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

Tarea 2

Departamento de Matemática Aplicada

Curso 2021-22. Última actualización: 2021-09-30

Instrucciones preliminares

- Empieza abriendo el proyecto de RStudio correspondiente a tu repositorio personal de la asignatura.
- En todas las tareas tendrás que repetir un proceso como el descrito en la sección *Repite los pasos Creando un fichero Rmarkdown para esta práctica* de la *Práctica00*. Puedes releer la sección *Practicando la entrega de las Tareas* de esa misma práctica para recordar el procedimiento de entrega.

Ejercicio 1. Simulando variables aleatorias discretas.

Apartado 1: La variable aleatoria discreta $X1$ tiene esta tabla de densidad de probabilidad (es la variable que se usa como ejemplo en la Sesión):

valor de $X1$	0	1	2	3
Probabilidad de ese valor $P(X = x_i)$	$\frac{64}{125}$	$\frac{48}{125}$	$\frac{12}{125}$	$\frac{1}{125}$

Calcula la media y la varianza teóricas de esta variable.

Solución: La fórmula teórica para la media es:

$$\mu_X = \sum_{i=0}^{3} x_i P(x = x_i)$$

Hacemos esa suma con R:

```
valores = 0:3
probs = c(64, 48, 12, 1) / 125
(mu = sum(valores * probs))
```

```
## [1] 0.6
```

Y la varianza teórica es:

$$\sigma_X^2 = \sum_{i=0}^{3} (x_i - \mu)^2 P(x = x_i)$$

que con R se traduce en:

```r
(sigma2 = sum((valores - mu)^2 * probs))
```

```
## [1] 0.48
```

Apartado 2: Combina `sample` con `replicate` para simular cien mil muestras de tamaño 10 de esta variable $X1$. Estudia la distribución de las medias muestrales como hemos hecho en ejemplos previos, ilustrando con gráficas la distribución de esas medias muestrales. Cambia después el tamaño de la muestra a 30 y repite el análisis.

Solución: El código necesario es este:

```r
N = 100000 # número de muestras
n = 10 # tamaño de cada muestra
medias_muestrales_10 = replicate(N, {
  mean(sample(valores, size = n, prob = probs, replace = TRUE))
})
```

Con esto hemos obtenido 100000 muestras de X y hemos almacenado la media muestral de cada una de ellas. Las primeras son:

```r
head(medias_muestrales_10)
```

```
## [1] 0.6 0.3 0.7 0.6 0.6 0.7
```

Y la media de las medias muestrales (que es una estimación de μ) resulta ser:

```r
mean(medias_muestrales_10)
```

```
## [1] 0.600554
```

¿Cómo de dispersas son estas medias muestrales alrededor de μ? Lo medimos calculando:

```r
sd(medias_muestrales_10)
```

```
## [1] 0.2198854
```

Si repetimos este proceso con tamaños muestrales:

```r
N = 100000 # número de muestras
n = 30 # tamaño de cada muestra
medias_muestrales_30 = replicate(N, {
  mean(sample(valores, size = n, prob = probs, replace = TRUE))
})

mean(medias_muestrales_30)
```

```
## [1] 0.599768
```

