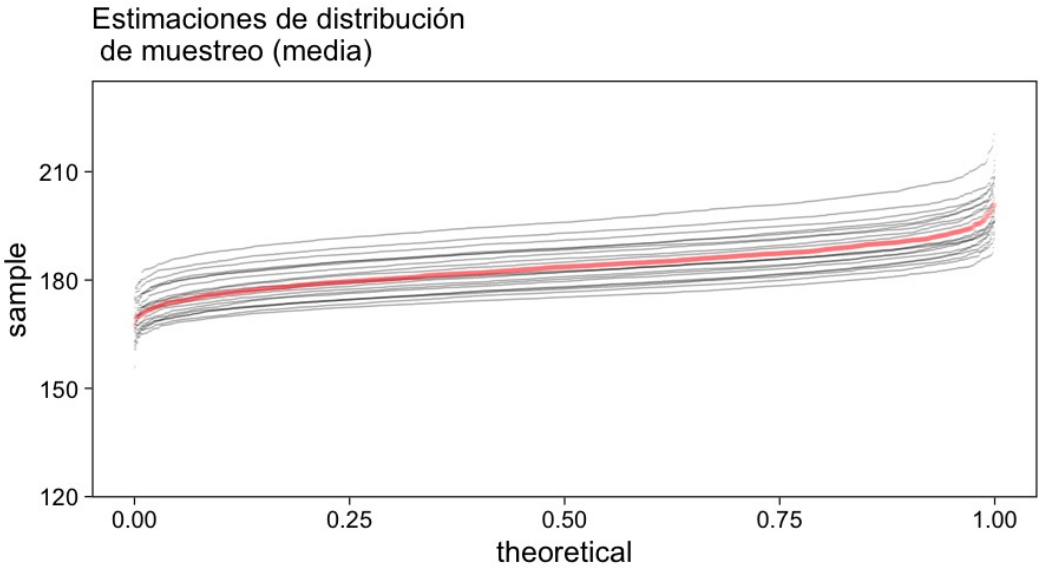
```
## paso 1: define el estimador
calc_estimador ← function(datos){
  media_precio ← datos ▷
    summarise(media = mean(precio_miles), .groups = "drop") ▷
    pull(media)
  media_precio
}
```

```
## paso 2: define el proceso de remuestreo
muestra_boot ← function(datos, n = NULL){
  ## tomar muestra con reemplazo del mismo ñtamao
  if(is.null(n)){
    m ← sample_n(datos, size = nrow(datos), replace = TRUE)}
  else {
      m ← sample_n(datos, size = n, replace = TRUE)
    }
  m
}
```

```
## paso 3: define el proceso de bootstrap
aplica_bootstrap ← function(data, n = NULL){
  data ▷
    muestra_boot(n) ▷
    calc_estimador()
}
```

```
## paso 3: realiza el remuestreo y calcula estimadores
dist_boot ← datos_sim ▷
  filter(tipo == "muestras") ▷
  select(precio_miles, rep) ▷
  group_by(rep) ▷ nest() ▷
  mutate(precio_miles =  map(data, function(data){
    tibble(precio_miles = rerun(1000, aplica_bootstrap(data)))
  })) ▷
  select(rep, precio_miles) ▷
  unnest(precio_miles) ▷
  mutate(precio_miles = unlist(precio_miles))
```

```
## extra: comparamos contra distribucion de muestreo
dist_muestreo ← datos_sim ▷
  filter(tipo == "ópoblacin") ▷
  group_by(rep) ▷ nest() ▷
  mutate(precio_miles =  map(data, function(data){
    tibble(precio_miles = rerun(1000, aplica_bootstrap(data, n = 200)))
  })) ▷
  select(rep, precio_miles) ▷
  unnest(precio_miles) ▷
  mutate(precio_miles = unlist(precio_miles))
```



Obsérvese que:

- En algunos casos la aproximación es mejor que en otros (a veces la muestra tiene valores ligeramente más altos o más bajos).
- La dispersión de cada una de estas distribuciones *bootstrap* es similar a la de la verdadera distribución de muestreo (en rojo), pero puede está desplazada dependiendo de la muestra original que utilizamos.
- Adicionalmente, los valores centrales de la distribución de *bootstrap* tiende cubrir el verdadero valor que buscamos estimar, que es:

```
poblacion_casas ▷
  summarise(media = mean(precio_miles), .groups = "drop")
```

```
# A tibble: 1 × 1
  media
  <dbl>
1  183.
```

4.2. Variación en distribución bootstrap

En el proceso de estimación bootstrap hay dos fuentes de variación pues:

