microdatos

November 21, 2024

1 Minicursillo de tratamiento de microdatos

1.1 Introducción

Tenemos una población finita, llamémosla $\mathcal{U}$, con N miembros.

Cada miembro de la población tiene una serie de valores que desconocemos, por ejemplo:

- Renta
- Gasto mensual en comida
- Capital inicial de su último préstamo al consumo concedido
- Número de empleados de la empresa a la que pertenece

Los miembros de la población no tienen por qué ser personas, pueden ser hogares, empresas,…

Asumiremos por el momento que de todos los valores que posee cada miembro de la población, sólo nos interesa uno. Llamemos y_i al valor de interés que posee el i-ésimo elemento de la población.

1.2 Censos y encuestas

Evidentemente, salvo en los *censos* realizados en cada vez menos países, no es factible salir a preguntar a todos y cada uno de los elementos de la población su valor de interés.

Para solventar este problema, seleccionaremos una **muestra aleatoria** de la población, es decir:

1. Tomamos el conjunto $\mathcal{S}$ de todas las muestras posibles.

   Nótese que este conjunto es gigante, ya que tiene 2^N elementos, donde N, el tamaño de la población, no suele ser precisamente pequeño.

2. Asignamos a cada posible muestra $s \in \mathcal{S}$ una *probabilidad de selección* $p(s) \geq 0$. La suma de todas las probabilidades de selección debe ser 1.

3. Seleccionamos una muestra de entre todas las posibles acorde con las probabilidades definidas anteriormente.

Definir métodos eficientes para seleccionar muestras aleatorias y razonar sobre sus propiedades es un arte en sí mismo.

1.2.1 Ejemplo práctico: Muestreo aleatorio simple

Tenemos una población de 1.000 habitantes y estamos interesados en estudiar la *altura* de sus habitantes, para ello, tomaremos una muestra aleatoria de tamaño 100, para lo cual:

1. Todas las posibles muestras que no tengan tamaño 100 tendrán asignada una probabilidad de 0.
2. Todas las $\binom{N}{100}$ posibles muestras de tamaño 100 tendrán asignadas la misma probabilidad.

Hay muchos métodos eficientes para seleccionar una muestra con estas probabilidades, pero no pensemos en ello, y usemos alguno ya implementado en nuestra libería de confianza de nuestro lenguaje de programación favorito.

[1]:
```
import pandas as pd # librería para el análisis de datos en Python

# es habitual encontrarnos archivos en diversos formatos, "csv", "parquet",␣
↪"feather",...
poblacion = pd.read_feather('population.feather')
muestra = poblacion.sample(100) # función para tomar una muestra aleatoria␣
↪simple
# https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html
muestra
```

[1]:
```
         altura
570  175.829283
332  177.772291
212  178.664760
712  188.874173
941  149.586816
..          …
91   179.987499
402  168.900119
465  217.286064
191  145.696242
811  147.272761

[100 rows x 1 columns]
```

Lo importante con lo que nos debemos quedar es:

- Los valores y_i no son valores aleatorios, sino fijos.
- Lo único aleatorio aquí es la muestra s seleccionada, si repitiéramos el proceso de selección de la misma (volviéramos a ejecutar la casilla) obtentremos otra.

Unos *microdatos* simplificados serían entonces simplemente una lista, como la variable `muestra`, con tantos elementos como tamaño tenga la muestra, conteniendo cada elemento el valor y_i correspondiente al elemento de la población encuestado.

1.3 Estimación de totales poblacionales

Si quisiéramos estimar la suma de los valores y_i de toda la población, una posibilidad sería hacer una suma ponderada de los valores y_i de la muestra. Dicho de otra forma:

Si $Y := \sum_{i=1}^{N} y_i$, una posible estimación de Y sería $\widehat{Y} := \sum_{i \in s} \lambda_i y_i$.

Esta estimación $\widehat{Y}$, evidentemente es una valor aleatorio, ya que depende de la muestra s, que es aleatoria.

Esto puede observarse al ejecutar múltiples veces la siguiente casilla.

