PEC 1 – APRENDIZAJE Y ANÁLISIS ESTADÍSTICO

PARTE 1

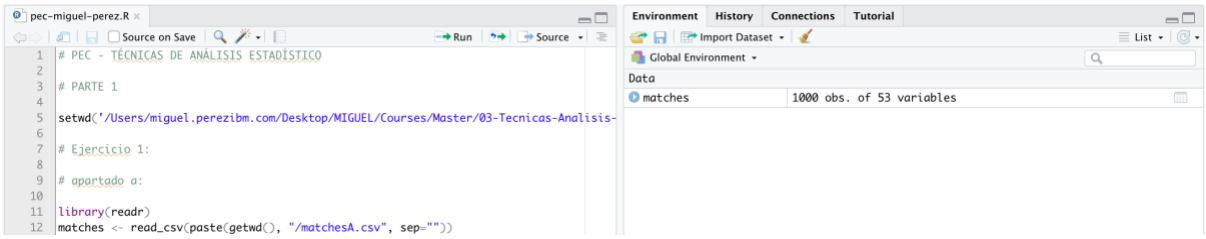
En esta práctica se va a realizar un análisis de estadística descriptiva sobre los datos contenidos en un dataset.

El conjunto de datos con que trabajaremos – llamado "matches"– contiene información de las partidas de un jugador extraída directamente del juego League of Legends. Este jugador ha participado en 1000 partidas en el último año y quiere conocer sus datos de rendimiento.
Nota: en caso de que haya algún dato no disponible o con valor incorrecto en la variable de interés podemos descartar esas filas sin problema.

1. **(Básico) En League of Legends, la variable de rendimiento más importante posiblemente sea el oro acumulado durante la partida. En el dataset se encuentra en la columna "goldEarned". Se pide:**

   a) **Cargar el fichero "matchesA.csv".**

   En primer lugar, nos situamos en el directorio donde se encuentra el fichero mediante setwd(), y después hacemos uso de la función *read_csv* del paquete *readr* para leer el fichero, que recibe como argumento la ruta del fichero. En la siguiente imagen se puede comprobar el código ejecutado, así como el dataframe generado en la pestaña entorno a la derecha de la imagen:



   b) **Calcular la media de oro sin usar funciones directas de R.**

   Para el cálculo del ejercicio se han sumado todos los elementos de la columna mediante la función *sum* y se ha dividido por la longitud de esta obtenida a través de la función *length*. Código ejecutado:

```
14 # apartado b:
15
16 mean_goldEarned <- sum(matches$goldEarned)/length(matches$goldEarned)
17 paste(mean_goldEarned)
18
```

Resultado:

```
> mean_goldEarned <- sum(matches$goldEarned)/length(matches$goldEarned)
> paste(mean_goldEarned)
[1] "11832.493"
>
```

c) Comprobar que la media de oro, calculada usando funciones directas de R, coincide con la anterior.

Hacemos uso de la función *mean* y comprobamos que el resultado es idéntico:

```
19  # apartado c:
20
21  mean_goldEarned1 <- mean(matches$goldEarned)
22  paste(mean_goldEarned1)
```

Resultado:

```
> # apartado c:
>
> mean_goldEarned1 <- mean(matches$goldEarned)
> paste(mean_goldEarned1)
[1] "11832.493"
>
```

Ambos resultados son de 11832.493.

d) Calcular la desviación típica sin usar funciones directas de R.

Para el cálculo de la desviación típica aplicamos la fórmula matemática en R. Es necesario especificar que se ha dividido por el número de registros menos 1, ya que ese es el denominador que se usa en la fórmula de la función sd de R, que se usa para calcular la desviación típica, y nos es útil seguir el mismo procedimiento para poder compararlos en el siguiente apartado:

```
24  # apartado d:
25
26  sd_goldEarned <- sqrt(sum((matches$goldEarned - mean_goldEarned)^2) /
27                        (length(matches$goldEarned)-1))
28  paste(sd_goldEarned)
```

Resultado:

```
> # apartado d:
>
> sd_goldEarned <- sqrt(sum((matches$goldEarned - mean_goldEarned)^2) /
+                       (length(matches$goldEarned)-1))
> paste(sd_goldEarned)
[1] "4892.70625548412"
```

e) Comprobar que la desviación típica, calculada usando funciones directas de R, coincide con la anterior.

Usamos la función *sd* de R, y comprobamos que el resultado es idéntico:

```
30  # apartado e:
31
32  sd_goldEarned1 <- sd(matches$goldEarned)
33  paste(sd_goldEarned1)
```

Resultado:

```
> sd_goldEarned1 <- sd(matches$goldEarned)
> paste(sd_goldEarned1)
[1] "4892.70625548412"
>
```

Se puede comprobar que el resultado es idéntico al del apartado anterior e igual a 4892.70625548412.

f) Obtener el valor máximo y mínimo y el rango.

Para la obtención de los valores máximo y mínimo se hace uso de las funciones *max* y *min* respectivamente. El rango, como sabemos que comprende los valores entre el mínimo y el máximo, lo generamos a partir de estos valores con la ayuda de la función *paste*. También se podría haber calculado de forma similar a los anteriores con la función *range*. Para la presentación del resultado se genera una tabla con los valores solicitados, se guarda en formato txt y se convierte al formato tabla.

```
35  # apartado f:
36
37  max_goldEarned <- max(matches$goldEarned)
38  min_goldEarned <- min(matches$goldEarned)
39  range_goldEarned <-paste(c("[", min_goldEarned, "-", max_goldEarned, "]"),
40                           collapse = "")
41
42  table_goldEarned <- matrix(c(max_goldEarned, min_goldEarned, range_goldEarned),
43                             ncol = 3, byrow = TRUE)
44
45  colnames(table_goldEarned) <- c("Máximo", "Mínimo", "Rango")
46
47  table_goldEarned <- as.table(table_goldEarned)
48  write.table(table_goldEarned, file = "ej1-apf.txt", sep = ",")
```

Resultado:

Máximo	Mínimo	Rango
33224	738	[738-33224]

g) Calcular los cuartiles de la variable oro.