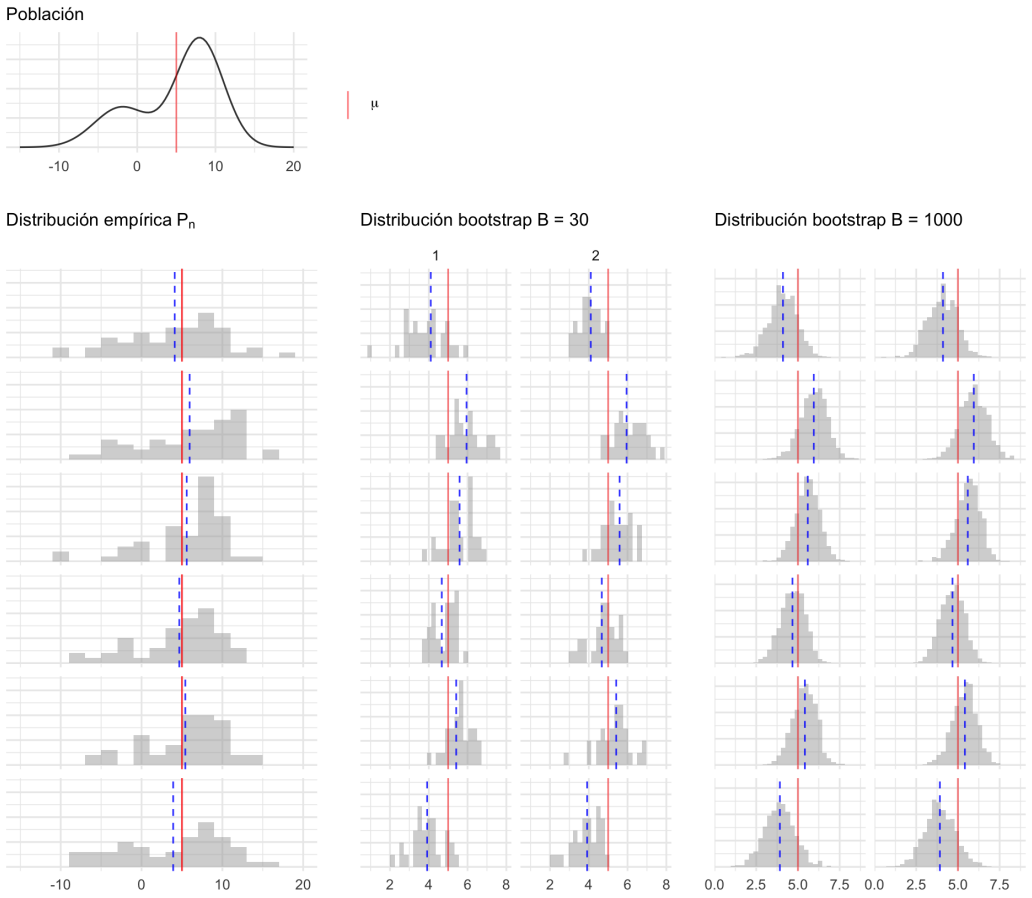
- La muestra original se selecciona con aleatoriedad de una población.

- Las muestras *bootstrap* se seleccionan con aleatoriedad de la muestra original. Esto es, la estimación bootstrap ideal es un resultado asintótico $B = \infty$, en esta caso $\hat{\mathsf{ee}}_B$ iguala la estimación *plug-in* $\mathsf{ee}_{\mathbb{P}_n}$.

En el proceso de **bootstrap** podemos controlar la variación del segundo aspecto, conocida como **implementación de muestreo Monte Carlo**, y la variación Monte Carlo decrece conforme incrementamos el número de muestras.

Podemos eliminar la variación Monte Carlo si seleccionamos todas las posibles muestras con reemplazo de tamaño n, hay $\binom{2n-1}{n}$ posibles muestras y si seleccionamos todas obtenemos $\hat{\mathsf{ee}}_\infty$ (bootstrap ideal), sin embargo, en la mayor parte de los problemas no es factible proceder así.

En la siguiente gráfica mostramos 6 posibles muestras de tamaño 50 simuladas de la población, para cada una de ellas se graficó la distribución empírica y se se realizan histogramas de la distribución bootstrap con $B = 30$ y $B = 1000$, en cada caso hacemos dos repeticiones, notemos que cuando el número de muestras bootstrap es grande las distribuciones bootstrap son muy similares (para una muestra de la población dada), esto es porque disminuimos el erro Monte Carlo. También vale la pena recalcar que la distribución bootstrap está centrada en el valor observado en la muestra (línea azúl punteada) y no en el valor poblacional sin embargo la forma de la distribución es similar a lo largo de las filas.



Entonces, ¿cuántas muestras bootstrap?

1. Incluso un número chico de replicaciones bootstrap, digamos $B = 25$ es informativo, y $B = 50$ con frecuencia es suficiente para dar una buena estimación de $\mathsf{ee}_P(\hat{\theta})$ ([2]).

