Lección 5

Bondad de ajuste

Una de las condiciones habituales que requerimos sobre una muestra, por ejemplo, al razonar sobre la distribución de sus estadísticos o al realizar contrastes de hipótesis, es que la población de la que la hemos extraído siga una determinada distribución. En la Lección 20 del primer volumen comprobábamos gráficamente el ajuste de una muestra a una distribución normal mediante histogramas y dibujando las curvas de densidad muestral y de densidad de la normal. En esta lección presentamos algunas instrucciones que implementan *tests de bondad de ajuste*,[1] técnicas cuantitativas que permiten contrastar si los datos de una muestra provienen de una determinada distribución de probabilidad.

Los tests de bondad de ajuste tienen el mismo significado que los contrastes de parámetros estadísticos estudiados en lecciones anteriores. Se contrasta una hipótesis nula

H_0: La muestra proviene de una población con distribución X

contra la hipótesis alternativa

H_1: La muestra *no* proviene de una población con distribución X.

Si el resultado del test permite rechazar la hipótesis nula, es porque hemos obtenido evidencia significativa de que la muestra no proviene de una población con distribución X. Pero que el test no permita rechazar la hipótesis nula no nos da evidencia de que la muestra sí que provenga de una población con distribución X: simplemente nos dice que no lo podemos rechazar. Naturalmente, a efectos prácticos, estamos dispuestos a aceptar «por defecto» la hipótesis nula si no la podemos rechazar, pero en el mejor de los casos se trata de un abuso de lenguaje.

[1] En inglés se utiliza el acrónimo *GOF*, de *goodness of fit*.

5.1. Bondad de ajuste

Los pasos habituales para contrastar la bondad del ajuste de una muestra a una distribución son los siguientes:

- Fijar la familia de distribuciones teóricas a la que queremos ajustar los datos. Esta familia estará parametrizada por uno o varios parámetros. Por ejemplo:
  - Si la familia es la Bernoulli, el parámetro es p: la probabilidad de éxito.
  - Si la familia es la Poisson, el parámetro es λ: la esperanza.
  - Si la familia es la binomial, los parámetros son n y p: el tamaño de las muestras y la probabilidad de éxito, respectivamente.
  - Si la familia es la normal, los parámetros son μ y σ: la esperanza y la desviación típica, respectivamente.
  - Si la familia es la exponencial, el parámetro es λ: el inverso de la esperanza.
  - Si la familia es la χ^2, el parámetro es n: el número de grados de libertad.
  - Si la familia es la t de Student, el parámetro es de nuevo n: el número de grados de libertad.
  - Otras familias de distribuciones tienen parámetros de localización (*location*), escala (*scale*) o forma (*shape*), por lo que no nos ha de extrañar si `R` nos pide que asignemos parámetros con estos nombres.
- Si el diseño del experimento no fija los valores de estos parámetros, tendremos que estimar a partir de la muestra los valores de los parámetros que mejor se ajusten a nuestros datos.
- Determinar qué tipo de contraste vamos a utilizar. En esta lección veremos dos tipos básicos de contrastes generales; concretamente:
  - El *test χ^2 de Pearson*. Este test es válido tanto para variables discretas como para continuas, pero sólo se puede aplicar a conjuntos grandes de datos (por fijar una cota concreta, de 30 o más elementos), y, si el *espacio muestral* (el conjunto de resultados posibles) es infinito, es necesario agrupar estos resultados en un número finito de clases.
  - El *test de Kolgomorov-Smirnov*. Este test sólo es válido para variables continuas, y compara la función de distribución acumulada muestral con la teórica. No requiere que la muestra sea grande, pero en cambio, en principio, no admite que los datos de la muestra se puedan repetir.[2] Por desgracia, las repeticiones suelen ser habituales si la precisión de los datos es baja y la variabilidad de la población muestreada es pequeña o la muestra es grande.

[2] A las repeticiones se las suele llamar empates, *ties* en inglés, porque la función de distribución acumulada muestral ordena los datos y las repeticiones producen empates en observaciones de valores sucesivos.

Aparte, determinadas familias de distribuciones tienen sus contrastes de bondad de ajustes específicos. Este es el caso especialmente de las distribuciones normales, para las que presentaremos algunos tests que permiten contrastar si una muestra proviene de *alguna* distribución normal.

- Realizar el contraste y redactar las conclusiones. Es conveniente apoyar los resultados del contraste con gráficos. En esta lección explicaremos los *gráficos cuantil-cuantil*, que sirven para visualizar el ajuste de unos datos a una distribución conocida y son una buena alternativa a los histogramas con curvas de densidad.

5.2. Estimación de parámetros

La estimación de los parámetros de una familia de distribuciones depende en gran medida de la propia familia. Algunas distribuciones disponen de estimadores sencillos e intuitivos. Por ejemplo, en las distribuciones Bernoulli, Poisson, exponenciales y normales, la media muestral es el estimador más adecuado de la media poblacional. En otras ocasiones se puede usar el método de máxima verosimilitud de estimación de un parámetro, que nos da el valor que maximiza la probabilidad de haber obtenido la muestra. Para algunas distribuciones este método da lugar a una fórmula cerrada, pero en otros casos nos tenemos que conformar con un valor aproximado obtenido mediante algún método numérico.

A continuación recordamos una lista de los estimadores máximo verosímiles de los parámetros de las distribuciones más comunes a partir de una muestra:

- Para la familia Bernoulli, el estimador máximo verosímil del parámetro p es la media muestral $\overline{X}$, que coincide con la proporción muestral de éxitos.
- Para la familia geométrica, el estimador máximo verosímil del parámetro p es $1/\overline{X}$.
- Para la familia Poisson, el estimador máximo verosímil del parámetro λ es $\overline{X}$.
- Para la familia exponencial, el estimador máximo verosímil del parámetro λ es $1/\overline{X}$.
- Para la familia normal, los estimadores máximo verosímiles de los parámetros μ y σ son, respectivamente, la media muestral $\overline{X}$ y la desviación típica «verdadera» S. Ni S ni $\widetilde{S}$ son estimadores insesgados de σ, aunque la varianza muestral $\widetilde{S}^2$ sí que es un estimador insesgado de la varianza poblacional σ^2.[3]

Cuando se estima algún parámetro de una distribución a partir de una muestra, es conveniente aportar el *error estándar* como medida de la finura de la estimación. Recordemos que el *error estándar de un estimador* es su desviación típica, y el *error estándar de una estimación* a partir de una muestra es la estimación resultante de dicha desviación típica

[3] Si necesitáis un estimador insesgado de la σ de una variable aleatoria normal a partir de una muestra aleatoria simple, lo podéis encontrar en `http://en.wikipedia.org/wiki/Unbiased_estimation_of_standard_deviation`. No obstante, el beneficio de usar este estimador insesgado no suele compensar lo complicado de su cálculo.

para esta muestra. Veamos un ejemplo sencillo. Supongamos que tenemos una muestra aleatoria simple de tamaño n de una variable X que sigue una distribución Bernoulli de parámetro p desconocido. Tomemos, para fijar ideas, la siguiente muestra de tamaño 100:

> 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
> 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
> 0, 0, 0, 0, 0, 1, 0.

En este caso, podemos estimar p mediante la media muestral, que coincide con la proporción muestral $\widehat{p}$ de éxitos. El error estándar de este estimador es $\sqrt{p(1-p)/n}$, y el error estándar de una estimación concreta es $\sqrt{\widehat{p}(1-\widehat{p})/n}$, que coincide con $s/\sqrt{n}$, donde s denota la desviación típica de la muestra.

Por lo tanto, a mano podemos estimar p y calcular el error estándar de dicha estimación de la manera siguiente:

```
> x=c(0,1,1,1,0,0,1,1,1,1,0,0,1,1,0,0,1,1,0,0,0,0,0,1,0,0,0,0,
   1,0,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,
   1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,
   0,0,1,0,0,0,0,0,0,0,1,0)
> n=length(x)
> n
[1] 100
> estim.p=mean(x)
> estim.p
[1] 0.26
> error.estan.p=sqrt(estim.p*(1-estim.p)/n)
> error.estan.p
[1] 0.04386342
> (sqrt((n-1)/n)*sd(x))/sqrt(n)
[1] 0.04386342
```

De esta manera, estimamos que $p = 0.26$ con un error estándar de 0.044.