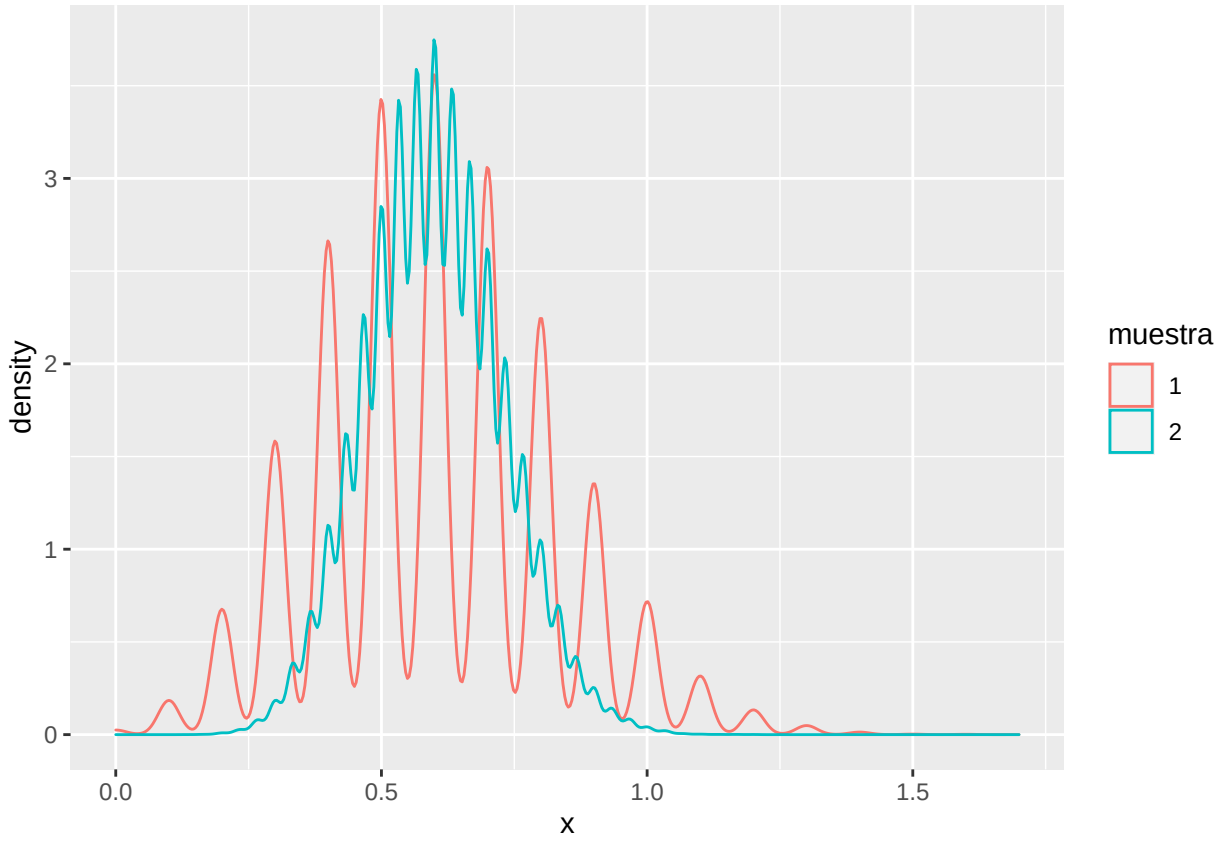
La aproximación es similar, pero la dispersión de las medias muestrales es:

```
sd(medias_muestrales_30)
```

```
## [1] 0.1264834
```

Como puede verse, tanto el propio valor como la dispersión indican que la estimación ha mejorado con el aumento del tamaño muestral. Las curvas de densidad de las medias muestrales para ambos tamaños muestrales son:

```
library(tidyverse)
tibble(x = c(n10 = medias_muestrales_10,
       n30 = medias_muestrales_30),
       muestra = gl(n = 2, k = N)) %>%
  ggplot() +
  geom_density(aes(x, color = muestra))
```



El patrón que se aprecia en estas figuras es típico de la distribución muestral de la media de variables discretas con pocos valores. Por un lado vemos curvas con muchas oscilaciones, que revelan la estructura discreta subyacente (esencialmente, hay un pico por cada valor posible de la media muestral). Pero la envolvente de la curva recuerda a la típica forma de campana de la normal, porque el Teorema Central del Límite empieza a manifestarse. Recomendamos encarecidamente que cambies el valor $n = 30$ por $n = 150$ y observes como el TCL es cada vez más influyente en la forma de la distribución de las medias muestrales. A medida que n aumenta la forma de campana se impone sobre las oscilaciones discretas.

Apartado 3: La variable aleatoria discreta $X2$ tiene esta tabla de densidad de probabilidad:

valor de $X2$	0	1	2
Probabilidad de ese valor $P(X = x_i)$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$

Suponemos que $X1$ y $X2$ son independientes. ¿Qué valores puede tomar la suma $X1 + X2$? ¿Cuál es su tabla de probabilidad?

Solución: Vamos a guardar los valores de X_1 y sus probabilidades con nombres cómodos:

```
vX1 = valores
pX1 = probs
```