[2]:
```
muestra = poblacion.sample(100) # volvemos a seleccionar una muestra
estimacion = muestra.sum() # en este caso es una suma sin ponderar, para no␣
 ↪complicar las cosas
verdadero_valor = poblacion.sum()

# se observa que la estimación siempre está por debajo del valor real, con lo␣
 ↪que esta ponderación no parece muy buena
print(f'verdadero valor: {verdadero_valor}')
print(f'estimación: {estimacion}')
```

```
verdadero valor: altura    170734.577696
dtype: float64
estimación: altura    17130.8791
dtype: float64
```

1.3.1 Propiedades deseables de $\widehat{Y}$

Como buen valor aleatorio, la estimación $\widehat{Y}$ tendrá cierta *distribución*.

Para poder considerar "buena" a esta estimación, su distribución deberá estar *centrada* en torno al verdadero valor (que recordemos, es fijo) y no tener demasiada dispersión.

Más concretamente, con que un valor aleatorio esté *centrado* en torno a un valor, nos referimos a que su *media* o *esperanza* sea igual a dicho valor, es decir, queremos que $E(\widehat{Y}) = Y$.

Una buena ponderación Veamos si hay alguna ponderación que haga que nuestra propuesta de estimación sea razonable:

$$E(\widehat{Y}) = E\left(\sum_{i\in s} \lambda_i y_i\right) = E\left(\sum_{i=1}^{N} \lambda_i y_i I_i\right),$$

donde $I_i = 1$ si $i \in s$ y 0 en caso contrario. Está claro que I_i es una cantidad aleatoria, ya que depende de s, a su vez aleatoria. De hecho, es la única cantidad aleatoria en el lado derecho de la ecuación, ya que el resto son valores fijos.

Por las propiedades de la esperanza es claro que:

$$E(\widehat{Y}) = E\left(\sum_{i=1}^{N} \lambda_i y_i I_i\right) = \sum_{i=1}^{N} \lambda_i y_i E(I_i),$$

luego si definimos $\lambda_i = \frac{1}{E(I_i)}$ tenemos que $E(\widehat{Y}) = Y$, como queríamos. A estos λ_i se les suele llamar **factores de elevación** o **pesos de muestreo**.

Qué demonios es $E(I_i)$ Quedaría pendiente por tanto calcular los valores $E(I_i)$, pero esto es relativamente sencillo. En efecto:

$$E(I_i) = \sum_{s \in \mathcal{S}} p(s) I_i(s),$$

donde $I_i(s) = 1$ si $i \in s$ y 0 en caso contrario. Expresado de otra forma, $E(I_i)$ es la suma de las probabilidades de las muestras que contienen al elemento i, o sea, *la probabilidad de que la muestra aleatoria s contenga al elemento i de la población*:

$$E(I_i) = \sum_{s \ni i} p(s) = P(s \ni i),$$

por comodidad, definimos $\pi_i := E(I_i)$.

Recapitulando...

- Una estimación razonable de un total poblacional es $\widehat{Y} = \sum_{i \in s} \frac{y_i}{\pi_i}$, a este se le conoce como *estimador de Horvitz-Thomson*, o, simplemente, *estimador HT*.
- En el caso del muestreo aleatorio simple de nuestro ejemplo, hay $\binom{N-1}{n-1}$ muestras de n elementos que contengan al elemento i, luego

$$\pi_i = \binom{N-1}{n-1} \Big/ \binom{N}{n} = \frac{n}{N}$$

[3]:
```
muestra = poblacion.sample(100) # volvemos a seleccionar una muestra
estimacion_ht = (muestra*10).sum()
verdadero_valor = poblacion.sum()

print(f'verdadero valor: {verdadero_valor}')
print(f'estimación: {estimacion_ht}')
```

```
verdadero valor: altura    170734.577696
dtype: float64
estimación: altura    170095.183421
dtype: float64
```

1.3.2 Un caso más realista

Normalmente, las encuestas no usan muestreo aleatorio simple, sino formas mucho más complejas de tomar muestras.