Los cuartiles son Q1, Q2, y Q3, que representan el 25%, el 50% y el 75% de los datos de una variable. Para calcularlos se hace uso de la función *quantile*, especificando los 3 valores que se necesitan, y se vuelve a presentar el resultado en formato tabla.

```
# apartado g:

quartile_goldEarned <- quantile(matches$goldEarned, c(0.25, 0.5, 0.75))

table_quantile <- matrix(c(quartile_goldEarned[1], quartile_goldEarned[2],
                           quartile_goldEarned[3]),
                           ncol = 3, byrow = TRUE)

colnames(table_quantile) <- c("Q1", "Q2", "Q3")

table_quantile <- as.table(table_quantile)
write.table(table_quantile, file = "ej1-apg.txt", sep = ",")
```

Resultado:

Q1	Q2	Q3
8524.25	11459.5	14406

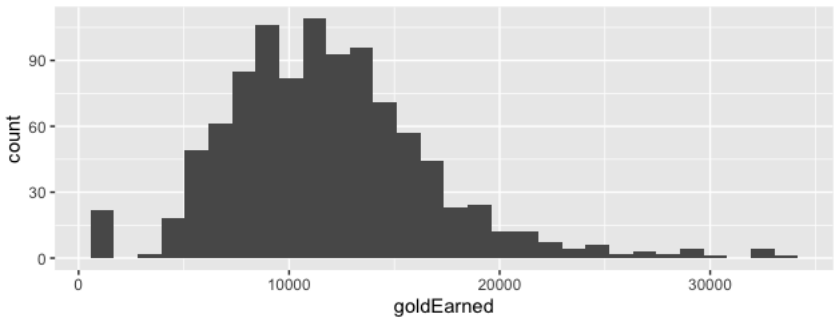
h) Dibujar el histograma para el oro ganado.

En este apartado, se hace uso de la librería ggplot2 para generar el histograma solicitado:

```
# apartado h:

library(ggplot2)
ggplot(data=matches, aes(goldEarned)) + geom_histogram()
```

Resultado:



2. **(Básico) En este tipo de juegos, el número de kills (enemigos derrotados), assists (asistencias a compañeros) o deaths (veces que se ha muerto) por sí solos no suelen ser muy indicativos, porque dependen de la longitud de la partida. Para ello, se suele usar una medida llamada KDA. El KDA se calcula sumando los kills y los assists, y luego dividiendo por el número de deaths (las columnas están en el dataset). Crea un vector que contenga, para cada jugador, su KDA.**

Para generar el vector, directamente creamos una columna nueva en el dataset inicial porque nos es de utilidad para la presentación de los resultados. En cuanto al procedimiento, he de destacar que en caso de que no haya ninguna muerte, el kda es igual al resultado de la suma de kills y assists multiplicado por dos, evitando así que la suma se divida por 0 y el resultado sea infinito.

```
68  # Ejercicio 2:
69
70  matches$kda <- ifelse(matches$deaths==0, round(2*(matches$kills+matches$assists),
71                                                 digits=2),
72                        round((matches$kills + matches$assists)/matches$deaths,
73                        digits=2))
74  library(magrittr)
75  library(dplyr)
76
77  matches_sample <- matches %>%
78    select("kills", "deaths", "assists", "kda") %>%
79    sample_n(10)
80  write.table(matches_sample, file = "ej2.txt", sep = ",", row.names=FALSE)
81
```

Para la presentación del resultado, se ha hecho uso del paquete *magrittr* que nos permite el uso de los pipes, y del paquete *dplyr* que contiene funciones para manipular un dataframe. Gracias a estos dos paquetes, hemos generado un dataframe con las variables que se usan en este ejercicio y se han recogido 10 valores aleatorios para presentarlos en formato de tabla como resultado del ejercicio:

kills	deaths	assists	kda
1	9	7	0.89
16	13	18	2.62
13	1	7	20
7	8	6	1.62
2	10	8	1
16	4	15	7.75
2	12	7	0.75
5	6	8	2.17
1	4	3	1
2	8	1	0.38

3. **(Avanzado) No obstante, nos podemos haber dado cuenta de una cosa: estamos mezclando partidas GANADAS y partidas PERDIDAS. Así, quizás las medias sean distintas en ambos casos. Necesitamos:**

   a) **Calcular la media del oro obtenido tanto en las partidas perdidas como en las ganadas.**

Para la realización de este apartado y los siguientes, se filtra por el valor de la variable *win* de forma que se generan dos dataframe, uno con los registros de las partidas ganadas y otro con el de las perdidas.

Con la función *mean* generamos la media de oro obtenido para las partidas ganadas y perdidas, y guardamos el resultado en formato tabla.

```
84  # apartado a:
85
86  matches_win <- filter(matches, win == TRUE)
87  matches_lost <- filter(matches, win == FALSE)
88
89  mean_goldEarned_win <- round(mean(matches_win$goldEarned), digits=2)
90  mean_goldEarned_lost <- round(mean(matches_lost$goldEarned), digits=2)
91
92  table_mean <- matrix(c(mean_goldEarned_win, mean_goldEarned_lost),
93                         ncol = 2, byrow = TRUE)
94
95  colnames(table_mean) <- c("Media Partidas Ganadas", "Media Partidas Perdidas")
96
97  table_mean <- as.table(table_mean)
98  write.table(table_mean, file = "ej3-apa.txt", sep = ",", row.names=FALSE)
99
```

Resultado:

Media Partidas Ganadas	Media Partidas Perdidas
13039.91	10644.24

   b) **Calcular también por separado la desviación típica del oro ganado.**

Mediante la función *sd* generamos la desviación típica para las partidas ganadas y perdidas. El resultado se almacena nuevamente en una tabla:

```r
# apartado b:

sd_goldEarned_win <- round(sd(matches_win$goldEarned), digits=2)
sd_goldEarned_lost <- round(sd(matches_lost$goldEarned), digits=2)

table_sd <- matrix(c(sd_goldEarned_win, sd_goldEarned_lost),
                   ncol = 2, byrow = TRUE)

colnames(table_sd) <- c("Sd Partidas Ganadas", "Sd Partidas Perdidas")

table_sd <- as.table(table_sd)
write.table(table_sd, file = "ej3-apb.txt", sep = ",", row.names=FALSE)

```

Resultado:

Sd Partidas Ganadas	Sd Partidas Perdidas
5131.65	4333.2

c) Calcular la mediana del oro obtenido en las partidas perdidas y el obtenido en las ganadas.