2. Cuando se busca estimar error estándar ([1]) recomienda $B = 1000$ muestras, o $B = 10,000$ muestras dependiendo la precisión que se busque.

5. 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 $\hat{\theta} = t(X_1, \ldots, X_n)$, 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

$$P(-2\mathsf{ee}(\hat{\theta}) < \hat{\theta} - \theta < 2\mathsf{ee}(\hat{\theta})) \approx 0.95\,, \tag{7}$$

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

$$P(\hat{\theta} - 2\mathsf{ee}(\hat{\theta}) < \theta < \hat{\theta} + 2\mathsf{ee}(\hat{\theta})) \approx 0.95\,. \tag{8}$$

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

$$\hat{\mathsf{ee}}_{\mathsf{boot}}(\hat{\theta})\,, \tag{9}$$

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

5.0.1. Definición: 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

$$[\hat{\theta} - 2\hat{\mathsf{ee}}_{\mathsf{boot}}(\hat{\theta}), \hat{\theta} + 2\hat{\mathsf{ee}}_{\mathsf{boot}}(\hat{\theta})]\,. \tag{10}$$

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.

5.1. Ejemplo

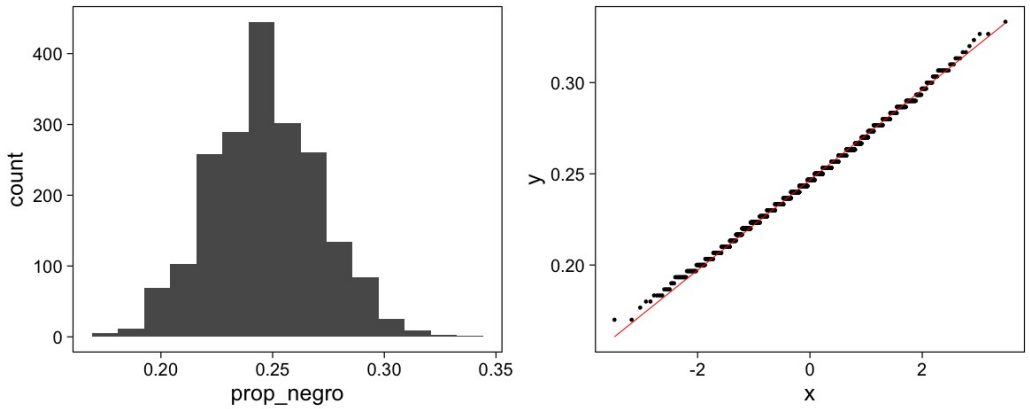
Consideremos la estimación que hicimos de el procentaje de tomadores de té que toma té negro:

```
## 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
}
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:

```
media_boot ← prop_negro_tbl ▷ pull(prop_negro) ▷ mean()
media_boot - prop_hat
```

```
[1] -0.00011667
```

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] 0.024178
```

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  0.295
```

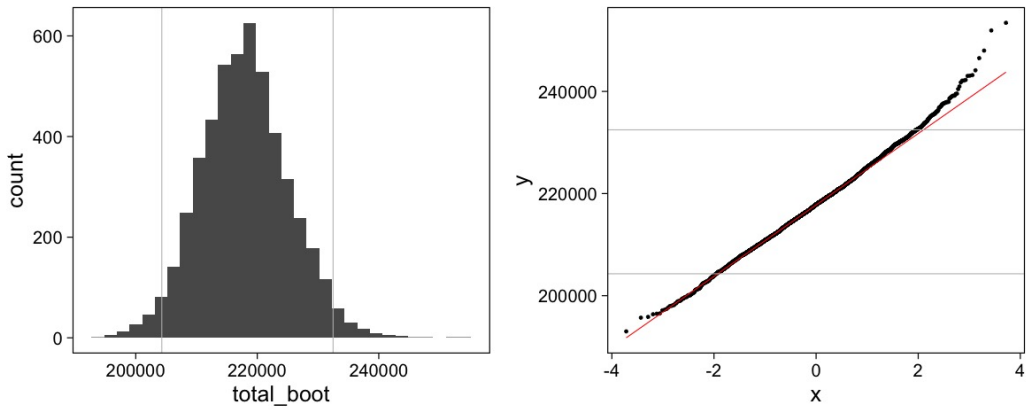
Este intervalo tiene probabilidad del 95 % de capturar al verdadero poblacional.

6. INVENTARIOS DE CASAS VENDIDAS

Ahora consideremos el problema de estimar el total del valor de las casas vendidas en un periodo. Tenemos una muestra de tamaño $n = 150$:

```
## muestra original
set.seed(121)
muestra_casas ← sample_n(poblacion_casas, size = 150)
## paso 1: define el estimador
calc_estimador_casas ← function(datos){
  N ← nrow(poblacion_casas)
  n ← nrow(datos)
  total_muestra ← sum(datos$precio_miles)
  estimador_total ← (N / n) * total_muestra
  estimador_total
}
```

```
## paso 2: define el proceso de remuestreo
muestra_boot ← function(datos){
  ## tomar muestra con reemplazo del mismo ñtamao
  sample_n(datos, size = nrow(datos), replace = TRUE)
}
```



En este caso, 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

```
total_est ← calc_estimador_casas(muestra_casas)
sesgo ← mean(totales_boot$total_boot) - total_est
sesgo
```

```
[1] 110.09
```

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

```
sesgo_relativo ← sesgo / total_est
sesgo_relativo
```

```
[1] 0.00050537
```

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

```
ee_boot ← sd(totales_boot$total_boot)
c(inf = total_est - 2*ee_boot, centro = total_est, sup = total_est + 2*ee_boot
    )
```

```
   inf centro    sup
203367 217832 232297
```

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

```
intervalo_total ← c(inf = total_est - 2*ee_boot,
                    centro = total_est,
                    sup = total_est + 2*ee_boot) / 1000
intervalo_total ▷ round(5)
```

```
   inf centro    sup
203.37 217.83 232.30
```

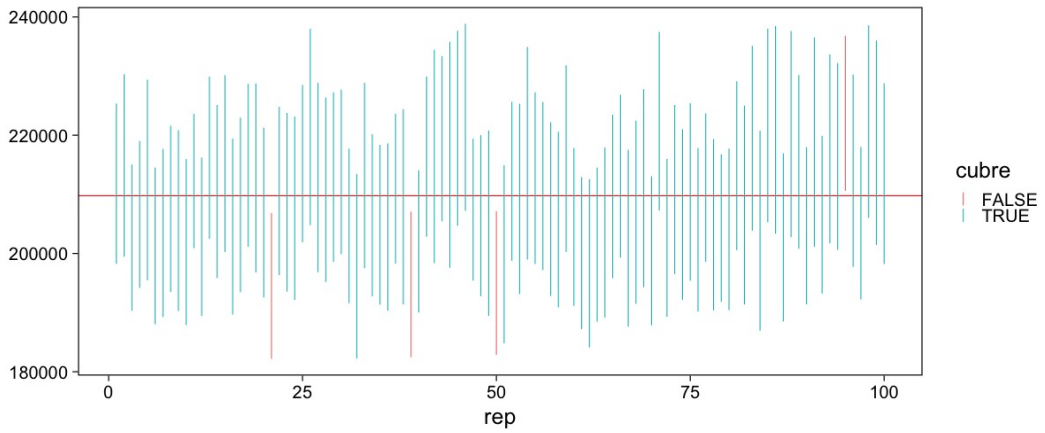
6.0.1. Nota: En este ejemplo mostraremos una alternativa de intervalos de confianza que es más apropiado cuando observamos asimetría. Sin embargo, primero tendremos que hablar de dos conceptos clave con respecto a intervalos de confianza: calibración e interpretación.

7. 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*?)

7.0.1. 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).

8. 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.

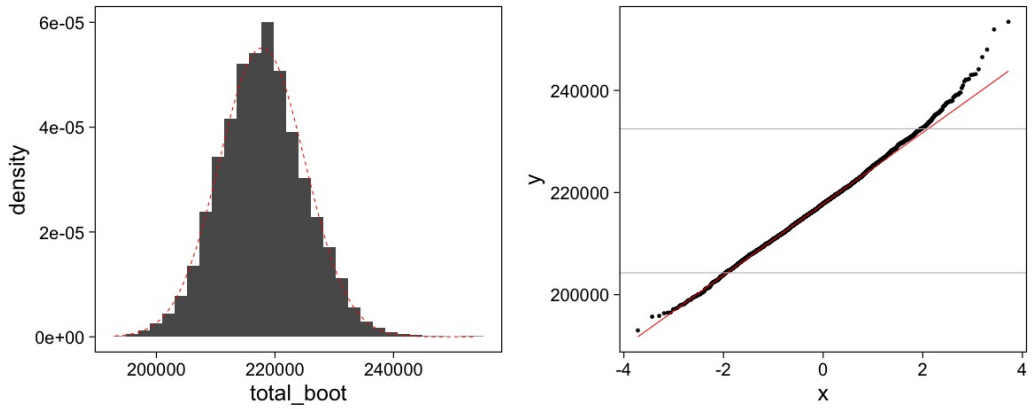
Sin embargo,

- 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.

9. 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

```
intervalo_total %>% round(1)
```

```
  inf centro    sup
203.4  217.8  232.3
```

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 ▷
  mutate(upper = total_boot ≥ max(intervalo_total * 1000),
         lower = total_boot ≤ min(intervalo_total * 1000)) ▷
  summarise(prop_inf = mean(lower),
            prop_sup = mean(upper))
```

```
# A tibble: 1 × 2
  prop_inf prop_sup
     <dbl>    <dbl>
1   0.0192    0.026
```

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*:

9.0.1. Definición: El **intervalo de percentiles *bootstrap*** al 95 % de confianza está dado por

$$[q_{0.025}, q_{0.975}]\,, \tag{11}$$

donde q_f es el percentil f de la distribución *bootstrap*.

