Master en Big Data. Fundamentos Matemáticos del Análisis de Datos (FMAD).

Solución de la Práctica 0

Fernando San Segundo

Curso 2021-22. Última actualización: 2021-09-29

Ejercicios

1. Usando la función `sample` crea un vector `dado_honesto` con 100 números del 1 al 6. Haz una tabla de frecuencias absolutas (de dos maneras, con `table` y `dplyr`) y una tabla de frecuencias relativas.

- **Solución:**

```
set.seed(2021) # Para obtener reproducibilidad, pero es opcional
(dado  <-  sample(1:6, 100, replace = TRUE))
```

```
  [1] 6 6 2 4 4 6 6 3 6 6 5 1 4 3 4 2 3 4 5 3 6 2 4 5 6 2 3
 [28] 4 5 6 5 1 6 2 3 3 2 6 6 6 2 5 6 3 2 1 1 6 5 4 4 6 3 3
 [55] 2 1 2 1 1 1 1 5 3 3 1 4 6 6 6 2 1 3 4 2 5 2 6 6 4 6 6
 [82] 3 4 5 1 6 5 3 1 5 3 1 3 6 4 6 6 5 1 3
```

La tabla de frecuencias absolutas:

```
table(dado)
```

```
dado
 1  2  3  4  5  6
15 13 18 14 13 27
```

y la de frecuencias realtivas:

```
prop.table(table(dado))
```

```
dado
   1    2    3    4    5    6
0.15 0.13 0.18 0.14 0.13 0.27
```

Ahora recreamos ambas tablas usando `dplyr` (hay muchas otras maneras de hacer esto):

```
library(tidyverse) # Empezamos cargando el tidyverse (incluye dplyr y ggplot)

tibble(dado) %>%
  count(dado) %>%
  mutate(frecAbsoluta = n, frecRelativa = frecAbsoluta / sum(n), n = NULL)
```

```
# A tibble: 6 x 3
   dado frecAbsoluta frecRelativa
  <int>        <int>        <dbl>
1     1           15         0.15
2     2           13         0.13
3     3           18         0.18
4     4           14         0.14
5     5           13         0.13
6     6           27         0.27
```

2. A continuación crea un nuevo vector `dado_cargado` de manera que la probabilidad de que el número elegido valga 6 sea el doble que la probabilidad de elegir cualquiera de los cinco números restantes. Lee la ayuda de `sample` si lo necesitas. De nuevo, haz tablas de frecuencias absolutas y relativas de este segundo vector.

- **Solución:**

```
(dadoCargado  <-  sample(1:6, 100, replace = TRUE, prob = c(rep(1, 5), 2)))
```

```
  [1] 5 6 6 6 5 4 1 6 3 1 6 3 3 1 4 6 6 4 1 1 6 4 3 6 6 5 6
 [28] 5 2 6 5 5 3 6 6 4 6 2 2 3 4 4 6 2 5 6 6 6 6 6 5 5 2 6
 [55] 3 5 4 3 2 5 4 3 4 6 6 5 6 1 3 4 6 4 3 6 5 1 4 6 3 3 6
 [82] 6 4 6 4 6 6 6 6 6 2 1 5 2 4 2 6 6 5 2
```

La tabla de frecuencias absolutas:

```
table(dadoCargado)
```

```
dadoCargado
 1  2  3  4  5  6
 8 10 13 16 15 38
```

y la de frecuencias realtivas:

```
prop.table(table(dadoCargado))
```

```
dadoCargado
   1    2    3    4    5    6
0.08 0.10 0.13 0.16 0.15 0.38
```

Puede verse claramente el efecto de la modificación en la tabla de probabilidades.
Ahora usando `dplyr`:

```
tibble(dadoCargado) %>%
  count(dadoCargado) %>%
  mutate(frecAbsoluta = n, frecRelativa = frecAbsoluta / sum(n), n = NULL)
```

```
# A tibble: 6 x 3
  dadoCargado frecAbsoluta frecRelativa
        <int>        <int>        <dbl>
1           1            8         0.08
2           2           10         0.1
3           3           13         0.13
4           4           16         0.16
5           5           15         0.15
6           6           38         0.38
```

3. Utiliza las funciones `rep` y `seq` para crear tres vectores `v1`, `v2` y `v3` con estos elementos respectivamente:

```
4, 4, 4, 4, 3, 3, 3, 3, 2, 2, 2, 2, 1, 1, 1, 1
```

```
1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5
```

```
1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4
```

- **Solución:** Para el primer vector:

```
rep(4:1, each = 4)
```

```
 [1] 4 4 4 4 3 3 3 3 2 2 2 2 1 1 1 1
```

Para el segundo

```
rep(1:5, times = 1:5)
```

```
 [1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
```

Y para el tercero:

```
rep(1:4, times = 4)
```

```
 [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
```

pero también:

```
rep(1:4, length.out = 16)
```

```
 [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
```

4. Utilizando la tabla `mpg` de la librería `tidyverse` crea una tabla `mpg2` que:
   - contenga las filas en las que la variable `class` toma el valor `pickup`.
   - y las columnas de la tabla original cuyos nombres empiezan por `c`. No se trata de que las selecciones *a mano*, por sus nombres. Busca información sobre funciones auxiliares para `select` en la Sección 5.4 de R4DS.