Para la mediana seguimos el mismo procedimiento comentado para los dos apartados anteriores, siendo en este caso la función *median* la que se ha de utilizar:

```r
# apartado c:

median_goldEarned_win <- median(matches_win$goldEarned)
median_goldEarned_lost <- median(matches_lost$goldEarned)

table_mediana <- matrix(c(median_goldEarned_win, median_goldEarned_lost),
                    ncol = 2, byrow = TRUE)

colnames(table_mediana) <- c("Mediana Partidas Ganadas",
                             "Mediana Partidas Perdidas")

table_mediana <- as.table(table_mediana)
write.table(table_mediana, file = "ej3-apc.txt", sep = ",", row.names=FALSE)
```

Resultado:

Mediana Partidas Ganadas	Mediana Partidas Perdidas
12688	10091.5

d) Calcular los cuartiles para ambos casos.

Al igual que en el primer ejercicio, calculamos los cuartiles con la función *quantile*, especificando los valores que queremos, que son los correspondientes al 25%, 50%, que es la mediana, y el 75%.

```r
# apartado d:

quartile_goldEarned_win <- quantile(matches_win$goldEarned, c(0.25, 0.5, 0.75))
quartile_goldEarned_lost <- quantile(matches_lost$goldEarned, c(0.25, 0.5, 0.75))

table_quartile <- matrix(c(quartile_goldEarned_win[1],
                           quartile_goldEarned_win[2],
                           quartile_goldEarned_win[3],
                           quartile_goldEarned_lost[1],
                           quartile_goldEarned_lost[2],
                           quartile_goldEarned_lost[3]),
                         ncol = 3, byrow = TRUE)

colnames(table_quartile) <- c("Q1", "Q2", "Q3")
rownames(table_quartile) <- c("Partidas Ganadas","Partidas Perdidas")

table_quartile <- as.table(table_quartile)
write.table(table_quartile, file = "ej3-apd.txt", sep = ",")

```

Resultado:

	Q1	Q2	Q3
Partidas Ganadas	9884.5	12688	15562.75
Partidas Perdidas	7741.5	10091.5	13070

e) Comentar los resultados obtenidos teniendo en cuenta el contexto de los datos.

Para comentar los resultados, se ha generado una tabla con los valores de la media, mediana, desviación estándar, Q1 y Q3 correspondientes a las partidas ganadas, perdidas y al total de partidas.

```r
# apartado e:

table_win_lost <- matrix(c(mean_goldEarned_win, sd_goldEarned_win,
                           quartile_goldEarned_win[1],
                           median_goldEarned_win,
                           quartile_goldEarned_win[3],
                           mean_goldEarned_lost, sd_goldEarned_lost,
                           quartile_goldEarned_lost[1],
                           median_goldEarned_lost,
                           quartile_goldEarned_lost[3],
                           mean_goldEarned, sd_goldEarned,
                           quartile_goldEarned[1], quartile_goldEarned[2],
                           quartile_goldEarned[3]), ncol = 5, byrow = TRUE)

colnames(table_win_lost) <- c("Mean","Sd","Quartile 25%", "Median",
                              "Quartile 75%")
rownames(table_win_lost) <- c("Partidas Ganadas","Partidas Perdidas",
                              "Partidas Totales")

table_win_lost <- as.table(table_win_lost)
write.table(table_win_lost, file = "ej3-ape.txt", sep = ",")
```

Resultado:

	Mean	Sd	Quartile 25%	Median	Quartile 75%
Partidas Ganadas	13039.91	5131.65	9884.5	12688	15562.75
Partidas Perdidas	10644.24	4333.2	7741.5	10091.5	13070
Partidas Totales	11832.493	4892.71	8524.25	11459.5	14406

Para comentar los resultados, parto de la base de que las partidas ganadas obtienen más oro, puesto que se entiende que el oro es una recompensa al rendimiento que se haya tenido en la partida, por lo que, si se gana, el rendimiento será mejor, y la cantidad de oro también debe ser mayor.

Siendo el primer cuartil el valor que representa el 25% de los datos ordenados de forma ascendente, la mediana el 50% y el tercer cuartil el 75%, se entiende que los valores de la media, el primer cuartil, la mediana y el tercer cuartil sean superiores en las partidas ganadas que, en las totales, y a su vez superiores en las partidas totales que, en las perdidas, por lo que tiene sentido el resultado obtenido.

Para comentar el valor de la desviación estándar es necesario recordar que ésta es una medida de dispersión que indica qué tan dispersos están los datos con respecto a la media. Por lo tanto, se observa que la variación de oro ganado es superior en las partidas ganadas que en las perdidas y, a diferencia de las medidas anteriores, donde el valor esperado era el obtenido, en este caso, no podía esperar que la dispersión fuera mayor en la partidas ganadas o perdidas.

PARTE 2

Ahora vamos a realizar un análisis de probabilidad y distribución sobre los datos contenidos en un dataset. El conjunto de datos con que trabajaremos – llamado "BoosterPacks"– contiene información sobre centenares de sobres de cartas Hearthstone que compró y abrió un jugador.

El dataset está formado por 10 columnas. La primera es poco importante y es el número del pack abierto. Cada sobre de cartas contiene 5 cartas y el jugador anota el número de:

- Cartas comunes con marco normal (Com).
- Cartas raras con marco normal (Rare).
- Cartas épicas con marco normal (Epic).
- Cartas legendarias con marco normal (Leg).
- Cartas comunes con marco dorado (gCom).
- Cartas raras con marco dorado (gRare).
- Cartas épicas con marco dorado (gEpic).
- Cartas legendarias con marco dorado (gLeg).
- Coste medio de maná (coste de invocación) de las cartas del sobre (Mana).