Con `R` podemos estimar un parámetro de una distribución a partir de una muestra y además obtener el error estándar de dicha estimación por medio de la función `fitdistr` del paquete `MASS`. Esta función calcula los estimadores máximo verosímiles de los parámetros de la mayoría de las familias de distribuciones disponibles en `R`. Su sintaxis básica es

```
fitdistr(x, densfun="distribución", start=...)
```

donde

- `x` es la muestra que usamos para estimar los parámetros.
- `distribución` es el nombre de la familia de distribuciones; se tiene que entrar entre comillas y puede tomar, entre otros, los valores siguientes: `"chi-squared"`,

`"exponential"`, `"f"`, `"geometric"`, `"lognormal"`, `"normal"` y `"poisson"`. La lista de posibles valores no incluye la Bernoulli ni la binomial.

- Si `fitdistr` no dispone de fórmulas cerradas para los estimadores máximo verosímiles de los parámetros, usa un algoritmo numérico para aproximarlos que requiere de un valor inicial para arrancar. Entonces, en `start` se ha de especificar una `list` con cada parámetro a estimar igualado a un valor inicial.

  Algunas distribuciones, como la `"normal"` o la `"poisson"`, disponen de fórmulas para calcular los estimadores máximo verosímiles; en estos casos no se ha de especificar el parámetro `start`. Para otras distribuciones, como la `"t"`, `fitdistr` sabe tomar valores iniciales razonables, y no es necesario especificar el parámetro `start`. Pero para otras distribuciones, como por ejemplo la `"chisq"`, es obligatorio especificarlo.

Para más información, podéis consultar `help(fitdistr)`.

Como no podemos usar `fitdistr` para estimar el parámetro p de una Bernoulli (los autores del paquete debieron de considerar que era muy fácil estimarlo directamente), vamos a usarla en otro ejemplo. Consideremos la siguiente muestra de 100 datos (que en realidad ha sido generada con una distribución de Poisson de parámetro $\lambda = 10$ mediante `set.seed(100); rpois(100,10)`):

8, 10, 9, 12, 10, 11, 8, 12, 7, 11, 11, 12, 7, 10, 9, 8, 11, 7, 6, 12, 10, 12, 7, 7, 10, 6, 7, 15, 9, 9, 7, 8, 15, 11, 12, 5, 14, 4, 7, 14, 8, 14, 10, 4, 9, 8, 11, 11, 10, 12, 7, 14, 7, 9, 10, 3, 10, 7, 9, 21, 14, 6, 13, 3, 10, 6, 3, 13, 9, 12, 8, 11, 10, 11, 11, 8, 6, 17, 7, 8, 10, 12, 15, 12, 13, 10, 9, 12, 8, 11, 12, 4, 10, 8, 5, 8, 8, 10, 8, 11.

Vamos ahora a estimar el parámetro λ de una distribución Poisson que haya generado esta lista:

```
> #Instalamos y cargamos el paquete "MASS"
...
> y=c(8,10,9,12,10,11,8,12,7,11,11,12,7,10,9,8,11,7,6,12,
   10,12,7,7,10,6,7,15,9,9,7,8,15,11,12,5,14,4,7,14,8,14,10,
   4,9,8,11,11,10,12,7,14,7,9,10,3,10,7,9,21,14,6,13,3,10,6,
   3,13,9,12,8,11,10,11,11,8,6,17,7,8,10,12,15,12,13,10,9,
   12,8,11,12,4,10,8,5,8,8,10,8,11)
> fitdistr(y, densfun="poisson")
    lambda
  9.5600000
 (0.3091925)
```

El resultado dice que el valor estimado de λ es 9.56, con un error estándar en esta estimación de aproximadamente 0.31. Veámoslo directamente: el estimador de λ es la media aritmética $\overline{X}$ y el error estándar de esta estimación es $\sqrt{\overline{X}}/\sqrt{n}$ (recordad que la desviación típica de una Poisson de parámetro λ es $\sqrt{\lambda}$).

```
> mean(y)
[1] 9.56
> sqrt(mean(y)/length(y))
[1] 0.3091925
```

También podemos estimar la media y la desviación típica de una variable normal que hubiera producido esta muestra.

```
> fitdistr(y, densfun="normal")
     mean          sd
  9.5600000   3.0832450
 (0.3083245) (0.2180183)
```

Observad que la estimación de la desviación típica que nos da `fitdistr` es la desviación típica «verdadera» y no la muestral:

```
> sd(y)
[1] 3.098778
> sqrt((length(y)-1)/length(y))*sd(y)
[1] 3.083245
```

Vamos a estimar ahora el número de grados de libertad de una t de Student que hubiera producido esta muestra.

```
> fitdistr(y, densfun="t")
       m           s           df
  9.5085516   2.7517475   9.9229027
 (0.2997949) (0.3072053) (8.4890355)
```

¡Vaya, han aparecido parámetros que no esperábamos! Los parámetros `m` y `s` son los parámetros locacional, μ, y de escala, σ, respectivamente, que definen una familia más general de distribuciones:[4] las t de Student que usamos en este curso tienen $\mu = 0$ y $\sigma = 1$. ¿Cómo podríamos estimar los grados de libertad de una t de Student de las nuestras? Especificando dentro de `fitdistr` los valores de los parámetros que queremos que tomen un valor concreto: en este caso, añadiendo `m=0` y `s=1`.

```
> fitdistr(y, densfun="t", m=0, s=1)
Error in fitdistr(y, densfun = "t", m = 0, s = 1) :
  'start' must be a named list
```

`R` nos pide que demos un valor inicial al número de grados de libertad, `df`, para poder arrancar el algoritmo numérico que usará. Vamos a inicializarlo a 1, y de paso veremos cómo se usa este parámetro:

```
> fitdistr(y, densfun="t",m=0,s=1,start=list(df=1))
```

[4] `http://en.wikipedia.org/wiki/Noncentral_t-distribution`

```
       df
  0.37265625
 (0.04277075)

Warning messages:
1: In stats::optim(x = c(8, 10, 9, 12, 10, 11, 8, 12, 7, 11, 11,
    12,  :
  one-dimensional optimization by Nelder-Mead is unreliable:
use "Brent" or optimize() directly
2: In dt((x - m)/s, df, log = TRUE) : NaNs produced
```

Obtenemos un número estimado de grados de libertad de la t de Student de aproximadamente 0. Por otro lado, `R` nos avisa de que el resultado es poco de fiar, pero tampoco nos importa mucho, porque el objetivo era mostrar un ejemplo de cómo especificar valores de parámetros, igualándolos a dichos valores, y el `start`, como una `list` donde asignamos a cada parámetro un valor inicial.

El resultado de `fitdistr` es una `list`, y por lo tanto el valor de cada estimador y su error estándar se pueden obtener con los sufijos adecuados. En concreto, los valores estimados forman la componente `estimate` y los errores estándar la componente `sd`. Para obtenerlos directamente, basta añadir los sufijos `$estimate` y `$sd`, respectivamente:

```
> fitdistr(y,"poisson")$estimate
lambda
  9.56
> fitdistr(y,"poisson")$sd
   lambda
0.3091925
> fitdistr(y,"normal")$estimate
    mean       sd
9.560000 3.083245
> fitdistr(y,"normal")$estimate[1]
mean
9.56
> fitdistr(y,"normal")$estimate[2]
      sd
3.083245
```

5.3. Pruebas gráficas: QQ-plots

Para comparar la distribución muestral con la teórica se pueden realizar diversas pruebas gráficas. En la Lección 20 del primer volumen usábamos histogramas con densidades muestrales y teóricas. En esta sección explicamos otro tipo de gráficos con el mismo fin, los *gráficos cuantil-cuantil*, o, para abreviar, *QQ-plots*. Este tipo de gráficos compara en un gráfico los cuantiles observados de la muestra con los cuantiles teóricos de la distribución con la que se quiere comparar. Hay varias maneras de generar este gráfico con `R`. Aquí sólo

explicaremos dos: la función básica `qqplot`, y la función `qqPlot` del paquete `car`.

La Figura 5.1 muestra un QQ-plot. Cada punto corresponde a un cuantil: en concreto, hay un punto para cada cuantil de la forma k/n, donde n es la longitud de la muestra y $k = 1, \ldots, n$. Para cada uno de estos cuantiles, el punto correspondiente tiene abscisa el valor del cuantil en la muestra, y ordenada el valor del cuantil para la distribución teórica (en este caso, una t de Student con 4 grados de libertad) sobre una muestra del mismo tamaño. Por lo tanto, si el ajuste es bueno, para cada k/n, el cuantil muestral y el cuantil teórico han de ser aproximadamente iguales, de manera que los puntos del gráfico (les llamaremos *QQ-puntos*, para abreviar) han de estar aproximadamente sobre la diagonal $y = x$. En general, se considera que un QQ-plot muestra un buen ajuste cuando no se observa una tendencia marcada de desviación respecto de la diagonal. Sin embargo, a menudo los QQ-plots son difíciles de interpretar, y es conveniente combinarlos con algún contraste de bondad de ajuste.