- **Solución:** Se muestran sólo las 10 primeras filas de la tabla `mpg2` usando `slice_head`.

```
mpg2 <- mpg %>%
  filter(class == "pickup") %>%
  select(starts_with("c"))

slice_head(mpg2, n = 10)
```

```
# A tibble: 10 x 3
     cyl   cty class
   <int> <int> <chr>
 1     6    15 pickup
 2     6    14 pickup
 3     6    13 pickup
 4     6    14 pickup
 5     8    14 pickup
 6     8    14 pickup
 7     8     9 pickup
 8     8    11 pickup
 9     8    11 pickup
10     8    12 pickup
```

5. Descarga el fichero census.dta. Averigua de qué tipo de fichero se trata y usa la herramienta `Import DataSet` del panel `Environment` de RStudio para leer con R los datos de ese fichero. Asegúrate de copiar en esta práctica los dos primeros comandos que llevan a cabo la importación (excluye el comando `View`) y que descubrirás al usar esa herramienta. Después completa los siguientes apartados con esos datos y usando `dplyr` y `ggplot`:

- **Solución:** Empezamos descargando y leyendo el fichero mediante `read_dta` de la librería *haven*. Es un fichero de Stata.

```
library(haven)
census <- read_dta("http://www.stata-press.com/data/r8/census.dta")
```

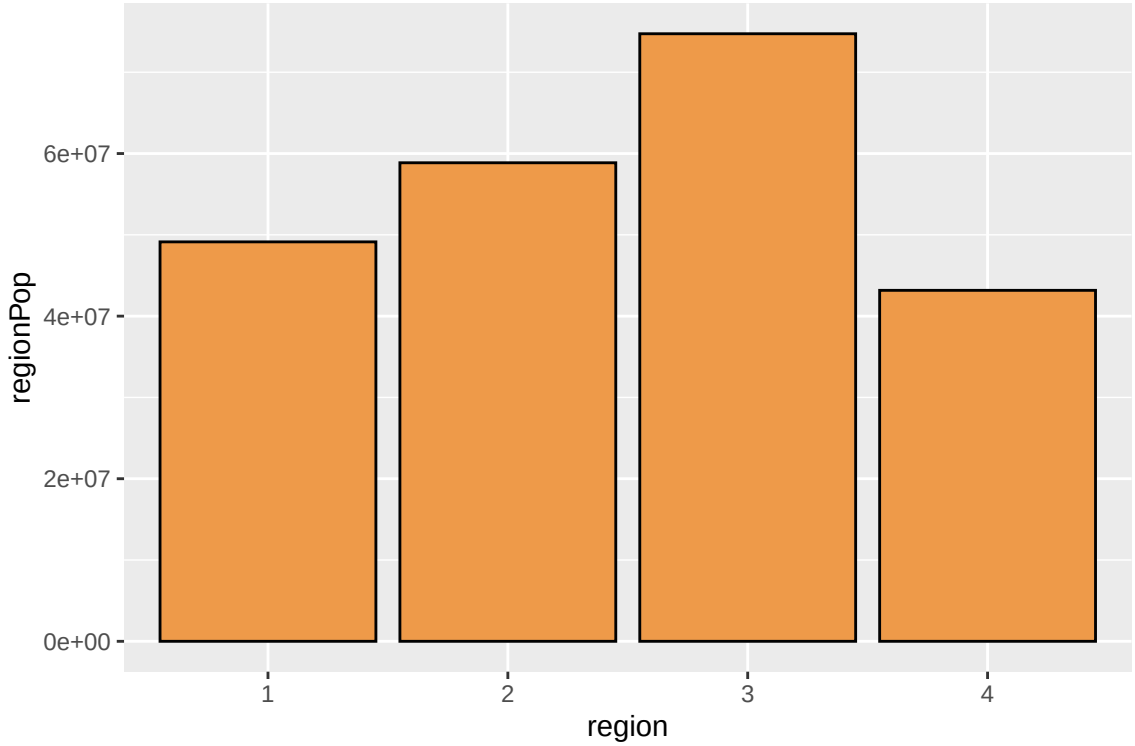
  - ¿Cuáles son las poblaciones totales de las regiones censales?
    Solución: Las obtenemos con `group_by` y `summarise`

```
(pobTotales <- census %>%
  mutate(region = factor(region)) %>%
  group_by(region) %>%
summarise(regionPop = sum(pop)))
```

```
# A tibble: 4 x 2
  region regionPop
  <fct>      <dbl>
1 1       49135283
2 2       58865670
3 3       74734029
4 4       43172490
```