4. (Básico) Vamos a intentar determinar la probabilidad de obtener un tipo u otro de carta. Para ello:

a) Carga el dataset

Al igual que en el primer ejercicio, hacemos uso de la función *read_csv* del paquete *readr* para leer el dataset y almacenarlo, tal y como se puede ver en la siguiente imagen:

b) Rellena la tabla inferior (con las frecuencias absolutas y relativas de cada caso)

Para el cálculo de las frecuencias absolutas hay que sumar el número de veces que aparece cada tipo de carta, por lo que es la suma de los valores de cada variable. En las doradas, se calcula con la suma de los 4 tipos de carta con marco dorado, y para las no doradas se resta del total de cartas la frecuencia absoluta de las cartas dorada puesto que la suma de ambas da como resultado el total de las cartas.

La frecuencia relativa es la frecuencia absoluta dividida por el número total de cartas, que se ha calculado multiplicando el número de registros del dataframe, que coincide con el número de sobres, por el número de cartas en cada sobre, que es 5.

```r
# apartado b

total_cartas <- 5*dim(boosterpacks)[1]

frec_abs_com <- sum(boosterpacks$Com)
frec_abs_rare <- sum(boosterpacks$Rare)
frec_abs_grare <- sum(boosterpacks$gRare)
frec_abs_dorada <- sum(boosterpacks$gCom) + sum(boosterpacks$gRare) +
  sum(boosterpacks$gEpic) + sum(boosterpacks$gLeg)
frec_abs_no_dorada <- total_cartas - frec_abs_dorada

frec_rel_com <- round(frec_abs_com/total_cartas, digits=2)
frec_rel_rare <- round(frec_abs_rare/total_cartas, digits=2)
frec_rel_grare <- round(frec_abs_grare/total_cartas, digits=2)
frec_rel_dorada <- round(frec_abs_dorada/total_cartas, digits=2)
frec_rel_no_dorada <- round(frec_abs_no_dorada/total_cartas, digits=2)

boosterpacks_freq <- matrix(c(frec_abs_com, frec_abs_rare, frec_abs_grare,
                              frec_abs_no_dorada, frec_abs_dorada,
                              frec_rel_com, frec_rel_rare, frec_rel_grare,
                              frec_rel_no_dorada, frec_rel_dorada),
                            ncol = 5, byrow = TRUE)

colnames(boosterpacks_freq) <- c("Com", "Rare", "gRare", "No Dorada", "Dorada")
rownames(boosterpacks_freq) <- c("Cartas Totales", "Prob. relativa")

boosterpacks_freq <- as.table(boosterpacks_freq)
write.table(boosterpacks_freq, file = "ej4-apb.txt", sep = ",")
```

El resultado se almacena en la siguiente tabla:

	Com	Rare	gRare	No Dorada	Dorada
Cartas Totales	3861	1155	85	5320	185
Prob. relativa	0.7	0.21	0.02	0.97	0.03

5. **(Básico) Del ejercicio anterior tenemos la probabilidad de que una carta sea dorada o no dorada. Si una baraja de Hearthstone tiene 30 cartas en total y la construyo aleatoriamente a partir de mi muestra, calcula la probabilidad de que:**

En este ejercicio se piden probabilidades respecto a que en un conjunto de 30 cartas, que se construye aleatoriamente a partir de la muestra, exista un número de cartas doradas determinado. Por lo tanto, interpreto que sigue una distribución binomial, puesto que hay dos posibilidades, que la carta sea dorada o que no lo sea, y considero que las probabilidades son constantes, ya que no he interpretado que haya un total de cartas 'x' del que se vayan cogiendo cartas sin reposición de forma que la probabilidad no es constante, sino que, como en un videojuego, no hay un número definido de cartas totales del que se vayan extrayendo cartas para el mazo, sino que se puede decir que las cartas son infinitas y la probabilidad permanece constante.

a) Tenga exactamente una carta dorada.

Para calcular la probabilidad de que haya una carta dorada, tenemos que usar la función *dbinom*, que es la función de densidad y calcula la probabilidad P(X=k). Los parámetros son el valor del que queremos calcular la probabilidad, en este caso 1 carta dorada, el conjunto total, que en este caso son las 30 cartas del mazo, y la probabilidad de éxito, en este caso es la probabilidad relativa de que la carta sea dorada calculado en el ejercicio anterior, que es 0.034. El resultado es el siguiente:

```
> # Ejercicio 5:
>
> # apartado a:
>
> dbinom(1,size = 30, prob = 0.034)
[1] 0.3740562
```

La probabilidad de que haya una carta dorada es del 37.41%.

b) Haya, como mucho, tres cartas doradas.

La probabilidad de que haya como mucho 3 cartas doradas es la suma de las probabilidades de que no haya ninguna carta dorada, de que haya 1, 2 o 3. Por lo tanto, se podría hace uso de la función anterior con los valores 0, 1, 2, y 3 y el resultado sería correcto. Pero R ofrece otra función, *pbinom*, que es la función de distribución de la función de densidad F(X=k) tal que:

$$F(X=k) = P(X \leq k)$$

Por lo tanto, la función *pbinom* nos permita calcular la probabilidad de que haya 3 o menos cartas doradas:

```
> # apartado b:
>
> pbinom(3,size = 30, prob = 0.034)
[1] 0.9819209
```

La probabilidad de que haya 3 o menos cartas doradas es del 98,19%.

c) Tenga más cartas doradas que normales

Para que en el mazo de cartas haya más cartas doradas que no doradas, tiene que haber al menos 16 cartas doradas, puesto que en el caso de que hubiera 15 serían las mismas. Por lo tanto, la pregunta es el valor de $P(X \geq 16)$ y, aunque R no proporciona ninguna función para calcular ese valor directamente, sabemos que:

$$P(X \geq 16) = 1 - P(X \leq 15)$$

Por lo tanto, podemos aplicar la función pbinom para calcular la probabilidad de que hay más cartas doradas que normales:

```
> # apartado c:
>
> 1 - pbinom(15,size = 30, prob = 0.034)
[1] 2.220446e-16
```

La probabilidad de que haya más cartas doradas que normales es prácticamente nula.