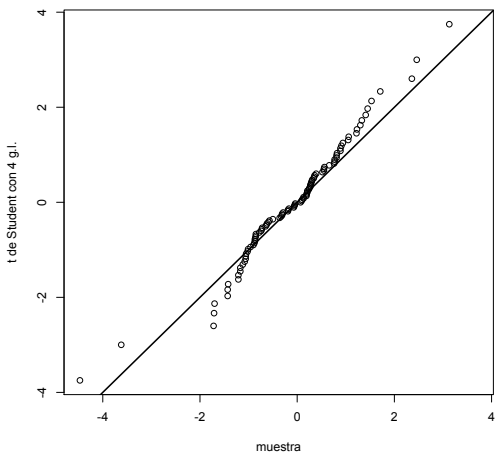


Figura 5.1. QQ-plot de la muestra del Ejemplo 5.1 contra una t de Student con 4 grados de libertad.

La función básica de `R` para producir QQ-plots es `qqplot`. Su sintaxis básica es

```
qqplot(x, y,  ...)
```

donde

- `x` e `y` son dos muestras cuyos cuantiles queremos comparar. Si lo que queremos es comparar una muestra contra una distribución teórica, entramos como `x` la muestra y, como `y`, la lista de los cuantiles k/n de la distribución teórica, para k desde 1 hasta la longitud n de la muestra.
- Se pueden usar los parámetros usuales en `plot` para poner nombres a los ejes, título,

modificar el estilo de los puntos, etc.

Como nos interesa comparar los QQ-puntos con la diagonal $y = x$, es conveniente añadir esta última con `abline(0,1)`.

Ejemplo 5.1. Tenemos la siguiente muestra

0.27,	0.81,	−0.73,	−0.96,	1.33,	0.91,	−1.70,	0.24,	−0.19,	0.29,
1.41,	0.13,	−0.06,	−0.85,	−0.59,	−3.62,	−1.02,	2.36,	0.34,	−0.31,
0.81,	−0.88,	0.27,	0.52,	1.05,	0.20,	0.76,	0.25,	−1.43,	3.71,
−0.78,	0.39,	−1.01,	1.53,	−0.72,	1.22,	0.56,	−1.17,	−0.65,	−0.33,
−0.07,	0.31,	−0.74,	0.36,	−1.72,	−1.21,	−0.05,	−1.17,	0.28,	1.30,
0.89,	1.45,	0.13,	−1.12,	3.13,	−1.21,	−0.90,	−0.31,	−1.05,	0.89,
−1.06,	0.21,	−0.50,	−0.36,	−0.29,	−0.19,	−1.71,	0.09,	0.21,	0.55,
−1.42,	0.19,	−0.62,	2.46,	−0.17,	−0.63,	0.77,	0.94,	0.55,	0.35,
−4.47,	1.71,	0.07,	−0.57,	−1.43,	−0.85,	1.06,	0.82,	0.19,	−1.08,
0.30,	−0.87,	0.77,	1.23,	−0.04,	0.66,	−0.87,	−0.86,	−1.06,	0.10

y queremos comprobar gráficamente si sigue una distribución t de Student de 4 grados de libertad.

```
> muestra=c(0.27,0.81,-0.73,-0.96,1.33,0.91,-1.70,0.24,-0.19,
  0.29,1.41,0.13,-0.06,-0.85,-0.59,-3.62,-1.02,2.36,0.34,-0.31,
  0.81,-0.88,0.27,0.52,1.05,0.20,0.76,0.25,-1.43,3.71,-0.78,
  0.39,-1.01,1.53,-0.72,1.22,0.56,-1.17,-0.65,-0.33,-0.07,0.31,
  -0.74,0.36,-1.72,-1.21,-0.05,-1.17,0.28,1.30,0.89,1.45,0.13,
  -1.12,3.13,-1.21,-0.90,-0.31,-1.05,0.89,-1.06,0.21,-0.50,
  -0.36,-0.29,-0.19,-1.71,0.09,0.21,0.55,-1.42,0.19,-0.62,2.46,
  -0.17,-0.63,0.77,0.94,0.55,0.35,-4.47,1.71,0.07,-0.57,-1.43,
  -0.85,1.06,0.82,0.19,-1.08,0.30,-0.87,0.77,1.23,-0.04,0.66,
  -0.87,-0.86,-1.06,0.10)
> x.teor=qt((1:length(muestra))/length(muestra),4) #Cuantiles de
    la t
> qqplot(muestra, x.teor, ylab="t de Student con 4 g.l.")
> abline(0,1,lwd=2)
```

Obtenemos el gráfico de la Figura 5.1, donde vemos que los puntos centrales se ajustan bien a la diagonal, pero en los dos extremos los puntos se separan de la misma.

El paquete `car` lleva la función `qqPlot` (fijaos en la mayúscula) que produce QQ-plots de muestras contra distribuciones teóricas más informativos. Su sintaxis básica es

```
qqPlot(x, distribution="distribución",  parámetros, ...)
```

donde

- `x` es la muestra.

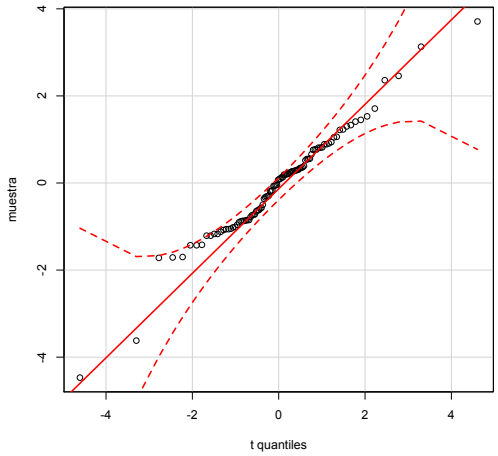


Figura 5.2. QQ-plot de la muestra del Ejemplo 5.1 contra una t de Student con 4 grados de libertad usando `qqPlot`.

- `distribución` es el nombre de la familia de distribuciones; se ha de entrar entre comillas, y puede tomar como valor cualquier nombre de familia de distribuciones de la que `R` sepa calcular la densidad y los cuantiles: esto incluye las distribuciones que hemos estudiado hasta el momento: `"norm"`, `"binom"`, `"poisson"`, `"t"`, etc.
- A continuación, se tienen que entrar los parámetros de la distribución, igualando su nombre habitual (`mean` para la media, `sd` para la desviación típica, `df` para los grados de libertad, etc.) a su valor. En algunos casos, si no se especifican los parámetros, `qqPlot` toma sus valores por defecto: por ejemplo, si queremos realizar un QQ-plot contra una normal y no especificamos los valores de la media y la desviación típica de la distribución teórica, `qqPlot` los toma iguales a 0 y 1, respectivamente.
- Se pueden usar los parámetros usuales de `plot` para poner nombres a los ejes, título, modificar el estilo de los puntos, etc., y otros parámetros específicos para modificar el aspecto del gráfico. Consultad el `help` de la función. En particular, `col.lines` sirve para especificar el color de las líneas que, como veremos, añade y que por defecto son rojas.

Ejemplo 5.2. Continuemos con el ejemplo anterior. Vamos a usar `qqPlot` para producir el QQ-plot deseado. El código

```
> #Instalamos y cargamos el paquete "car"
...
> qqPlot(muestra, distribution="t", df=4)
```

produce el gráfico de la Figura 5.2. Observad de entrada (por los nombres de los ejes) que ha intercambiado los ejes de abscisas y de ordenadas respecto del QQ-plot anterior: aquí, las abscisas de los QQ-puntos representan los cuantiles de la distribución teórica y sus ordenadas los de la muestra. Los puntos tampoco son exactamente los mismos, ya que usa una versión diferente de los cuantiles. Pero lo más evidente es que ha añadido una línea recta y dos curvas discontinuas:

- La línea recta, que sustituye a la diagonal, es la que une los puntos definidos por los cuartiles primero y tercero: se la llama *recta cuartil-cuartil*. Un ajuste de los QQ-puntos a esta recta significa que la muestra se ajusta a la distribución teórica, pero posiblemente con parámetros diferentes a los especificados.
- Las curvas discontinuas abrazan una región que representa un intervalo de confianza del 95 % para el QQ-plot: si todos los puntos caen dentro de esta franja, no hay evidencia para rechazar que la muestra provenga de la distribución teórica. Como esto es justamente lo que pasa en este gráfico, hemos de aceptar que la muestra proviene de una t de Student, pero como la recta cuartil-cuartil no es exactamente la diagonal, seguramente con un número de grados de libertad ligeramente diferente.