Por este motivo, todos los microdatos que nos podamos encontrar no son simples listas de valores, sino *tablas*, donde

- Cada fila representa los datos asociados a un elemento de la población encuestado
- Cada columna representa un dato de interés por el se ha preguntado
- Hay una columna especial que contiene los factores de elevación, es decir, los valores $1/\pi_i$ de los elementos encuestados.

Con estos elementos ya somos capaces de hacer estimaciones de totales poblacionales a partir de los microdatos de una encuesta real.

[4]:
```python
# ejemplo con la EPF (encuesta de presupuestos familiares)
microdatos = pd.read_csv('EPFhogar_2023.tab', sep='\t') # en este caso los␣
↪microdatos son archivos csv separados por tabuladores en lugar de comas
microdatos
```

[4]:
```
       ANOENC  NUMERO  CCAA  NUTS1  CAPROV  TAMAMU  DENSIDAD  CLAVE  CLATEO  \
0        2023       1    16      2       6       5         2      1       2
1        2023       2     9      5       6       5         2      1       1
2        2023       3     1      6       6       2         2      2       2
3        2023       4     2      2       6       5         3      1       1
4        2023       5    15      2       6       3         1      1       2
…         …       …     …     …      …       …        …      …       …
20702    2023   20703    16      2       1       1         1      1       2
20703    2023   20704     6      1       6       5         3      2       2
20704    2023   20705     9      5       6       4         2      1       2
20705    2023   20706     7      4       6       5         3      1       1
20706    2023   20707     2      2       1       1         1      2       2

            FACTOR  …  FUENPRIN  FUENPRINRED  IMPEXAC  INTERIN  NUMPERI  \
0       400.954060  …       2.0          2.0     1206        3        1
1      1575.616402  …       2.0          2.0      859        2        1
2       962.465360  …       2.0          2.0      850        2        1
3       621.096916  …       3.0          3.0     2102        5        1
4       260.852439  …       2.0          2.0     3500        7        2
…              …   …        …            …        …        …
20702   314.700969  …       7.0          3.0     2603        6        1
20703   360.352153  …       3.0          3.0     2662        6        2
20704  1420.714232  …       3.0          3.0     2400        5        2
20705   432.570078  …       1.0          1.0     3067        7        2
20706   544.759442  …       2.0          2.0     4463        7        2

       COMIMH  COMISD  COMIHU  COMIINV  COMITOT
0           8       0       0        0        8
1          42       0       0        0       42
2          14       0       0        0       14
3          28       0       0        0       28
4          41       0       0        0       41
…         …       …       …        …
20702      14       0       0        0       14
20703      84       0       0        0       84
20704      28       0       0        0       28
20705      52       0       0        4       56
20706      23       0       0        0       23

[20707 rows x 188 columns]
```

Que no cunda el pánico Hay muchas columnas con nombres impronunciables, y a priori no sabemos lo que es nada.

Ante esto solo queda tener un poco de paciencia e ir leyendo, según lo vayamos necesitando, la documentación de la encuesta.

En este caso:

- Los elementos de la población encuestados son hogares, luego cada fila representa un hogar
- La columna *IMPEXAC* representa los ingresos mensuales netos del hogar
- La columna *FACTOR* representa el factor de elevación de cada hogar encuestado

[5]:
```
# estimemos la suma de los ingresos mensuales netos de los hogares
suma_total_ingresos = (microdatos['IMPEXAC'] * microdatos['FACTOR']).sum()
print(f'Estimación suma total ingresos netos: {suma_total_ingresos}')
```

```
Estimación suma total ingresos netos: 47800955199.41489
```

1.4 Más allá de los totales poblacionales

Evidentemente, hay vida más allá de estimar totales poblacionales, a continuación revisamos algunas de las tareas más habituales.

1.4.1 Subpoblaciones

En ocasiones también queremos estimar totales, pero no de la población completa, sino únicamente de un subconjunto de la misma.

Para hacer esto, basta con quedarnos con las filas de los microdatos asociadas a elementos de nuestra subpoblación de interés.