Y ahora hacemos lo mismo con $X2$:

```
vX2 = 0:2
pX2 = c(1/2, 1/4, 1/4)
```

Para obtener la tabla de densidad de probabilidad de la variable suma todo puede hacerse *"manualmente"* (nos referimos, por ejemplo a soluciones con bucles for anidados). Pero incluimos dos soluciones distintas usando `outer` y *dplyr* que pueden resultar interesantes. La primera usa la función `outer`, que es útil para combinar los elementos de dos vectores de todas las formas posibles:

```
(vSuma = outer(vX1, vX2, function(x, y)x + y))
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    2
## [2,]    1    2    3
## [3,]    2    3    4
## [4,]    3    4    5
```

```
(pSuma = outer(pX1, pX2, function(x, y)x * y))
```

```
##       [,1]  [,2]  [,3]
## [1,] 0.256 0.128 0.128
## [2,] 0.192 0.096 0.096
## [3,] 0.048 0.024 0.024
## [4,] 0.004 0.002 0.002
```

```
tb_Suma = tibble(vSuma = c(vSuma), pSuma = c(pSuma)) %>%
  group_by(vSuma) %>%
  summarise(pSuma = sum(pSuma))
tb_Suma
```

```
## # A tibble: 6 x 2
##   vSuma pSuma
##   <int> <dbl>
## 1     0 0.256
## 2     1 0.32
## 3     2 0.272
## 4     3 0.124
## 5     4 0.026
## 6     5 0.002
```

La segunda es una solución puramente basada en *dplyr* que utiliza la función `full_join` para obtener el *producto* de las dos tablas. Recomendamos leer el Capítulo 13 de R4DS y después consultar la ayuda de `full_join` para entender el argumento `by= character()` que hemos usado debajo:

```
X1 <-  tibble(val1 = vX1, prob1 = pX1)
X2 <-  tibble(val2 = vX2, prob2 = pX2)

(Xsum <- full_join(X1, X2, by = character()))
```

```
## # A tibble: 12 x 4
##     val1 prob1  val2 prob2
##    <int> <dbl> <int> <dbl>
##  1     0 0.512     0  0.5
##  2     0 0.512     1  0.25
##  3     0 0.512     2  0.25
##  4     1 0.384     0  0.5
##  5     1 0.384     1  0.25
##  6     1 0.384     2  0.25
##  7     2 0.096     0  0.5
##  8     2 0.096     1  0.25
##  9     2 0.096     2  0.25
## 10     3 0.008     0  0.5
## 11     3 0.008     1  0.25
## 12     3 0.008     2  0.25
```

Y ahora que tenemos toda la información de las dos variables condensada en una tabla podemos usar las herramientas conocidas de *dplyr* para llegar al resultado:

```
Xsum <- Xsum %>%
  mutate(val = val1 + val2,
         prob = prob1 * prob2,
         .keep = "none") %>%
  group_by(val) %>%
  summarise(prob = sum(prob))
Xsum
```

```
## # A tibble: 6 x 2
##     val  prob
##   <int> <dbl>
## 1     0 0.256
## 2     1 0.32
## 3     2 0.272
## 4     3 0.124
## 5     4 0.026
## 6     5 0.002
```

Apartado 4: Calcula la media teórica de la suma $X_1 + X_2$. Después usa `sample` y `replicate` para simular cien mil *valores* de esta variable suma. Calcula la media de esos valores. *Advertencia:* no es el mismo tipo de análisis que hemos hecho en el segundo apartado.

Solución: Una vez que tenemos la tabla de probabilidad de $X_1 + X_2$ en un `tibble`, es fácil calcular su media:

```
muSum = Xsum %>%
  summarise(mu = sum(val * prob)) %>%
  pull(mu)
```

Las medias de $X1$ y $X2$ eran respectivamente

```
mu1 = X1 %>%
  summarise(mu = sum(val1 * prob1)) %>%
  pull(mu)
mu2 = X2 %>%
  summarise(mu = sum(val2 * prob2)) %>%
  pull(mu)
```

cuya suma, por supuesto, coincide con la media de la suma:

```
mu1 + mu2
```

```
## [1] 1.35
```

```
muSum
```

```
## [1] 1.35
```

Ahora vamos a generar 100000 valores de la variable suma $X1 + X2$ usando su tabla de probabilidad con `sample` y `replicate`:

```
N = 100000 # número de muestras
valoresSuma = replicate(N, {
  sample(Xsum$val, size = 1, prob = Xsum$prob)
})
head(valoresSuma)
```

```
## [1] 2 0 0 1 3 0
```

Si hacemos una tabla de frecuencias relativas de estos 100000 valores:

```
prop.table(table(valoresSuma))
```

```
## valoresSuma
##       0       1       2       3       4       5
## 0.25708 0.31808 0.27221 0.12379 0.02672 0.00212
```

vemos que están razonablemente cerca de los valores teóricos:

```
Xsum$prob
```

```
## [1] 0.256 0.320 0.272 0.124 0.026 0.002
```

Y la media de esos 100000 valores (empíricos) también está razonablemente cerca de la media (teórica) de esta variable, que recordemos que es 1.35:

```
mean(valoresSuma)
```

```
## [1] 1.35135
```