Veamos otro ejemplo.

Ejemplo 5.3. Recordaréis el *data frame* `iris`, que nos da información sobre medidas relacionadas con las flores de una muestra de iris de tres especies. Vamos a producir un QQ-plot que ilustre si las longitudes de los sépalos de las plantas iris recogidas en esta tabla de datos siguen una distribución normal.

```
> iris.sl=iris$Sepal.Length
> qqPlot(iris.sl, distribution="norm")
```

Obtenemos la Figura 5.3. Vemos cómo los primeros puntos salen de la franja del 95 % de confianza. Esto significa que los datos están más desplazados hacia la izquierda de la media que lo que se esperaría en una muestra aleatoria de una normal: el *boxplot* en la misma figura muestra este desplazamiento. Interpretamos este QQ-plot como evidencia de que estas longitudes no siguen una distribución normal. Más adelante usaremos tests de normalidad específicos para contrastar la normalidad de estos datos.

5.4. El test χ^2 de Pearson

El test χ^2 de Pearson contrasta si una muestra ha sido generada o no con una cierta distribución, cuantificando si sus valores aparecen con una frecuencia cercana a la que sería de esperar si la muestra siguiera esa distribución. Esto se lleva a cabo calculando el estadístico de contraste

$$X^2 = \sum_{\text{clases } i} \frac{(\text{frec. observadas}_i - \text{frec. esperadas}_i)^2}{\text{frec. esperadas}_i}$$

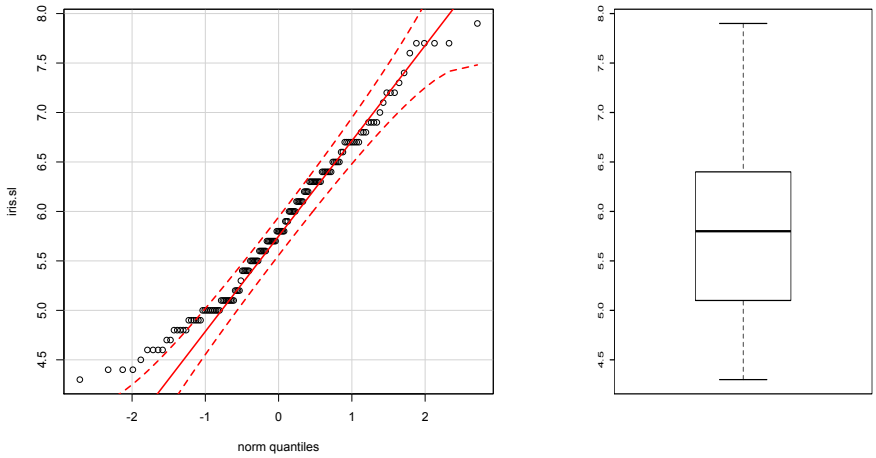


Figura 5.3. QQ-plot de las longitudes de los sépalos de las flores iris contra una normal y el *boxplot* de los datos.

que, si se satisfacen una serie de condiciones (la muestra ha de ser grande; si los posibles valores son infinitos, hay que agruparlos en un número finito de clases que cubran todos los posibles valores; y las frecuencias esperadas de las clases en las que hemos agrupado el espacio muestral han de ser todas, o al menos una gran mayoría, mayores o iguales que 5), sigue aproximadamente una ley χ^2 con un número de grados de libertad igual al número de clases menos uno y menos el número de parámetros que hemos estimado.

La instrucción básica en `R` para realizar un test χ^2 es `chisq.test`. Su sintaxis básica es

```
chisq.test(x, p=..., rescale.p=..., simulate.p.value=...)
```

donde:

- `x` es la tabla o el vector de frecuencias absolutas observadas de las clases en la muestra.
- `p` es el vector de probabilidades teóricas de las clases para la distribución que queremos contrastar. Si no lo especificamos, se entiende que la probabilidad es la misma para todas las clases. Obviamente, estas probabilidades se tienen que especificar en el mismo orden que las frecuencias de `x` y, como son las probabilidades de todos los resultados posibles, han de sumar uno.
- `rescale.p` es un parámetro lógico que, si se iguala a `TRUE`, indica que los valores de `p` no son probabilidades, sino sólo proporcionales a las probabilidades; esto hace que `R` tome como probabilidades teóricas los valores de `p` partidos por su suma, para que sumen 1. Por defecto vale `FALSE`, es decir, se supone que el vector que se entra como `p` son probabilidades y por lo tanto deben sumar 1, y si esto no pasa se genera un mensaje de error indicándolo.

Especificar que valga `TRUE` puede ser útil, porque nos permite especificar las probabilidades mediante las frecuencias esperadas o mediante porcentajes. Pero también es peligroso, porque nos puede llevar a error si los valores de `p` no corresponden a una probabilidad, sin que `R` emita ningún mensaje de advertencia.

- `simulate.p.value` es un parámetro lógico que indica a la función si debe optar por una simulación para el cálculo del p-valor del contraste. Por defecto vale `FALSE`, en cuyo caso este p-valor no se simula sino que se calcula mediante la distribución χ^2 correspondiente.

  Si se especifica como `TRUE`, `R` realiza una serie de replicaciones aleatorias de la situación teórica: por defecto, 2000, pero su número se puede especificar mediante el parámetro `B`. Es decir, se genera un conjunto de vectores aleatorios de frecuencias con la distribución que queremos contrastar, cada uno de suma total la de `x`. A continuación, calcula la proporción de estas repeticiones en las que el estadístico de contraste es mayor o igual que el obtenido para `x`, y éste es el p-valor que da. Cuando no se satisfacen las condiciones para que X^2 siga aproximadamente una distribución χ^2, estimar el p-valor mediante simulaciones puede ser una buena alternativa.

Veamos un primer ejemplo sencillo.

Ejemplo 5.4. Tenemos un dado, y queremos contrastar si está equilibrado o trucado. Lo hemos lanzado 40 veces y hemos obtenido los resultados siguientes:

resultado	1	2	3	4	5	6
frecuencia	8	4	6	3	7	12

Si el dado está equilibrado, la probabilidad de cada resultado es $1/6$ y por lo tanto la frecuencia esperada de cada resultado es $40/6 = 6.667$. Como la muestra tiene más de 30 elementos y las frecuencias esperadas son todas mayores que 5, podemos realizar de manera segura un test χ^2. Por lo tanto, entraremos estas frecuencias en un vector y le aplicaremos la función `chisq.test`. Como las probabilidades que contrastamos son todas iguales, no hace falta especificar el valor del parámetro `p`.

```
> freqs=c(8,4,6,3,7,12)
> chisq.test(freqs)

     Chi-squared test for given probabilities

data:  freqs
X-squared = 7.7, df = 5, p-value = 0.1736
```

Observemos la estructura del resultado de un `chisq.test`. Nos da el valor del estadístico X^2 (`X-squared`), los grados de libertad del mismo (`df`), y el p-valor del contraste (`p-value`). En este caso, el p-valor es 0.1736, y por lo tanto no podemos rechazar que el dado esté equilibrado.

El resultado de un `chisq.test` es una `list`, de la que podemos extraer directamente la información que deseemos con los sufijos adecuados. En concreto, podemos obtener el valor del estadístico X^2 con el sufijo `$statistic`, los grados de libertad con el sufijo `$parameter` y el p-valor con el sufijo `$p.value`.

```
> chisq.test(freqs)$statistic
X-squared
      7.7
> chisq.test(freqs)$parameter
df
 5
> chisq.test(freqs)$p.value
[1] 0.1735627
```

Imaginemos ahora que, en vez de lanzar el dado 40 veces, lo lanzamos 20 veces, y obtenemos los resultados siguientes:

resultado	1	2	3	4	5	6
frecuencia	4	2	3	2	3	6

¿Hay evidencia de que el dado esté trucado? Ahora la muestra no es grande y las frecuencias esperadas son todas $20/6 = 3.333$, menores que 5. Por tanto, el p-valor del test χ^2 que se obtiene aproximando el estadístico X^2 por una distribución χ^2_5 no tiene por qué tener ningún significado. En una situación como ésta es cuando conviene usar el parámetro `simulate.p.value`. Vamos a pedir a R que simule 5000 veces el experimento de lanzar 20 veces un dado equilibrado, y que calcule como p-valor la proporción de simulaciones en las que el estadístico X^2 haya dado un valor mayor o igual que el que se obtiene con nuestra muestra.

```
> freqs2=c(4,2,3,2,3,6)
> chisq.test(freqs2, simulate.p.value=TRUE, B=5000)

     Chi-squared test for given probabilities with simulated
     p-value (based on 5000 replicates)

data:  freqs2
X-squared = 3.4, df = NA, p-value = 0.6919
```

Resulta que en un 69 % de las simulaciones el valor de X^2 ha sido mayor o igual que el de nuestra muestra, 3.4. Por lo tanto, no hay evidencia de que el dado esté trucado.