Por ejemplo, si nos interesa estimar el número de habitantes de Murcia:

- La columna *CCAA* representa el código de comunidad autónoma donde está el hogar encuestado
- La columna *NMIEMB* representa el número de miembros del hogar
- Nos quedaremos solo con los registros cuya columna *CCAA* sea 14
- Estimaremos como antes

[6]:
```
microdatos_murcia = microdatos[microdatos['CCAA'] == 14]
habitantes_murcia = (microdatos_murcia['NMIEMB'] * microdatos_murcia['FACTOR']).
 ↪sum()
print(f'Estimación de habitantes de murcia: {habitantes_murcia}')
```

```
Estimación de habitantes de murcia: 1551640.000032
```

1.4.2 Medias

En un mundo ideal... Si queremos estimar la media de una determinada variable, es decir, si queremos estimar

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{Y}{N}$$

podemos reutilizar lo aprendido para estimar totales, y usar como estimación $\hat{\bar{y}} := \frac{\widehat{Y}}{N}$, ya que

$$E(\hat{\bar{y}}) = E(\widehat{Y}/N) = \frac{E(\widehat{Y})}{N} = \frac{Y}{N} = \bar{y}.$$

En la realidad Lamentablemente, usualmente no conocemos N. Cuando esto suceda, podemos usar la siguiente estimación, llamada *estimación de razón*:

$$\hat{\bar{y}}^{\text{Rat}} = \frac{\widehat{Y}}{\widehat{N}}$$

Desgraciadamente, aquí ya no podemos asegurar que $E(\hat{\bar{y}}^{\text{Rat}}) = \bar{y}$, pero se puede demostrar que, a pesar de todo es un método de estimación con muy buenas propiedades.

Sabiendo todo esto, podemos estimar el ingreso neto mensual medio de los habitantes murcianos.

Recordemos que antes hicimos una estimación de los habitantes murcianos, la variable `habitantes_murcia`.

[7]:
```python
ingresos_totales_murcia = (microdatos_murcia['IMPEXAC'] *␣
  ↪microdatos_murcia['FACTOR']).sum()
ingresos_medios_murcia = ingresos_totales_murcia / habitantes_murcia
# Los ingresos salen tan bajos debido a que estamos contando también a los niños
print(f'Estimación de ingresos mensuales netos medios de los habitantes de␣
  ↪Murcia: {ingresos_medios_murcia}')
hogares_murcia = microdatos_murcia['FACTOR'].sum()
ingresos_medios_hogares_murcia = ingresos_totales_murcia / hogares_murcia
print(f'Estimación de ingresos mensuales netos medios de los hogares de Murcia:␣
  ↪{ingresos_medios_hogares_murcia}')
```

```
Estimación de ingresos mensuales netos medios de los habitantes de Murcia:
890.0072362814517
Estimación de ingresos mensuales netos medios de los hogares de Murcia:
2439.5954674951863
```

1.4.3 Cuantiles

- Los datos con los que trabajamos usualmente siguen distribuciones muy asimétricas (e.g. renta, patrimonio).
- Las medias, que son muy sensibles a estos fenómenos, suelen arrojar resultados de difícil interpretación.
- Por este motivo, suele ser un requisito la estimación de *cuantiles* (e.g. mediana, deciles, quintiles, percentiles varios,…)

Teóricamente… Se define el cuantil q de la población $\mathcal{U}$ como el valor y tal que la proporción de la población con valores menores o iguales que y es igual a q.

Escrito de una forma más compacta:

$$\frac{|\{y_k \in \mathcal{U} \text{ tales que } y_k \leq y\}|}{N} = q$$

El numerador puede ser reescrito como $Z_y := \sum_{k=1}^{N} z_y(y_k)$, que es un total poblacional, donde $z_y(y_k) = 1$ si $y_k \leq y$, y 0 en caso contrario.

Por ende, dado un y, con lo que ya sabemos, podemos estimar tanto el numerador como el denominador de la parte izquierda de la ecuación, obteniendo así la estimación $\widehat{Z_y}/\widehat{N}$ para la fracción completa Z_y/N.

Esta estimación recibe el nombre de *estimación de Hájek*.