PARTE 3

6. (Intermedio) Luka Doncic, jugador de baloncesto de los Dallas Mavericks, tiene un porcentaje de acierto en tiros de triple del 32%. Si en un partido concreto hace 12 intentos desde la línea de triples...

Este ejercicio es un claro ejemplo de una distribución binomial de forma que la probabilidad de éxito es 0.32 y se mantiene constante.

a) ... ¿cuál es la probabilidad de que acierte exactamente dos?

Para calcular la probabilidad de que acierte dos tiros, tenemos que usar la función *dbinom*, que es la función de densidad y calcula la probabilidad P(X=k), siendo k los tiros libres que acierta, es decir, 2, el número total de intentos es 12 y la probabilidad de acierto es 0.32. De esta forma, el resultado es:

```
> # Ejercicio 6
>
> # apartado a:
>
> dbinom(2,size = 12, prob = 0.32)
[1] 0.1428674
```

La probabilidad de que acierte exactamente dos tiros es de 14.29%.

b) ... ¿cuál es la probabilidad de que no los meta todos?

La probabilidad de que no meta todos los tiros libres es la suma de todas las probabilidades excepto P(X=12), por lo tanto, se puede calcular tal que 1 – P(X=12):

```
> # apartado b:
>
> 1-dbinom(12, size= 12, prob = 0.32)
[1] 0.9999988
```

La probabilidad de que no los meta todos es prácticamente del 100%.

c) ... ¿y la probabilidad de que enceste entre 4 y 8 (ambos inclusive)?

La probabilidad de que enceste entre 4 y 8 es $P(4 \leq X \leq 8)$, y para poder calcularlo con la función pbinom podemos hacerlo de la siguiente manera:

$$P(X \leq 8) - P(X \leq 3)$$

De tal forma que el resultado es el siguiente:

```
> # apartado c:
>
> pbinom(8, size = 12, prob = 0.32) - pbinom(3, size = 12, prob = 0.32)
[1] 0.5652454
```

La probabilidad de que enceste entre 4 y 8 tiros, ambos inclusive, es del 56.52%.

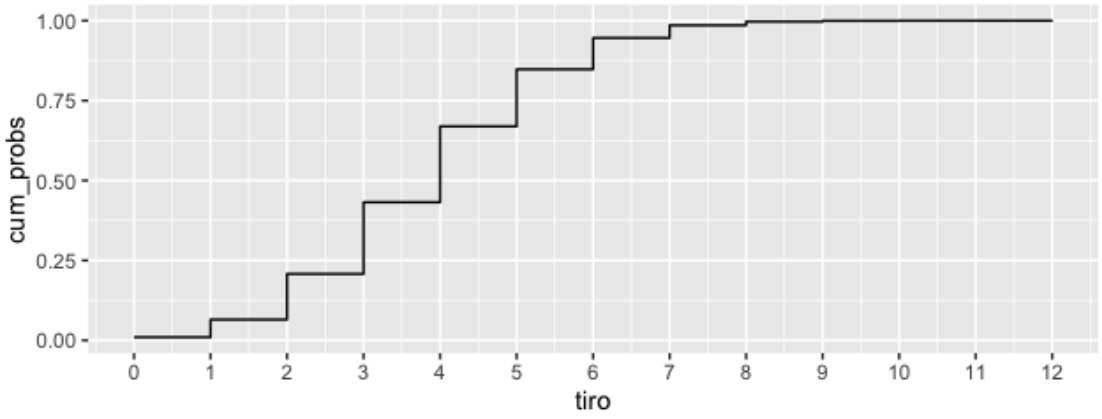
d) ... ¿cuál sería la mediana? (Es decir, ¿en qué número de aciertos dejamos el 50% de la probabilidad a cada lado?).

Para mostrar cual sería la mediana se puede representar en un gráfico la probabilidad acumulada de forma que se observe entre que número de aciertos se encuentra la mediana.

Para ello, almacenamos las probabilidades acumuladas en un vector, así como el acierto al que corresponde cada probabilidad acumulada y generamos un dataframe con la librería *tibble* para poder generar el gráfico deseado con la librería *ggplot2*.

```
232 # apartado d:
233
234 cum_probs <- pbinom(0:12, size = 12, prob = 0.32)
235 tiro <- c(0:12)
236
237 library(tibble)
238 cum_probs_df <- tibble(cum_probs, tiro)
239 ggplot(cum_probs_df, aes(x = tiro, y = cum_probs)) + geom_step() +
240   scale_x_continuous(breaks = seq(0, 12, by = 1))
```

El gráfico generado es el siguiente:



Se puede observar que el número de aciertos que deja el 50% a cada lado estaría entre los aciertos 3 y 4. Al ser una distribución discreta, solo se pueden acertar 3 o 4 tiros, por lo que no existe un valor exacto que deje el 50% a cada lado.

7. **(Avanzado) Cuando hago muffins (o magdalenas grandes) les pongo pepitas de chocolate. Voy a hacer 100 magdalenas y tengo 300 pepitas de chocolate que reparto de forma homogénea en toda la masa. Si cojo la masa para una magdalena:**

En este ejercicio se pide calcular la probabilidad de que aparezcan n pepitas de chocolate en una magdalena, por lo que interpreto que sigue una distribución de Poisson, cuya definición indica que se aplica al cálculo de la probabilidad de la aparición de n elementos de la población en un intervalo de duración, longitud, cantidad u otra magnitud fija. Por lo tanto, se asume que el proceso es estable, siendo el ratio de aparición medio constante e igual a 3 pepitas de chocolate por magdalena.

a) ... ¿cuál es la probabilidad de que no contenga pepitas de chocolate?