- Representa esas poblaciones totales en un diagrama de barras (una barra por región censal).
  Solución: Usamos `geom_col` (y no `geom_bar`) porque las alturas de las barras del diagrama corresponden a la variable `regionPop` de nuestra tabla. Usaríamos `geom_bar` si quisiéramos que R calculara una tabla de frecuencias de una variable y la usara para asignar las alturas.

```r
ggplot(pobTotales) +
  geom_col(aes(x = region, y = regionPop),
           color = "black", fill = "tan2") +
  theme(legend.position='none')
```



- Ordena los estados por población, de mayor a menor.
  Solución: es un uso directo de `arrange`. Usamo `slice_head` y `slice_tail` para ver el principio y fin de esa tabla:

```r
census %>%
  arrange(desc(pop)) %>%
  select(state, pop) %>%
  slice_head(n = 6)
```

```
# A tibble: 6 x 2
  state             pop
  <chr>           <dbl>
1 California   23667902
2 New York     17558072
3 Texas        14229191
4 Pennsylvania 11863895
5 Illinois     11426518
6 Ohio         10797630
```

```r
census %>%
  arrange(desc(pop)) %>%
  select(state, pop) %>%
  slice_tail(n = 6)
```

```
# A tibble: 6 x 2
  state        pop
  <chr>      <dbl>
1 S. Dakota 690768
2 N. Dakota 652717
3 Delaware  594338
4 Vermont   511456
5 Wyoming   469557
6 Alaska    401851
```

- Crea una nueva variable que contenga la tasa de divorcios /matrimonios para cada estado.
Solución: Es fácil usando `mutate`.

```
census %>%
  mutate(divorceRate = divorce / marriage) %>%
  select(state, divorceRate)
```

```
# A tibble: 50 x 2
   state       divorceRate
   <chr>             <dbl>
 1 Alabama           0.546
 2 Alaska            0.656
 3 Arizona           0.659
 4 Arkansas          0.599
 5 California        0.633
 6 Colorado          0.532
 7 Connecticut       0.518
 8 Delaware          0.521
 9 Florida           0.661
10 Georgia           0.492
# ... with 40 more rows
```

Es conveniente darse cuenta de que ni las parejas que se divorcian un año no son un subconjunto de las parejas que se casan ese año ni viceversa.

- Si nos preguntamos cuáles son los estados más envejecidos podemos responder de dos maneras. Mirando la edad mediana o mirando en qué estados la franja de mayor edad representa una proporción más alta de la población total. Haz una tabla en la que aparezcan los valores de estos dos criterios, ordenada según la edad mediana decreciente y muestra los 10 primeros estados de esa tabla.
Solución: Es importante no usar `select` demasiado pronto, antes de hacer el `mutate` que crea la nueva variable que representa la proporción de la franja de mayor edad.

```
census %>%
  mutate(propOlder = pop65p / pop) %>%
  arrange(desc(medage)) %>%
  select(state, medage, propOlder) %>%
  slice_head(n = 10)
```

```
# A tibble: 10 x 3
   state         medage propOlder
   <chr>          <dbl>     <dbl>
 1 Florida         34.7     0.173
 2 New Jersey      32.2     0.117
```

```
 3 Pennsylvania    32.1     0.129
 4 Connecticut     32       0.117
 5 New York        31.9     0.123
 6 Rhode Island    31.8     0.134
 7 Massachusetts   31.2     0.127
 8 Missouri        30.9     0.132
 9 Arkansas        30.6     0.137
10 Maine           30.4     0.125
```

- Haz un histograma (con 10 intervalos) de los valores de la variable `medage` (edad mediana) y con la curva de densidad de la variable superpuesta.
  Solución: hemos usado el parámetro `adjust` de la curva de densidad para *suavizarla* ligeramente.

```
census %>%
  ggplot(aes(x = medage)) +
  geom_histogram(aes(y = stat(density)),
                 bins = 10, fill= "tan2", color = "black") +
  geom_density(color = "red", size = 1.5, adjust = 1.7)
```