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 ▷ pull(total_boot) %>%
  quantile(., probs = c(0.025, 0.50, 0.975)) / 1000
(intervalo_95) %>% round(1)
```

```
 2.5%   50% 97.5%
204.3 217.8 232.5
```

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 - total_est/1000)
```

```
      2.5%        50%      97.5%
13.5391163  0.0033367 14.6467418
```

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.

9.0.2. 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.

9.1. 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 ódigo
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)
}
```

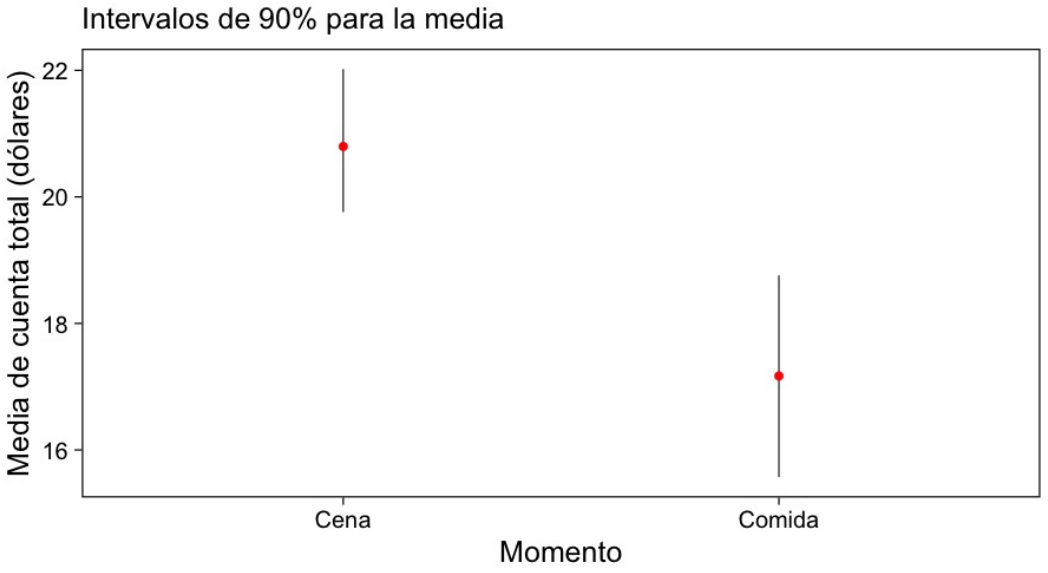
```
## paso 2: remuestrea y calcula estimador
boot_samples ← bootstraps(propinas, strata = momento, 1000) ▷
  mutate(res_boot = map(splits, estimador))
## paso 3: construye intervalos de confianza
intervalo_propinas_90 ← boot_samples ▷
  int_pctl(res_boot, alpha = 0.10) ▷
```

```
  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     19.8      20.8   22.0    0.1 percentile
2 Comida   15.6      17.1   18.8    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**):

9.1.1. 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 [244/88]> Bootstrap0001 <tibble [2 × 3]>
 2 <split [244/83]> Bootstrap0002 <tibble [2 × 3]>
 3 <split [244/92]> Bootstrap0003 <tibble [2 × 3]>
```

```
 4 <split [244/97]> Bootstrap0004 <tibble [2 × 3]>
 5 <split [244/81]> Bootstrap0005 <tibble [2 × 3]>
 6 <split [244/96]> Bootstrap0006 <tibble [2 × 3]>
 7 <split [244/95]> Bootstrap0007 <tibble [2 × 3]>
 8 <split [244/84]> Bootstrap0008 <tibble [2 × 3]>
 9 <split [244/92]> 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>
<244/88/244>
```

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