Como este p-valor se basa en simulaciones, en cada aplicación del test el p-valor puede dar resultados diferentes, pero en general la conclusión es robusta si se toma un número suficiente de simulaciones.

```
> chisq.test(freqs2,simulate.p.value=TRUE,B=5000)$p.value
[1] 0.689862
```

```
> chisq.test(freqs2,simulate.p.value=TRUE,B=5000)$p.value
[1] 0.6968606
> chisq.test(freqs2,simulate.p.value=TRUE,B=5000)$p.value
[1] 0.7006599
```

Todos los p-valores son similares. Por curiosidad, ¿qué p-valor da el test χ^2 usando la distribución de χ^2_5?

```
> chisq.test(freqs2)$p.value
[1] 0.6385699
Warning message:
In chisq.test(freqs2) : Chi-squared approximation may be
    incorrect
```

El p-valor no es muy diferente, pero fijaos en el mensaje de advertencia: para la muestra dada, la aproximación de X^2 mediante una χ^2_5 puede ser incorrecta.

Ejemplo 5.5. Vamos a estudiar las frecuencias de los nucleótidos en una cadena de ADN, y contrastar si aparecen los cuatro con la misma probabilidad o no. En este caso, el espacio muestral son los cuatro nucleótidos: adenina (A), citosina (C), guanina (G) y timina (T). Identificaremos una cadena de ADN con un vector de letras `a`, `c`, `g`, `t`. Si llamamos p_a, p_g, p_c, p_t a las probabilidades de aparición de estas letras, el contraste que queremos realizar es

$$\begin{cases} H_0 : p_a = p_c = p_g = p_t = 0.25 \\ H_1 : \text{Algunos nucleótidos son más probables que otros} \end{cases}$$

Vamos a analizar cadenas de ADN «de verdad», extraídas de la base de datos *GenBank* (`http://www.ncbi.nlm.nih.gov/genbank/`). Para ello, utilizaremos el paquete `ape`, que incorpora una función `read.GenBank` que permite leer secuencias de genes incluidas en esta base de datos y convertirlas en vectores de letras `a`, `c`, `g`, `t`.

```
> #Instalamos y cargamos el paquete "ape"
...
> help(read.GenBank)
```

En el `help` leemos que si aplicamos la función `read.GenBank` al número de acceso (*accession number*) de una secuencia (entrado entre comillas, ya que es una palabra) y usamos el parámetro `as.character=TRUE`, nos devuelve dicha secuencia como un vector de letras junto con otra información sobre la secuencia.

En este ejemplo, nos vamos a interesar por los exones que codifican las tres partes de la mioglobina humana, que es una proteína relativamente pequeña constituida por una sola cadena polipeptídica de 153 aminoácidos. Los números de acceso de estos exones son M10090.1, M14602.1 y M14603.1. Así, pues, el código siguiente lee estos tres exones y los guarda en tres objetos.

```
> myoglobin.exon1=read.GenBank("M10090.1", as.character=TRUE)
> myoglobin.exon2=read.GenBank("M14602.1", as.character=TRUE)
```

```
> myoglobin.exon3=read.GenBank("M14603.1", as.character=TRUE)
```

Veamos la estructura de uno de estos objetos.

```
> str(myoglobin.exon1)
List of 1
 $ M10090.1: chr [1:2552] "g" "t" "a" "c" ...
 - attr(*, "species")= chr "Homo_sapiens"
```

Vemos que cada uno de ellos es en realidad una `list` formada por un solo objeto, el vector de bases, y un atributo. Ahora tenemos que extraer el vector, para poder trabajar con él. La manera más sencilla es añadiendo a cada `list` el sufijo `[[1]]`:

```
> myoglobin.exon1[[1]][1:10]
 [1] "g" "t" "a" "c" "t" "g" "t" "a" "t" "t"
```

Así pues, vamos a quedarnos sólo con las cadenas de los tres exones.

```
> myoglobin.exon1.nuc=myoglobin.exon1[[1]]
> myoglobin.exon2.nuc=myoglobin.exon2[[1]]
> myoglobin.exon3.nuc=myoglobin.exon3[[1]]
```

Nos preguntamos si en alguna de estas tres secuencias las cuatro bases aparecen de manera equiprobable. Para responder esta pregunta, calculamos las frecuencias de las letras en cada secuencia (con `table`) y aplicamos el test χ^2 a los resultados. Puesto que miramos si todos los resultados aparecen con la misma probabilidad, no hace falta especificar el vector `p` de probabilidades.

```
> table(myoglobin.exon1.nuc)
myoglobin.exon1.nuc
  a   c   g   t
709 450 798 595
> chisq.test(table(myoglobin.exon1.nuc))

     Chi-squared test for given probabilities

data:  table(myoglobin.exon1.nuc)
X-squared = 106.3229, df = 3, p-value < 2.2e-16
```

El p-valor es prácticamente 0, podemos rechazar que las cuatro bases aparezcan con la misma probabilidad. Veamos los otros dos exones.

```
> table(myoglobin.exon2.nuc)
myoglobin.exon2.nuc
  a   c   g   t
686 621 780 584
> chisq.test(table(myoglobin.exon2.nuc))
```

```
      Chi-squared test for given probabilities

data:  table(myoglobin.exon2.nuc)
X-squared = 33.1453, df = 3, p-value = 3.001e-07

> table(myoglobin.exon3.nuc)
myoglobin.exon3.nuc
  a   c   g   t
323 382 441 320
> chisq.test(table(myoglobin.exon3.nuc))

     Chi-squared test for given probabilities

data:  table(myoglobin.exon3.nuc)
X-squared = 26.8622, df = 3, p-value = 6.292e-06
```

En los tres contrastes tenemos que rechazar la hipótesis nula: en ninguna de las tres cadenas parece que las bases aparezcan con la misma probabilidad.

Ejemplo 5.6. Siguiendo con el ejemplo anterior, vamos a contrastar ahora si las bases del exón 1 siguen una distribución en la que A y G aparecen un 50 % de veces más que C y T. Usaremos `p=c(1.5,1,1.5,1)` para especificar estas proporciones, y entonces es necesario especificar `rescale.p=TRUE`.

```
> chisq.test(table(myoglobin.exon1.nuc),
  p=c(1.5,1,1.5,1))$p.value #MAL!
Error in chisq.test(table(myoglobin.exon1.nuc), p = c(1.5, 1,
    1.5, 1)) : probabilities must sum to 1.
> chisq.test(table(myoglobin.exon1.nuc), p=c(1.5,1,1.5,1),
  rescale.p=TRUE)$p.value
[1] 6.720275e-06
```

De nuevo, tenemos que rechazar la hipótesis nula.

Ejemplo 5.7. Vamos a realizar otro experimento con los datos de los ejemplos anteriores. Construiremos una cadena formada por los tres exones, y vamos a comparar si la frecuencia de bases en cada exón es similar a la de la cadena total, que hará de distribución teórica.

```
> myoglobin.todos.nuc=c(myoglobin.exon1.nuc,myoglobin.exon2.nuc,
    myoglobin.exon3.nuc)   #La cadena total
> probs.tot=prop.table(table(myoglobin.todos.nuc))
```

La tabla `probs.tot` contiene las frecuencias relativas de cada base en la cadena completa. La vamos a usar como parámetro `p` de la función `chisq.test`:

```
> chisq.test(table(myoglobin.exon1.nuc),p=probs.tot)$p.value
[1] 9.784579e-06
> chisq.test(table(myoglobin.exon2.nuc),p=probs.tot)$p.value
```

```
[1] 0.2533979
> chisq.test(table(myoglobin.exon3.nuc),p=probs.tot)$p.value
[1] 0.0001324974
```

Los exones 1 y 3 dan unos p-valores muy pequeños, y por lo tanto podemos rechazar que en estos exones las bases aparezcan con la misma probabilidad que en la cadena total. En cambio, el exón 2 da un p-valor de 0.2534, por lo que para este exón no podemos rechazar la hipótesis nula y podemos concluir que las bases aparecen en él con la misma probabilidad que en la cadena total. Podemos ilustrar estas conclusiones gráficamente mediante un diagrama de barras conjunto.

```
> barplot(rbind(prop.table(table(myoglobin.todos.nuc)),
  prop.table(table(myoglobin.exon1.nuc)),
  prop.table(table(myoglobin.exon2.nuc)),
  prop.table(table(myoglobin.exon3.nuc))),
  beside=TRUE, legend=c("Totales","Exon1","Exon2","Exon3"),
  ylim=c(0,0.45), main="Frecuencias relativas")
```

Obtenemos el gráfico de la Fig. 5.4, que confirma que las frecuencias relativas de las bases en el exón 2 se ajustan bastante a las globales.