De formar similar a los ejercicios anteriores, R dispone de funciones para el cálculo de probabilidades para la distribución de Poisson. En este caso, se pide el cálculo de P (X=0), por lo que hay que usar la función *dpois* con los parámetros correspondientes, que son el número de pepitas, en este caso 0, y el valor de lambda, que es el ratio de aparición, que ya hemos comentado que es igual a 3:

```
> # Ejercicio 7:
>
> # apartado a:
>
> dpois(0, lambda = 3)
[1] 0.04978707
>
```

La probabilidad de que no contenga pepitas es del 4.98%.

b) ... ¿cuál es la probabilidad de que tenga más de 5 pepitas?

Para el cálculo de la probabilidad P(X>5), al igual que sucedía con la distribución binomial, R solo dispone de la función *ppois* que calcula probabilidades tal que P(X≤k), por lo que tenemos que modificar la probabilidad anterior de la siguiente forma:

$$P(X>5) = 1 - P(X\leq 5)$$

Como se nos pide que tenga más de 5 pepitas, hay que excluir también la probabilidad de que haya 5 pepitas:

```
> # apartado b:
>
> 1 - ppois(5, lambda = 3)
[1] 0.08391794
```

La probabilidad de que contenga más de 5 pepitas es del 8.39%.

c) ... ¿y la probabilidad de que tenga entre 2 y 4 (ambos inclusive)?

De nuevo, hemos de transformar la probabilidad solicitada en el enunciado para adaptarla al cálculo que nos ofrece la función *ppois* de la siguiente manera:

$$P(2 \leq X \leq 4) = P(X \leq 4) - P(X \leq 1)$$

```
> # apartado c:
>
> ppois(4, lambda = 3) - ppois(1, lambda = 3)
[1] 0.616115
```

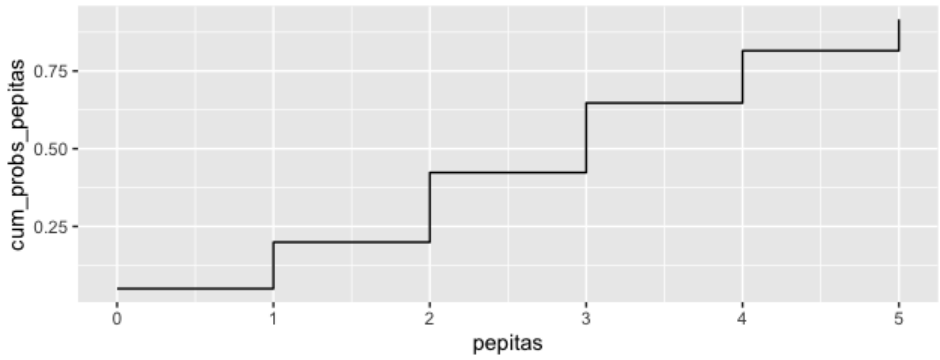
La probabilidad de que tenga entre 2 y 4 pepitas, ambas inclusive, es del 61.61%.

d) ... ¿cuál sería la mediana? (Es decir, ¿en qué número de pepitas dejamos el 50% de la probabilidad a cada lado?).

Siguiendo el mismo procedimiento que en el apartado d del ejercicio anterior, se va a representar en un gráfico la probabilidad acumulada de forma que se observe entre que número de pepitas se encuentra la mediana. Para ello, almacenamos las probabilidades acumuladas en un vector, así como el número de pepitas al que corresponde cada probabilidad acumulada y generamos un dataframe con la librería *tibble* para poder generar el gráfico deseado con la librería *ggplot2*.

```
256 # apartado d:
257
258 cum_probs_pepitas <- ppois(0:5, lambda = 3)
259 pepitas <- c(0:5)
260
261 cum_probs_pepitas_df <- tibble(cum_probs_pepitas, pepitas)
262 ggplot(cum_probs_pepitas_df, aes(x = pepitas, y = cum_probs_pepitas)) +
263   geom_step() + scale_x_continuous(breaks = seq(0, 5, by = 1))
```

El gráfico resultante es:



Se observa que el valor de probabilidad acumulada del 50% se sitúa entre las probabilidades de 2 y 3 pepitas. Al ser una distribución discreta, no hay un valor de pepitas que coincida exactamente con el valor del 50%, de forma que dejemos un 50% a cada lado.

PARTE 4

8. (Intermedio) El coeficiente intelectual de las personas sigue una distribución normal.

a) Si sabemos que lo hace con una media de 100 y una desviación estándar de 16:

Para el cálculo de este ejercicio, conocemos el tipo de distribución, y los valores de la media y de la desviación estándar, por lo que disponemos de los datos necesarios para el cálculo de probabilidades.

i. ¿Cuál es la probabilidad de encontrar a una persona con un coeficiente intelectual superior a 125?

Para este apartado, se pide la probabilidad P(X>125). Al igual que con las distribuciones binomial y Poisson, R dispone de funciones que nos facilitan el cálculo de probabilidades para la distribución normal, pero se han de realizar algunas modificaciones para calcular la probabilidad solicitada tal que:

$$P(X>125) = 1 - P(X<125)$$

Al igual que en casos anteriores, hemos de restar a 1 el valor de la probabilidad contraria para poder conseguir el resultado solicitado, pero en este caso, la distribución con la estamos trabajando es continua, y hay que tenerlo en cuenta a la hora del cálculo de probabilidades:

```
> # Ejercicio 8:
>
> # apartado a1:
>
> 1 - pnorm(125, mean = 100, sd = 16)
[1] 0.05908512
```

La probabilidad de encontrar a una persona con un coeficiente intelectual superior a 125 es de 5.91%.

ii. ¿Y que tenga el coeficiente intelectual entre 90 y 120?

El hecho de que estemos trabajando con una distribución continua simplifica las modificaciones necesarias para calcular P(90<X<120), de forma que se calcula directamente:

$$P(X<120) - P(X<90)$$

```
> # apartado a2:
>
> pnorm(120, mean = 100, sd = 16) - pnorm(90, mean = 100, sd = 16)
[1] 0.6283647
```

Por lo tanto, la probabilidad de que una persona tenga un coeficiente intelectual entre 90 y 120 es del 62.84%.

b) Hemos cogido a 50 alumnos que han realizado el test de IQ y su media es de 96.