En la práctica Sin entrar en detalles, lo que hacen la mayoría de los paquetes que calculan estimaciones de cuantiles es calcular $\widehat{Z_{y_i}}/\widehat{N}$ para cada $i \in s$ y ofrecer como estimación del cuantil q el valor y_i cuya fracción $\widehat{Z_{y_i}}/\widehat{N}$ asociada esté más cerca de q.

[8]:
```python
# calculemos el ingreso neto mensual mediano de los hogares de murcia
import numpy as np # paquete para cálculo numérico

# solo tenemos dos opciones, o picarlo nosotros o ver si lo ha picado alguien␣
↪antes y leer su documentación
# https://numpy.org/doc/stable/reference/generated/numpy.quantile.html
mediana = np.quantile(microdatos_murcia['IMPEXAC'], q=[0.5],␣
↪method='inverted_cdf', weights=microdatos_murcia['FACTOR'])[0]
print(f'Estimación de ingresos mensuales netos medianos de los hogares de␣
↪Murcia: {mediana}')
```

```
Estimación de ingresos mensuales netos medianos de los hogares de Murcia: 2175
```

1.4.4 División de la población por cuantiles

A veces, será interesante estudiar el comportamiento de determinada variable para cada cuantil de un población.

Por ejemplo, podemos estudiar cuál es la proporción de hogares de Murcia de cada decil de ingresos netos que vive en Murcia capital.

Hagamos esto paso por paso (hay muchas maneras de hacerlos):

[9]:
```python
# calculamos los deciles de ingresos netos de los hogares de murcia, como antes
q = np.linspace(0.1, 1, num=10) # [0.1, 0.2, 0.3,...,1]
deciles = np.quantile(microdatos_murcia['IMPEXAC'], q=q, method='inverted_cdf',␣
↪weights=microdatos_murcia['FACTOR'])

# añadimos a cada registro una columna que indique el decil de ingresos netos␣
↪al que pertenece
# https://pandas.pydata.org/docs/reference/api/pandas.cut.html
microdatos_murcia = microdatos_murcia.copy()
labels = [str(int(decil*10)) for decil in q[:-1]]
microdatos_murcia['DECIL'] = pd.cut(microdatos_murcia['IMPEXAC'], bins=deciles,␣
↪labels=labels)

microdatos_murcia[['IMPEXAC', 'DECIL']]
```

```
[9]:        IMPEXAC DECIL
16        1300     2
17        2741     6
18        1300     2
28        1323     2
65        3958     8
…            …   …
20664     3500     8
20667     2272     5
20676     1745     3
20698     1786     4
20699     2239     5

[948 rows x 2 columns]
```

La columna *CAPROV* de los microdatos, toma el valor 1 si el hogar está en una capital de provincia y 6 en caso contrario.

Vamos a crear una columna nueva, *CAPITAL*, cuyos valores serán los mismos que *CAPROV*, pero sustituyendo los 6 por 0.

```
[10]: # creamos una columna que indique pertenencia a una provincia
microdatos_murcia['CAPITAL'] = microdatos_murcia['CAPROV']
microdatos_murcia.loc[microdatos_murcia['CAPITAL'] == 6, 'CAPITAL'] = 0
microdatos_murcia['CAPITAL']
```

```
[10]: 16       0
17       0
18       1
28       0
65       0
        ..
20664    0
20667    0
20676    1
20698    0
20699    1
Name: CAPITAL, Length: 948, dtype: int64
```