Ejercicio 2. Datos limpios

Descarga el fichero *testResults.csv* de este enlace (hemos usado un acortador de URLs con respecto a la dirección del enunciado original):

https://bit.ly/3ogqQXw

Solución:

```
URLdata = "https://bit.ly/3ogqQXw"
```

```
testResults = read_csv(URLdata)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   name = col_character(),
##   id = col_double(),
##   gender_age = col_character(),
##   test_number = col_double(),
##   week1 = col_double(),
##   week2 = col_double(),
##   week3 = col_double(),
##   week4 = col_double(),
##   week5 = col_double()
## )
```

- Este fichero contiene las notas de los alumnos de una clase, que hicieron dos tests cada semana durante cinco semanas. La tabla de datos no cumple los principios de *tidy data* que hemos visto en clase. Tu tarea en este ejercicio es explicar por qué no se cumplen y obtener una tabla de datos limpios con la misma información usando *tidyR*.
  Indicación: lee la ayuda de la función `separate` de *tidyR*.

Solución: los datos no son limpios porque: + hay valores de variables que aparecen como nombres de columna. Por ejemplo las columnas llamadas `week1` hasta `week5`

```
testResults <- testResults %>%
  pivot_longer(cols = starts_with("week"),
               names_prefix = "week",
               names_to = "week",
               values_to = "grade") %>%
  mutate(week = as.integer(week))
head(testResults)
```

```
## # A tibble: 6 x 6
##   name     id gender_age test_number  week grade
##   <chr> <dbl> <chr>            <dbl> <int> <dbl>
## 1 Jacob   108 m_20                 1     1     8
## 2 Jacob   108 m_20                 1     2     5
## 3 Jacob   108 m_20                 1     3     7
## 4 Jacob   108 m_20                 1     4     5
## 5 Jacob   108 m_20                 1     5     6
## 6 Jacob   108 m_20                 2     1     2
```

A continuación tratamos la columna `gender_age` que contiene dos variables. El remedio es usar `separate` como apuntaba el enunciado:

```
testResults <- testResults %>%
  separate(gender_age, sep = "_", into = c("gender", "age"))
```

La tabla que hemos obtenido ya es una tabla limpia.

```
testResults %>%
  slice_head(n = 15)
```

```
## # A tibble: 15 x 7
##    name       id gender age   test_number  week grade
##    <chr>   <dbl> <chr>  <chr>       <dbl> <int> <dbl>
##  1 Jacob     108 m      20              1     1     8
##  2 Jacob     108 m      20              1     2     5
##  3 Jacob     108 m      20              1     3     7
##  4 Jacob     108 m      20              1     4     5
##  5 Jacob     108 m      20              1     5     6
##  6 Jacob     108 m      20              2     1     2
##  7 Jacob     108 m      20              2     2     2
##  8 Jacob     108 m      20              2     3     4
##  9 Jacob     108 m      20              2     4     0
## 10 Jacob     108 m      20              2     5     3
## 11 Michael   490 m      19              1     1    10
## 12 Michael   490 m      19              1     2     0
## 13 Michael   490 m      19              1     3     5
## 14 Michael   490 m      19              1     4     4
## 15 Michael   490 m      19              1     5     0
```

Ejercicio 3. Lectura de R4DS.

Continuando con nuestra *lectura conjunta* de este libro, si revisas el índice verás que hemos cubierto (holgadamente en algún caso) el contenido de los Capítulos 6, 8, 9, 10 y 11. Todos esos Capítulos son relativamente ligeros. Por eso esta semana conviene detenerse un poco en la lectura de los Capítulos 7 y 12, que son los más densos en información. Y como motivación os proponemos un par de ejercicios, uno por cada uno de esos capítulos.

- Haz el ejercicio 2 de la Sección 7.5.1.1 de R4DS. Las ideas de esa sección son importantes para nuestro trabajo de las próximas sesiones.

Solución: para este ejercicio os recomendamos la lectura de la solución propuesta en la Web de J.B. Arnold de la que ya hemos hablado en la anterior tarea.