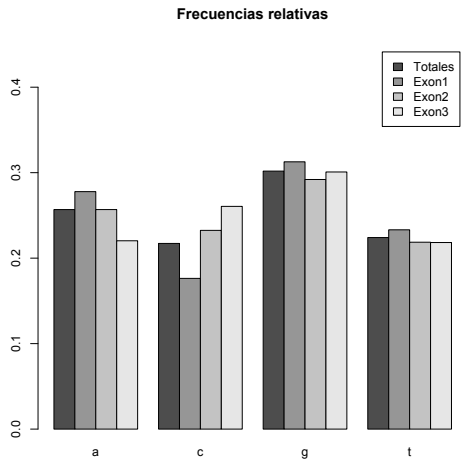


Figura 5.4. Frecuencias relativas de las cuatro bases en los tres exones de la mioglobina humana y en la cadena total.

Ejemplo 5.8. Vamos ahora a realizar el experimento siguiente. Tomaremos la secuencia completa de la mioglobina humana, que tenemos almacenada en `myoglobin.todos.nuc`, y repetiremos 100 veces el proceso siguiente: extraemos una muestra aleatoria simple de 20 posiciones de la secuencia y contamos cuántas de ellas contienen el par de bases «cg».

Fijamos la semilla de aleatoriedad para que se pueda reproducir el experimento. El código, que luego explicamos, y el resultado son los siguientes:

```
> cg.in.sample=function(x,S){#x un vector, S vector de índices
  x.despl=c(x[-1],0);
  length(which(x[S]=="c" & x.despl[S]=="g"))
}
> set.seed(1660)  #Año de nacimiento de D. Defoe
> muestra=replicate(100, cg.in.sample(myoglobin.todos.nuc,
   sample(1:(length(myoglobin.todos.nuc)-1), 20,
  replace=TRUE)))
> muestra
  [1] 2 1 1 0 0 0 0 2 1 0 2 0 0 0 2 0 0 0 1 0 1 0 1 1 0 0 0 0 0
 [30] 0 0 1 0 2 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0
 [59] 0 0 1 0 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 2 0 0 0 0 1 0 0
 [88] 0 0 1 1 0 1 0 0 0 1 0 0 0
```

La función `cg.in.sample` que hemos definido toma un vector x y un vector de índices S y cuenta el número de índices $j \in S$ en los que $x[j]$ es `c` y $x[j+1]$ es `g`. Para hacerlo de manera sencilla, crea una copia de x desplazada a la izquierda, $x.despl$, que empieza en la segunda posición de x y termina con un 0 extra al final para conservar la longitud, y cuenta los índices $j \in S$ en los que x tiene una `c` y $x.despl$ una `g`. Entonces, con el `replicate`, hemos repetido 100 veces el proceso de extraer una muestra aleatoria simple de índices de `myoglobin.todos.nuc` (excluyendo el último, para que sean posiciones donde empieza un par de letras) y aplicar la función `cg.in.sample` a `myoglobin.todos.nuc` y a esta muestra de índices.

Queremos contrastar si esta muestra sigue una distribución binomial, y dos casos de esta pregunta. Por un lado, con probabilidad de aparición de la pareja «cg» $0.25 \cdot 0.25 = 0.0625$, que correspondería al hecho de que las dos bases aparecieran de manera equiprobable e independiente. Por otro lado, estimando el valor «real» de p.

Empezamos con el primer caso. Calculemos las frecuencias con las que aparecen los diferentes resultados en la muestra, y las frecuencias esperadas con las que deberían aparecer si siguieran una distribución $B(20, 0.0625)$.

```
> table(muestra)
muestra
 0  1  2
68 26  6
> round(dbinom(0:20,20,0.0625)*100,2)  #Frecuencias teóricas
 [1] 27.51 36.67 23.23  9.29  2.63  0.56  0.09  0.01  0.00  0.00
[11]  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
[21]  0.00
```

Las frecuencias teóricas a partir de 4 son inferiores a 5, y además su suma es inferior a 5. Por lo tanto, vamos a agrupar en una sola clase los resultados mayores o iguales que 3. La nueva tabla de frecuencias esperadas es:

```
> round(c(dbinom(0:2,20,0.0625),1-pbinom(2,20,0.0625))*100,2)
[1] 27.51 36.67 23.23 12.59
```

Ahora tenemos dos opciones: o bien tomamos como resultados posibles «0», «1», «2» y «3 o más», en cuyo caso contaríamos que hemos observado 0 veces este último resultado en nuestra muestra, o bien tomamos como resultados posibles «0», «1», y «2 o más», que se corresponde con los valores observados. Como norma general, es recomendable usar el mayor número de clases, ya que de esta manera aumenta la potencia del contraste. Por consiguiente, vamos a optar por la primera estrategia: 4 clases en vez de 3.

```
> freq.obs=c(table(muestra),0)
> prob.teor=c(dbinom(0:2,20,0.0625),1-pbinom(2,20,0.0625))
> chisq.test(freq.obs,p=prob.teor)$p.value
[1] 5.628013e-19
```

El p-valor es prácticamente 0, por lo que podemos concluir que la muestra no sigue una distribución $B(20, 0.0625)$.

¿Hubiera variado la conclusión si hubiéramos optado por sólo considerar tres clases?

```
> freq.obs=table(muestra)
> prob.teor=c(dbinom(0:1,20,0.0625),1-pbinom(1,20,0.0625))
> chisq.test(freq.obs,p=prob.teor)$p.value
[1] 9.759239e-20
```

El p-valor es prácticamente el mismo, la conclusión es la misma.

Pasemos al segundo caso de nuestro problema, vamos a estimar el parámetro p. Como la binomial es otra de las distribuciones no cubiertas por `fitdistr`, lo tenemos que hacer apelando a lo que sabemos de teoría. Sabemos que el valor esperado de una variable aleatoria $X \sim B(n, p)$ es np, por lo que podemos estimar p mediante $\overline{X}/n$. De hecho, éste es el estimador máximo verosímil de p cuando n es conocida.

```
> mean(muestra)/20
[1] 0.019
```

Repetimos el proceso: calculemos las frecuencias teóricas

```
> round(dbinom(0:20,20,0.019)*100,2)
 [1] 68.14 26.39  4.86  0.56  0.05  0.00  0.00  0.00  0.00
[10]  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
[19]  0.00  0.00  0.00
```

En este caso, hemos de agrupar los resultados en «0», «1» y «2 o más», para que las frecuencias teóricas sean mayores que 5. Coincide con los diferentes valores observados en la muestra.

```
> freq.obs
```

```
muestra
 0  1  2
68 26  6
> prob.teor=c(dbinom(0:1,20,0.019),1-pbinom(1,20,0.019))
> chisq.test(freq.obs,p=prob.teor)

  Chi-squared test for given probabilities

data:  freq.obs
X-squared = 0.0575, df = 2, p-value = 0.9717
```

¡Cuidado! Este p-valor no es el correcto. Hemos estimado un parámetro, pero `R` no lo sabe. Por lo tanto tenemos que bajar en 1 los grados de libertad y calcular el p-valor a mano, mediante

$$1 - P(\chi_1^2 \geqslant X^2) = 1 - P(\chi_1^2 \geqslant 0.0575)$$

```
> 1-pchisq(0.0575,1)
[1] 0.8104918
```

El p-valor es grande. De nuestro experimento concluimos que no podemos rechazar la hipótesis nula de que las apariciones de «cg» en muestras aleatorias de 20 posiciones sigan una ley binomial de parámetro $p = 0.019$.