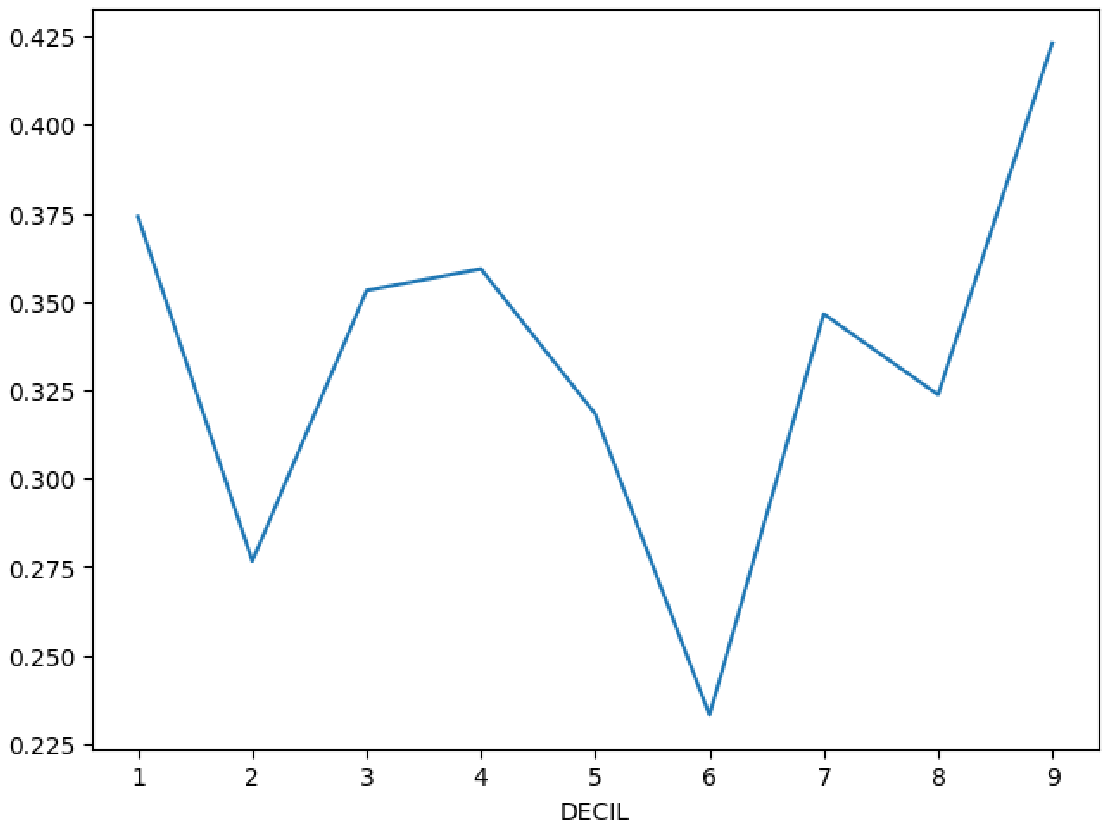
```
[11]: # agrupamos a la población por deciles
# https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html
murcia_grouped = microdatos_murcia.groupby('DECIL', observed=True)


# para cada decil, estimamos la proporción de hogares que vive en una capital␣
↪de provincia
proporcion_capital = murcia_grouped.apply(lambda group: (group['CAPITAL'] *␣
↪group['FACTOR']).sum() / group['FACTOR'].sum(), include_groups=False)
```

```
proporcion_capital.name = 'Proporción en capital de provincia'
proporcion_capital
```

[11]:
```
DECIL
1    0.374098
2    0.276620
3    0.353190
4    0.359286
5    0.318294
6    0.233082
7    0.346524
8    0.323708
9    0.423189
Name: Proporción en capital de provincia, dtype: float64
```

[12]:
```
# para finalizar, pintamos un gráfico con los resultados
import matplotlib.pyplot as plt
fig = plt.figure(layout='constrained')
ax = fig.add_subplot()
ax = proporcion_capital.plot(ax=ax)
```



1.4.5 Distribuciones

Para visualizar la distribución de una determinada variable, por ejemplo, los ingresos netos mensuales, se pueden usar varias herramientas, una de ellas son los *histogramas*.

Simplificando, estos dividen los valores de la variable en intervalos, y dibujan, por cada uno de ellos, una barra con altura proporcional al número de elementos de la población cuyo valor cae en dicho intervalo.

Por suerte, la mayoría de paquetes de gráficos tienen la opción de asignarle un peso a cada observación, con lo que usualmente no tendremos que picar demasiado código para conseguir el resultado deseado.

[13]:
```
histograma = plt.figure(layout='constrained')
ax_histograma = histograma.add_subplot()
ax_histograma = microdatos_murcia['IMPEXAC'].plot.
  ↪hist(weights=microdatos_murcia['FACTOR'], density=True, ax=ax_histograma)
```