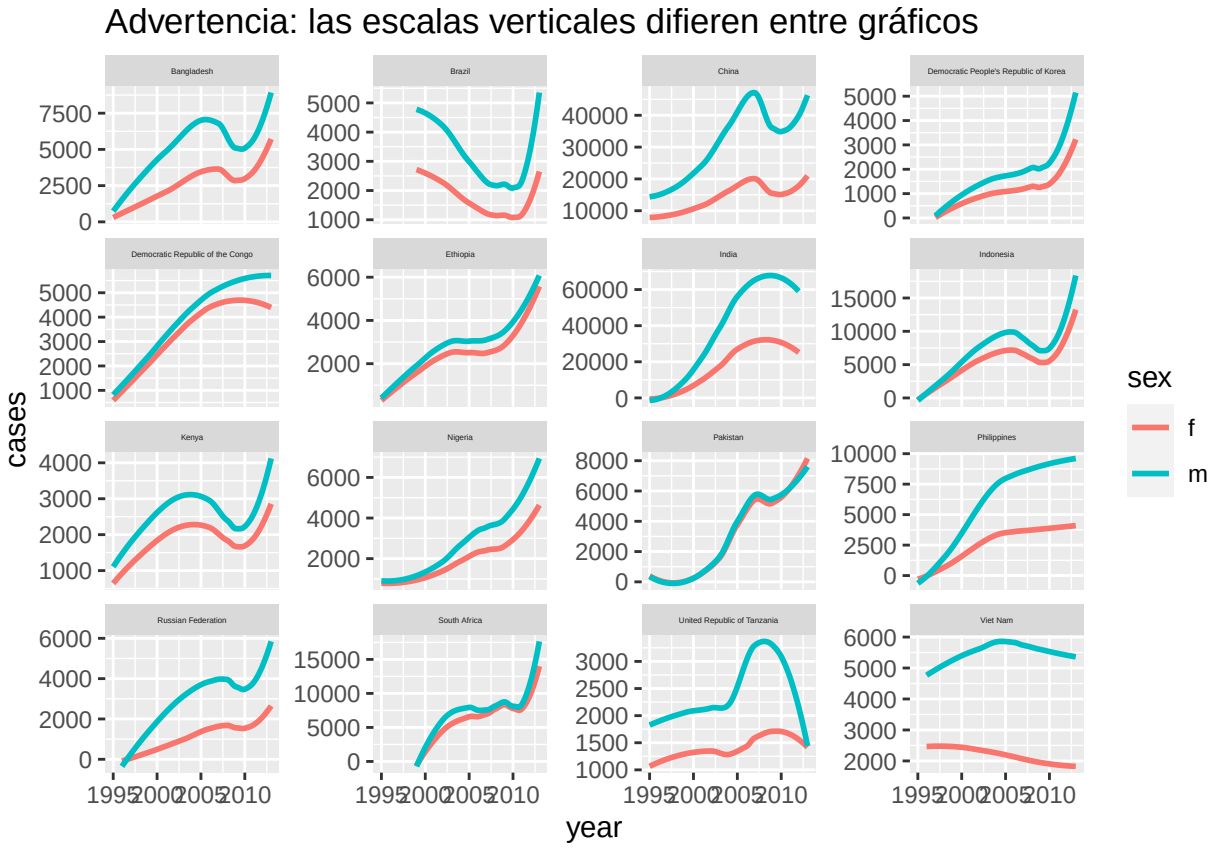
- Haz el ejercicio 4 de la Sección 12.6.1 de R4DS. ¡Aprovecha el código previo de esa sección para trabajar con datos limpios!

Solución: de nuevo, es muy recomendable leer la solución en la Web de Arnold. Recuerda que antes de usar esa solución debes haber ejecutado este código de R4DS.

```r
who5 <- who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  ) %>%
  mutate(
    key = stringr::str_replace(key, "newrel", "new_rel")
  ) %>%
  separate(key, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

Aparte de la solución que aparece allí puede ser interesante explorar otra opción en la que, en primer lugar, nos limitamos a los 16 paises cuya suma de casos totales a lo largo del periodo de estudio es superior a medio millón. Y a continuación usamos `facet_wrap` para mostrar para cada uno de esos países cuáles son las curvas de evolución de casos en cada género a lo largo de los años:

```r
who5 %>%
  group_by(country) %>%
  mutate(totalCases = sum(cases)) %>%
  filter(totalCases > 5 * 10^5) %>%
  ggplot(aes(x = year, y = cases, color = sex)) +
  geom_smooth(se=FALSE) +
  facet_wrap(~ country, scales = "free_y") +
  ggtitle("Advertencia: las escalas verticales difieren entre gráficos") +
    theme( strip.text = element_text(size = 3))
```



Hemos usado la opción `strip.text` para hacer más pequeño el título de cada subgráfico, a expensas de la legibilidad. Sería quizá preferible usar `fct_recode` (ver el Capítulo 15 de R4DS) para abreviar algunos nombres de países.

En general, para la exploración de un conjunto de datos como este es razonable recurrir a una herramienta como Shiny que permite adaptar la visualización dinámicamente a medida que nos hacemos preguntas sobre las variables.