En este ejercicio, los datos que conocemos son pertenecientes a una muestra y en estos casos, las conclusiones se infieren a partir de la muestra junto con el conocimiento de la distribución, pudiendo darse únicamente con cierta probabilidad.

i. Calcula el intervalo de confianza al 95% para la media poblacional del IQ de los alumnos. Puedes suponer que la desviación estándar es la misma que en el caso (a).

En primer lugar, para muestras suficientemente grandes, las medias muestrales de todas las variables se comportan como variables muestrales, pero, si además hemos empezado con una variable normal, entonces el tamaño de la muestra es irrelevante. En este caso, sabemos que la distribución de la que hemos extraído la muestra es una distribución Normal, por lo que la media muestral también lo es.

Por lo tanto, conocemos que la población de partida sigue una distribución normal y que la desviación estándar es la misma en la muestra que en la población de partida, por lo que podemos calcular el intervalo de confianza al 95% con la siguiente expresión:

$$\overline{x} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \leq \mu_x \leq \overline{x} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$$

La fórmula anterior se puede aplicar en R de la siguiente manera:

```
275 # apartado b1:
276 
277 nc_8 <- 0.95
278 alfa_8 <- 1-nc_8
279 alfa_medios_8 <- alfa_8/2
280 z_alfa2_8 <- qnorm(1-alfa_medios_8)
281 
282 media_muestral_8 <- 96
283 n_8 <- 50
284 s_8 <- 16
285 
286 int_izq_8 <- media_muestral_8 - z_alfa2_8*(s_8/sqrt(n_8))
287 int_dch_8 <- media_muestral_8 + z_alfa2_8*(s_8/sqrt(n_8))
288 
289 int_izq_8
290 int_dch_8
```

Con ellos, calculamos los intervalos izquierdo y derecho del intervalo, cuyos resultados son:

```
> int_izq_8
[1] 91.56511
> int_dch_8
[1] 100.4349
```

Y el intervalo queda tal que:

$$91.57 \leq \mu_x \leq 100.43$$

ii. ¿Podríamos decir que, con el 95% de confianza, la media de los alumnos es la misma que la general descrita en (a)?

En este apartado, nos preguntamos la hipótesis de si la media de la muestra es igual a la de la población al 95%. Para ello, necesitamos una hipótesis nula, que es la que contrastamos, y una hipótesis alternativa.

Para aceptar o rechazar una hipótesis, se ha de estudiar la distribución de probabilidad de nuestras observaciones, a lo que se le llama contraste. Dicho esto, vamos a definir el problema:

Hipótesis nula: $\mu = 100$
Hipótesis alternativa: $\mu < 100$

Como la media muestral es 96, nos centramos solo en una dirección de forma que el test es más poderoso.

De esta forma, la región de aceptación es:

$$-z_\alpha < z_0 < z_\alpha$$

Y el valor de z_0 es el siguiente:

$$z_0 = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

Podemos calcular los valores a través de R con el siguiente código:

```r
# apartado b2:

nc_8 <- 0.95
alfa_8 <- 1-nc_8
z_alfa_8 <- qnorm(1-alfa_8)

media_muestral_8 <- 96
media_pob_8 <- 100
n_8 <- 50
s_8 <- 16

z_cero_8 <- (media_muestral_8-media_pob_8)/s_8/sqrt(n_8)

-z_alfa_8
z_cero_8
z_alfa_8
```

Cuyos resultados son:

```
> -z_alfa_8
[1] -1.644854
> z_cero_8
[1] -0.03535534
> z_alfa_8
[1] 1.644854
```

Por lo tanto, el intervalo queda:

$$-1.64 < -0.04 < 1.64$$

Podemos concluir que no descartaríamos la hipótesis nula, y que la media de la muestra es la misma que la media poblacional a un 95% de confianza.

9. (Intermedio) El peso de un niño al nacer sigue una distribución normal.

a) Si sabemos que lo hace con una media de 2950 g y una desviación estándar de 300:

Para el cálculo de este ejercicio, conocemos el tipo de distribución, y los valores de la media y de la desviación estándar, por lo que disponemos de los datos necesarios para el cálculo de probabilidades.

i. ¿Cuál es la probabilidad de que nazca un niño con un peso menor a 3100 g?

Para este apartado, se pide la probabilidad P(X<3100). Por lo tanto, podemos aplicar la función *pnorm* directamente tal que:

```
> # Ejercicio 9:
>
> # apartado a1:
>
> pnorm(3100, mean = 2950, sd = 300)
[1] 0.6914625
```

Se observa que la probabilidad de que nazca un niño con un peso menor a 3100 gramos es del 69.15%.

ii. ¿Y que pese menos de 2750g o más de 3250g?

En este caso, se pide la probabilidad de dos situaciones, por lo que lo podemos calcularlo tal que:

$$1 - P(2750<X<3250) = 1 - P(X<3250) + P(X<2750)$$

```
> # apartado a2:
>
> 1 - pnorm(3250, mean = 2950, sd = 300) + pnorm(2750, mean = 2950, sd = 300)
[1] 0.4111478
```

Se observa que la probabilidad de que nazca un niño con un peso inferior a 2750 gramos o superior a 3250 es de 41.11%

b) Hemos cogido a 40 bebés varones que han nacido esta semana en Madrid y su media es de 3000.