```
# A tibble: 244 × 7
# Groups:   id [156]
   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         17.0    1.01 No      Dom   Cena               2     1
 4         10.3    1.66 No      Dom   Cena               3     2
 5         10.3    1.66 No      Dom   Cena               3     2
 6         10.3    1.66 No      Dom   Cena               3     2
 7         21.0    3.5  No      Dom   Cena               3     3
 8         21.0    3.5  No      Dom   Cena               3     3
 9         23.7    3.31 No      Dom   Cena               2     4
10         23.7    3.31 No      Dom   Cena               2     4
# ... 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
```

2. El paquete `boot` está asociado al libro **Bootstrap Methods and Their Applications** 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) intevalos ABC con la función `abc.ci()`.
3. 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) intevalos ABC con la función `abcnon()`.

9.1.2. 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?

10. BOOSTRAP Y OTRAS ESTADÍSTICAS

El *bootstrap* es una técnica versátil. Un ejemplo son **estimadores de razón**, que tienen la forma

$$\hat{r} = \frac{\overline{y}}{\overline{x}}\,. \tag{12}$$

Por ejemplo, ¿cómo haríamos estimación para el procentaje de área area habitable de las casas en relación al tamaño del lote? Una manera de estimar esta cantidad es dividiendo la suma del área habitable de nuestra muestra y dividirlo entre la suma del área de los lotes de nuestra muestra, como en la fórmula anterior. Esta fórmula es más difícil pues tanto numerador como denominador tienen variabilidad, y estas dos cantidades no varían independientemente.

Con el *bootstrap* podemos atacar estos problemas.

10.1. Estimadores de razón

Nuestra muestra original es:

```
set.seed(250)
casas_muestra ← sample_n(poblacion_casas, 200)
casas_muestra ▷ as.data.frame() ▷ str()
```

```
'data.frame':   200 obs. of  46 variables:
 $ id                  : num  1166 855 579 1158 882 ...
 $ tipo_zona           : chr  "RL" "RL" "FV" "RL" ...
 $ frente_lote         : num  79 102 34 34 44 81 70 78 64 61 ...
 $ calle               : chr  "Pave" "Pave" "Pave" "Pave" ...
 $ forma_lote          : chr  "IR1" "Reg" "Reg" "IR1" ...
 $ nombre_zona         : chr  "NridgHt" "Sawyer" "Somerst" "NridgHt" ...
 $ tipo_edificio       : chr  "1Fam" "1Fam" "TwnhsE" "Twnhs" ...
 $ estilo              : chr  "1Story" "1Story" "2Story" "1Story" ...
 $ calidad_gral        : num  7 5 7 7 7 6 5 6 6 5 ...
 $ condicion_gral      : num  5 4 5 5 5 5 5 6 5 7 ...
 $ ñao_construccion    : num  2009 1955 2007 2007 1990 ...
```

```
 $ calidad_exteriores   : chr  "Gd" "TA" "Gd" "Gd" ...
 $ material_exteriores  : chr  "VinylSd" "Wd Sdng" "VinylSd" "VinylSd" ...
 $ condicion_exteriores : chr  "TA" "TA" "TA" "TA" ...
 $ calidad_sotano       : chr  "Gd" "TA" "Gd" "Gd" ...
 $ condicion_sotano     : chr  "TA" "TA" "TA" "TA" ...
 $ tipo_sotano          : chr  "Unf" "ALQ" "Unf" "GLQ" ...
 $ calefaccion          : chr  "GasA" "GasA" "GasA" "GasA" ...
 $ calidad_calefaccion  : chr  "Ex" "TA" "Ex" "Ex" ...
 $ aire_acondicionado   : chr  "Y" "Y" "Y" "Y" ...
 $ ñbaos_completos      : num  2 1 2 2 2 1 1 2 2 2 ...
 $ ñbaos_medios         : num  0 1 0 0 1 0 0 0 1 0 ...
 $ recamaras_sup        : num  3 3 2 2 3 3 3 3 3 3 ...
 $ calidad_cocina       : chr  "Gd" "TA" "Gd" "Gd" ...
 $ cuartos_sup          : num  7 6 5 6 7 5 6 7 7 5 ...
 $ tipo_garage          : chr  "Attchd" "Attchd" "Detchd" "Attchd" ...
 $ terminado_garage     : chr  "RFn" "Unf" "Unf" "RFn" ...
 $ num_coches           : num  2 2 2 2 2 0 0 2 2 2 ...
 $ calidad_garage       : chr  "TA" "TA" "TA" "TA" ...
 $ condicion_garage     : chr  "TA" "TA" "TA" "TA" ...
 $ ñao_venta            : num  2009 2006 2008 2009 2007 ...
 $ mes_venta            : num  9 7 2 7 4 5 12 6 2 9 ...
 $ tipo_venta           : chr  "New" "WD" "WD" "WD" ...
 $ condicion_venta      : chr  "Partial" "Abnorml" "Abnorml" "Normal" ...
 $ lat                  : num  42.1 42 42.1 42.1 42 ...
 $ long                 : num  -93.7 -93.7 -93.6 -93.7 -93.6 ...
 $ area_sotano_m2       : num  140 164 64 122 107 ...
 $ area_1er_piso_m2     : num  139.5 165.3 65.3 122.1 110.3 ...
 $ area_2o_piso_m2      : num  0 0 64 0 49.2 ...
 $ area_habitable_sup_m2: num  140 165 129 122 160 ...
 $ area_garage_m2       : num  59.8 42.2 50.2 58.2 37.2 ...
 $ area_lote_m2         : num  886 1665 335 465 1278 ...
 $ precio_miles         : num  233 170 146 230 188 ...
 $ valor_misc_miles     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ precio_m2_miles      : num  1.67 1.03 1.13 1.88 1.18 ...
 $ precio_m2            : num  1671 1029 1129 1884 1175 ...
```

El estimador de interés es:

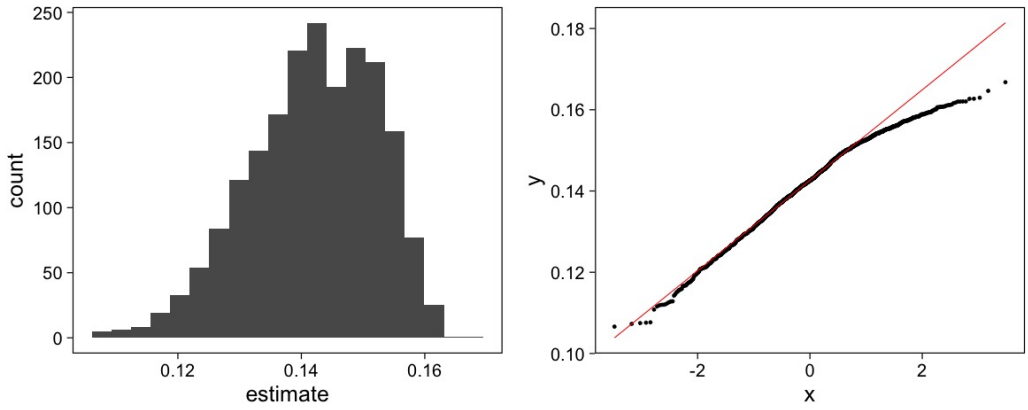
```
estimador_razon ← function(split, ...){
  muestra ← analysis(split)
  muestra ▷
    summarise(estimate = sum(area_habitable_sup_m2) / sum(area_lote_m2),
              .groups = "drop") ▷
    mutate(term = "area del lote construida")
}
```

Y nuestra estimación puntual es