5.5. El test χ^2 para distribuciones continuas

El procedimiento de contraste de bondad de ajuste mediante el test χ^2 para variables continuas tiene la particularidad de que es necesario un paso preliminar que consiste en definir los intervalos de clase para los que realizaremos el conteo de las frecuencias observadas. El proceso es similar al que estudiamos[5] para dibujar histogramas. Por lo tanto, necesitaremos definir unos intervalos de clase para el conteo de frecuencias absolutas observadas, y con las funciones `cut` y `table` obtendremos las frecuencias observadas de estas clases en la muestra de la variable continua.

Para obtener los intervalos podemos seguir dos estrategias razonables: reutilizar los generados por la función `hist`, o dividir el rango de la variable en un número prefijado K de intervalos de amplitud fija. Vamos a ver en detalle un ejemplo de cada tipo.

Ejemplo 5.9. Vamos a contrastar si las longitudes de los sépalos de las plantas iris recogidas en la tabla de datos `iris` siguen una distribución normal. Recordaréis que el QQ-plot de estas longitudes que obteníamos en el Ejemplo 5.3 mostraba evidencia de que no la siguen.

Primero vamos a estimar la media y la desviación típica de la distribución de estas longitudes.

[5] En las Lecciones 19 y 20 del primer volumen.

```
> iris.sl=iris$Sepal.Length
> mu=fitdistr(iris.sl,"normal")$estimate[1]
> sigma=fitdistr(iris.sl,"normal")$estimate[2]
> round(c(mu,sigma),3)
 mean    sd
5.843 0.825
```

En este ejemplo, usaremos los intervalos en los que agrupa por defecto estos datos la función `hist`.

```
> h=hist(iris.sl, plot=FALSE)
> h$breaks
[1] 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0
```

Ahora se nos presenta el problema de que los intervalos que definen estos puntos de corte (*breaks*) no cubren toda la recta real, que es el espacio muestral de una variable aleatoria normal. Así que hemos de reemplazar los extremos de este vector de `breaks` por los límites del espacio muestral de la variable, que en este caso son $-\infty$ e ∞.

```
> breaks2=h$breaks
> breaks2[1]=-Inf    #Cambiamos el 4 por -Infinito
> breaks2[length(breaks2)]=Inf   #Cambiamos el 8 por Infinito
> breaks2
[1] -Inf  4.5  5.0  5.5  6.0  6.5  7.0  7.5  Inf
```

Ahora podemos calcular las frecuencias de la muestra en los intervalos definidos por estos puntos de corte:

```
> freq.obs=table(cut(iris.sl,breaks=breaks2))
> freq.obs
(-Inf,4.5]    (4.5,5]    (5,5.5]    (5.5,6]
         5         27         27         30
   (6,6.5]    (6.5,7]    (7,7.5] (7.5, Inf]
        31         18          6          6
```

Ahora calcularemos las probabilidades teóricas. Para cada intervalo $(x, y]$ en los que hemos cortado la recta real, tenemos que calcular

$$P(x < X \leqslant y) = P(X \leqslant y) - P(X \leqslant x) = \texttt{pnorm(y,mu,sigma)-pnorm(x,mu,sigma)}.$$

Definimos dos vectores que nos den los extremos izquierdos y derechos de cada intervalo.

```
> extremo.izq=breaks2[1:(length(breaks2)-1)]
> extremo.der=breaks2[-1]
> extremo.izq
[1] -Inf  4.5  5.0  5.5  6.0  6.5  7.0  7.5
> extremo.der
```

```
[1] 4.5 5.0 5.5 6.0 6.5 7.0 7.5 Inf
```

Ahora podemos calcular las probabilidades y las frecuencias esperadas de todos los intervalos de golpe:

```
> probs.teor=pnorm(extremo.der,mu,sigma)-pnorm(extremo.izq,mu,
    sigma) #Probabilidades teóricas
> probs.teor
[1] 0.05179549 0.10163069 0.18527528 0.23657724 0.21160908
[6] 0.13258117 0.05817473 0.02235633
> freq.esp=probs.teor*length(iris.sl) #Frecuencias esperadas
> round(freq.esp,3)
[1]  7.769 15.245 27.791 35.487 31.741 19.887  8.726  3.353
```

La frecuencia esperada de la última clase es inferior a 5, así que vamos a agruparla con la penúltima y así la clase resultante tendrá una frecuencia esperada superior a 5.

```
> length(probs.teor)
[1] 8
> probs.teor2=c(probs.teor[1:6],sum(probs.teor[7:8])) #Nuevas
    probabilidades teóricas
> freq.obs2=c(freq.obs[1:6],sum(freq.obs[7:8])) #Nuevas
    frecuencias observadas
> chisq.test(freq.obs2,p=probs.teor2)

     Chi-squared test for given probabilities

data:  freq.obs2
X-squared = 11.1196, df = 6, p-value = 0.08475
```

Recordemos que el p-valor obtenido no es el del test que realizamos: como hemos estimado dos parámetros, lo tenemos que calcular con una χ^2 con 4 grados de libertad (dos menos de los que ha usado `chisq.test`):

```
> 1-pchisq(11.1196, 4)
[1] 0.02525225
```

El p-valor es inferior a 0.05, por tanto obtenemos evidencia de que la muestra no proviene de una población normal, es decir, de que las longitudes de los sépalos de las flores iris no siguen una ley normal.

Podríamos automatizar el cálculo del p-valor verdadero de la manera siguiente:

```
> test.iris=chisq.test(freq.obs2,p=probs.teor2)
> 1-pchisq(test.iris$statistic,test.iris$parameter-2)
X-squared
0.02525177
```

Ejemplo 5.10. Vamos a repetir el estudio del ejemplo anterior, pero ahora calculando a mano los intervalos. En general, el número de intervalos debe ser suficiente para cubrir toda la forma de la distribución, pero tampoco conviene que haya muchos para evitar frecuencias esperadas demasiado pequeñas que obliguen a agrupar intervalos. Por decir una cantidad, para una distribución normal es costumbre tomar entre 5 y 15 intervalos. Otra posibilidad es decidir el número de intervalos con alguna de las reglas explicadas en la Lección 19 del primer volumen.

En nuestro ejemplo, vamos a usar 10 intervalos. Para calcularlos, tomamos el máximo y el mínimo de las observaciones, los restamos y dividimos por el número de intervalos (y, si fuera necesario, redondearíamos adecuadamente).

```
> Ampl=(max(iris.sl)-min(iris.sl))/10
> Ampl
[1] 0.36
```

Los extremos de los intervalos en los que dividimos la muestra forman la secuencia que empieza en el mínimo y va sumando la amplitud hasta definir los $k = 10$ intervalos. Luego hay que adecuar los dos extremos para que cubran el dominio de la densidad de la distribución teórica, en nuestro caso la recta real.

```
> breaks=min(iris.sl)+Ampl*(0:10)
> breaks
 [1] 4.30 4.66 5.02 5.38 5.74 6.10 6.46 6.82 7.18 7.54 7.90
> breaks2=breaks
> breaks2[1]=-Inf
> breaks2[length(breaks2)]=Inf
> breaks2
 [1] -Inf 4.66 5.02 5.38 5.74 6.10 6.46 6.82 7.18 7.54  Inf
```

Calculemos, como en el ejemplo anterior, las frecuencias observadas, las probabilidades teóricas y las frecuencias esperadas.

```
> frec.obs=table(cut(iris.sl,breaks=breaks2))
> frec.obs

(-Inf,4.66] (4.66,5.02] (5.02,5.38] (5.38,5.74]  (5.74,6.1]
          9          23          14          27          22
 (6.1,6.46] (6.46,6.82] (6.82,7.18] (7.18,7.54] (7.54, Inf]
         20          18           6           5           6
> mu=mean(iris.sl)
> sigma=sd(iris.sl)
> extremo.izq=breaks2[1:(length(breaks2)-1)]
> extremo.der=breaks2[-1]
> prob.teor=pnorm(extremo.der,mu,sigma)-pnorm(extremo.izq,mu,
    sigma)
> frec.esp=round(prob.teor*length(iris.sl),2)
> frec.esp
```

```
 [1] 11.37 12.51 19.20 24.44 25.79 22.56 16.37  9.85  4.91  2.99
```

Agruparemos las frecuencias de los dos últimos intervalos y aplicaremos el test χ^2 con el número adecuado de grados de libertad:

```
> frec.obs2=c(frec.obs[1:8], sum(frec.obs[9:10]))
> prob.teor2=c(prob.teor[1:8], sum(prob.teor[9:10]))
> test.iris.2=chisq.test(freq.obs2, p=probs.teor2)
> 1-pchisq(test.iris.2$statistic, test.iris.2$parameter-2)
 X-squared
0.02525177
```

El p-valor es inferior a 0.05: de esta manera también obtenemos evidencia significativa de que la muestra no proviene de una población normal.