De forma similar al apartado b del ejercicio 8, disponemos de los datos pertenecientes a una muestra y en estos casos, las conclusiones se infieren a partir de la muestra junto con el conocimiento de la distribución, pudiendo darse únicamente con cierta probabilidad.

i. **Calcula el intervalo de confianza al 90% para la media poblacional del peso de los bebés. Puedes suponer que la desviación estándar es la misma que en el caso (a).**

Conocemos que la población de partida es normal y que la desviación estándar es la misma en la muestra que en la población de partida, por lo que podemos calcular el intervalo de confianza al 90% con la siguiente expresión:

$$\overline{x} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \leq \mu_x \leq \overline{x} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$$

La fórmula anterior se puede aplicar en R de la siguiente manera:

```
319 # apartado b1:
320
321 nc_9 <- 0.90
322 alfa_9 <- 1-nc_9
323 alfa_medios_9 <- alfa_9/2
324 z_alfa2_9 <- qnorm(1-alfa_medios_9)
325
326 media_muestral_9 <- 3000
327 n_9 <- 40
328 s_9 <- 300
329
330 int_izq_9 <- media_muestral_9 - z_alfa2_9*(s_9/sqrt(n_9))
331 int_dch_9 <- media_muestral_9 + z_alfa2_9*(s_9/sqrt(n_9))
332
333 int_izq_9
334 int_dch_9
```

Con ellos, calculamos los intervalos izquierdo y derecho del intervalo, cuyos resultados son:

```
> int_izq_9
[1] 2921.978
> int_dch_9
[1] 3078.022
```

Por lo tanto, el intervalo queda tal que:

$$2921.98 \leq \mu_x \leq 3078.02$$

ii. ¿Podríamos decir que, con el 90% de confianza, los bebés nacidos en Madrid son más gorditos que la media habitual?

En este apartado, nos preguntamos la hipótesis de si la media de la muestra es superior a la media de la población. Para ello, necesitamos una hipótesis nula, que es la que contrastamos, y una hipótesis alternativa. Dicho esto, podemos definir el problema:

Hipótesis nula: $\mu = 2950$
Hipótesis alternativa: $\mu > 2950$

Como la media muestral es 3000, nos centramos solo en una dirección de forma que el test es más poderoso. De esta forma, la región de aceptación es:

$$-z_\alpha < z_0 < z_\alpha$$

Y el valor de z_0 es el siguiente:

$$z_0 = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

Podemos calcular los valores a través de R con el siguiente código:

```
336 # apartado b2:
337 
338 nc_9 <- 0.90
339 alfa_9 <- 1-nc_9
340 z_alfa_9 <- qnorm(1-alfa_9)
341 
342 media_muestral_9 <- 3000
343 media_pob_9 <- 2950
344 n_9 <- 40
345 s_9 <- 300
346 
347 z_cero_9 <- (media_muestral_9-media_pob_9)/s_9/sqrt(n_9)
348 
349 -z_alfa_9
350 z_cero_9
351 z_alfa_9
```

Cuyos resultados son:

```
> -z_alfa_9
[1] -1.281552
> z_cero_9
[1] 0.02635231
> z_alfa_9
[1] 1.281552
```

Por lo tanto, el intervalo queda:

$$-1.28 < -0.03 < 1.28$$

Podemos concluir que no descartaríamos la hipótesis nula, y que la media de la muestra es la misma que la media poblacional a un 90% de confianza, por lo que no podemos concluir que los bebes nacidos en Madrid sean más gorditos.

10. (Básico) Las notas de estadística siguen una distribución normal.

a) Si sabemos que lo hacen con una media de 5.8 y una desviación estándar de 1.7:

Para el cálculo de este ejercicio, conocemos el tipo de distribución, y los valores de la media y de la desviación estándar, por lo que disponemos de los datos necesarios para el cálculo de probabilidades.

i. ¿Cuál es la probabilidad de encontrar a un alumno con una nota menor a 5?

Para este apartado, se pide la probabilidad P(X<5). Por lo tanto, podemos aplicar la función *pnorm* directamente tal que:

```
> # Ejercicio 10:
>
> # apartado a1:
>
> pnorm(5, mean = 5.8, sd = 1.7)
[1] 0.3189674
```

Se observa que la probabilidad de encontrar a un alumno con una nota menor a 5 es 31.9%.

ii. ¿Y que haya sacado entre 5 y 8?

El hecho de que estemos trabajando con una distribución continua simplifica las modificaciones necesarias para calcular P(5<X<8), de forma que se calcula directamente:

$$P(X<8) - P(X<5)$$

```
> # apartado a2:
>
> pnorm(8, mean = 5.8, sd = 1.7) - pnorm(5, mean = 5.8, sd = 1.7)
[1] 0.5832202
```

La probabilidad de encontrar a un alumno con una nota entre 5 y 8 es del 58.32%.

b) 35 alumnos de esta clase han sacado un 6 de media en la asignatura.

De forma similar al apartado b de los ejercicios 8 y 9, disponemos de los datos pertenecientes a una muestra y en estos casos, las conclusiones se infieren a partir de la muestra junto con el conocimiento de la distribución, pudiendo darse únicamente con cierta probabilidad.

i. **Calcula el intervalo de confianza al 95% para la media poblacional de los alumnos de estadística de este curso. Puedes suponer que la desviación estándar es la misma que en el caso (a).**

Conocemos que la población de partida es normal y que la desviación estándar es la misma en la muestra que en la población de partida, por lo que podemos calcular el intervalo de confianza al 90% con la siguiente expresión:

$$\overline{X} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \leq \mu_x \leq \overline{X} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$$

La fórmula anterior se puede aplicar en R de la siguiente manera:

```
363 # apartado b:
364
365 nc_10 <- 0.95
366 alfa_10 <- 1-nc_10
367 alfa_medios_10 <- alfa_10/2
368 z_alfa2_10 <- qnorm(1-alfa_medios_10)
369
370 media_muestral_10 <- 6
371 n_10 <- 35
372 s_10 <- 1.7
373
374 int_izq_10 <- media_muestral_10 - z_alfa2_10*(s_10/sqrt(n_10))
375 int_dch_10 <- media_muestral_10 + z_alfa2_10*(s_10/sqrt(n_10))
376
377 int_izq_10
378 int_dch_10
```

Con ellos, calculamos los intervalos izquierdo y derecho del intervalo, cuyos resultados son:

```
> int_izq_10
[1] 5.4368
> int_dch_10
[1] 6.5632
```

Por lo tanto, el intervalo queda tal que:

$$5.44 \leq \mu_x \leq 6.56$$