```
estimador ← muestra_casas ▷
  summarise(estimate = sum(area_habitable_sup_m2) / sum(area_lote_m2))
estimador
```

```
# A tibble: 1 × 1
  estimate
     <dbl>
1    0.148
```

Es decir que en promedio, un poco más de 15 % del lote total es ocupado por área habitable. Ahora hacemos bootstrap para construir un intervalo:



En este caso la cola derecha parece tener menos dispersión que una distribución normal. Usamos un intervalo de percentiles para obtener:

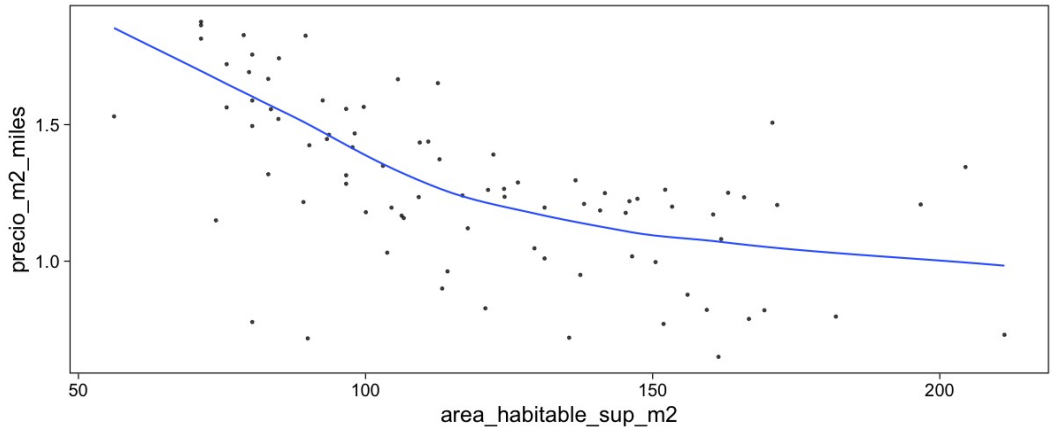
```
dist_boot ▷ int_pctl(res_boot) ▷
  mutate(estimador = estimador$estimate) ▷
  rename(media_boot = .estimate) ▷
  mutate(sesgo = media_boot - estimador) ▷
  select(-.method, -term)
```

```
# A tibble: 1 × 6
  .lower media_boot .upper .alpha estimador    sesgo
   <dbl>      <dbl>  <dbl>  <dbl>     <dbl>    <dbl>
1  0.121      0.142  0.159   0.05     0.148 -0.00560
```

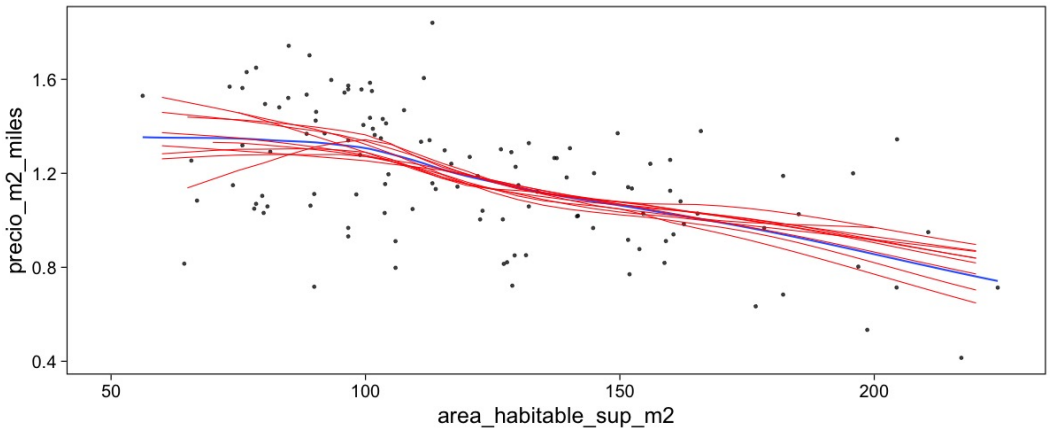
Nótese que el sesgo es bajo. De modo que en esta zona, entre 12 % y 16 % de toda el área disponible es ocupada por área habitable: estas son casas que tienen jardines o terrenos, garage relativamente grandes.

10.2. Suavizadores

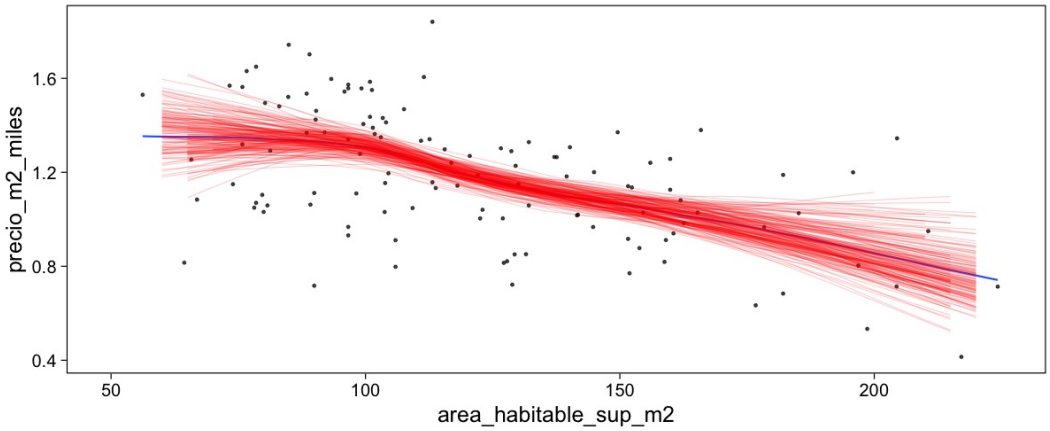
Podemos usar el *bootstrap* para juzgar la variabilidad de un suavizador, que consideramos como nuestra estadística:



Podemos hacer bootstrap para juzgar la estabilidad del suavizador:



Donde vemos que algunas cambios de pendiente del suavizador original no son muy interpretables (por ejemplo, para áreas chicas) y alta variabilidad en general en los extremos. Podemos hacer más iteraciones para calcular bandas de confianza:



Donde observamos cómo tenemos incertidumbre en cuanto al nivel y forma de las curvas en los extremos de los datos (casas grandes y chicas), lo cual es natural. Aunque podemos resumir para hacer bandas de confianza, mostrar remuestras de esta manera es informativo: por ejempo: vemos cómo es probable también que para casas de emnos de 70 metros cuadrados el precio por metro cuadrado no cambia tanto (líneas constantes)

11. 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] L. M. Chihara and T. C. Hesterberg. *Mathematical Statistics with Resampling and R*. John Wiley & Sons, Inc., Hoboken, NJ, USA, aug 2018. ISBN 978-1-119-50596-9 978-1-119-41654-8. . 8, 15

[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. . 9, 14, 25

[3] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer New York, New York, NY, 2004. ISBN 978-1-4419-2322-6 978-0-387-21736-9. . 9