5.6. El test de Kolgomorov-Smirnov

El test de Kolgomorov-Smirnov (K-S) es un test genérico para contrastar la bondad de ajuste a distribuciones continuas. Se puede usar con muestras pequeñas (se suele recomendar 5 elementos como el tamaño mínimo para que el resultado sea significativo), pero la muestra no puede contener valores repetidos.

Hay que tener en cuenta que el test K-S realiza un contraste en el que la hipótesis nula es que la muestra proviene de una distribución contínua completamente especificada. Es decir, no sirve para contrastar si la muestra proviene, pongamos, de «alguna» distribución normal, sino sólo para contrastar si proviene de una distribución normal con una media y una desviación típica determinadas. Por lo tanto, si queremos contrastar que la muestra proviene de alguna distribución de una familia concreta y estimamos sus parámetros a partir de la muestra, el test K-S sólo nos permite rechazar o no la hipótesis de que la muestra proviene de la distribución de esa familia con exactamente esos parámetros. Por lo tanto, si el resultado es rechazar la hipótesis nula, esto no excluye que la muestra provenga de una distribución de la misma familia con otros parámetros. En la próxima sección veremos algunos tests que permiten contrastar, en general, si una muestra proviene de alguna distribución normal.

La función básica para realizar el test K-S es `ks.test`. Su sintaxis básica para una muestra es

```
ks.test(x, y, parámetros)
```

donde:

- `x` es la muestra de una variable continua.
- `y` tanto puede ser un segundo vector, y entonces se contrasta si ambos vectores han sido generados por la misma distribución continua, o el nombre de la función de

distribución que queremos contrastar, entre comillas; por ejemplo `"pnorm"` para la distribución normal.

- Los `parámetros` de la función de distribución si se ha especificado una; por ejemplo `mean=0, sd=1` para una distribución normal estándar.

Ejemplo 5.11. Efectuemos el test de Kolmogorov-Smirnov para contrastar si las longitudes de sépalos de flores iris siguen una distribución normal de media y desviación típica sus estimaciones máximo verosímiles a partir la muestra:

```
> iris.sl=iris$Sepal.Length
> mu=fitdistr(iris.sl,"normal")$estimate[1]
> sigma=fitdistr(iris.sl,"normal")$estimate[2]
> round(c(mu,sigma),3)
 mean    sd
5.843 0.825
> ks.test(iris.sl, "pnorm", mean=mu, sd=sigma)

  One-sample Kolmogorov-Smirnov test

data:  iris.sl
D = 0.0895, p-value = 0.1812
alternative hypothesis: two-sided

Warning message:
In ks.test(iris.sl, "pnorm", mu, sigma) :
  ties should not be present for the Kolmogorov-Smirnov test
```

Obtenemos un p-valor de 0.1812, que no nos permite rechazar la hipótesis de que siguen una ley $N(5.843, 0.825)$. Pero `R` nos avisa de que hay empates. ¿Hay muchos? La función `unique` aplicada a un vector nos da la lista de sus elementos sin repeticiones.

```
> length(unique(iris.sl))
[1] 35
> 1-length(unique(iris.sl))/length(iris.sl)
[1] 0.7666667
```

Por tanto, el vector (de 150 entradas) de longitudes de sépalos sólo tiene 35 valores diferentes. El resto, un 76.67 %, son valores repetidos. Hay muchos empates, y el resultado de este test en este caso es poco significativo.

Como hemos comentado, el test K-S también se puede usar para contrastar si dos muestras se han obtenido de poblaciones con la misma distribución continua. Para hacerlo, se ha de aplicar la función `ks.test` a las dos muestras.

Ejemplo 5.12. La tabla de datos `Salaries` del paquete `car` contiene información sobre los sueldos de 397 profesores de una universidad americana en el curso 2008-09.

```
> str(Salaries)
'data.frame': 397 obs. of  6 variables:
 $ rank         : Factor w/ 3 levels "AsstProf","AssocProf",..:
   3 3 1 3 3 2 3 3 3 3 ...
 $ discipline   : Factor w/ 2 levels "A","B": 2 2 2 2 2 2 2 2 2
   2 ...
 $ yrs.since.phd: int  19 20 4 45 40 6 30 45 21 18 ...
 $ yrs.service  : int  18 16 3 39 41 6 23 45 20 18 ...
 $ sex          : Factor w/ 2 levels "Female","Male": 2 2 2 2 2
   2 2 2 2 1 ...
 $ salary       : int  139750 173200 79750 115000 141500 97000
   175000 147765 119250 129000 ...
```

La variable `sex` nos da el sexo del profesor y la variable `salary` su sueldo anual. Queremos contrastar si los sueldos de hombres y mujeres siguen la misma distribución. Para ello, vamos a suponer que provienen de distribuciones continuas (en la tabla de datos se han redondeado a dólares) y usaremos el test K-S. Primero miraremos si hay muchos empates en cada muestra.

```
> sal.male=Salaries[Salaries$sex=="Male",]$salary
> sal.female=Salaries[Salaries$sex=="Female",]$salary
> 1-length(unique(sal.female))/length(sal.female)
[1] 0.05128205
> 1-length(unique(sal.male))/length(sal.male)
[1] 0.05027933
> 1-length(unique(Salaries$salary))/length(Salaries$salary)
[1] 0.06549118
```

Las repeticiones en cada lista significan alrededor del 5 % de los datos, y en total un 6.5 % de los datos. No son muchas, así que vamos a arriesgarnos con el test K-S.

```
> ks.test(sal.male,sal.female)

     Two-sample Kolmogorov-Smirnov test

data:  sal.male and sal.female
D = 0.2472, p-value = 0.02715
alternative hypothesis: two-sided

Warning message:
In ks.test(sal.male, sal.female) :
  p-value will be approximate in the presence of ties
```

El p-valor pequeño nos permite rechazar que los salarios de hombres y mujeres sigan la misma distribución. Si dibujamos un boxplot de los salarios según el sexo, obtenemos el gráfico de la izquierda de la Figura 5.5, que muestra que los sueldos de los hombres tienen mayor media y variabilidad que los de las mujeres. Si cancelamos este efecto, estandarizando

las muestras, ¿siguen saliendo distribuciones diferentes?

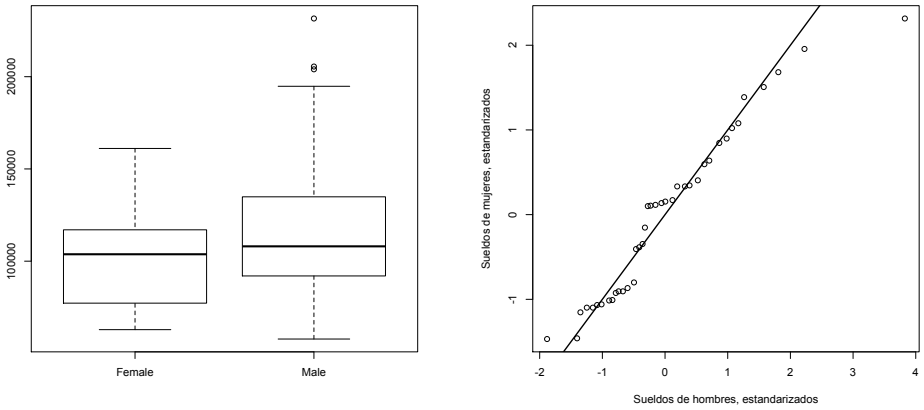
```
> ks.test(scale(sal.male),scale(sal.female))

     Two-sample Kolmogorov-Smirnov test

data:  scale(sal.male) and scale(sal.female)
D = 0.139, p-value = 0.5049
alternative hypothesis: two-sided

Warning message:
In ks.test(scale(sal.male), scale(sal.female)) :
  p-value will be approximate in the presence of ties
```

Al estandarizar, ya no tenemos evidencia de que provengan de distribuciones diferentes. El QQ-plot de los sueldos estandarizados de la Figura 5.5 muestra que, salvo un sueldo extremo de un hombre en la esquina superior derecha (¿el rector?), los QQ-puntos de los sueldos estandarizados se ajustan bastante a la diagonal.



(a) (b)

Figura 5.5. (a) Boxplot de sueldos según el sexo en la tabla de datos `Salaries`. (b) QQ-plot de sueldos estandarizados de hombres y mujeres.

5.7. Tests de normalidad

Existen algunos tests específicos de normalidad que permiten contrastar si una muestra proviene de alguna distribución normal. El más conocido es el *test de normalidad de*

Kolmogorov-Smirnov-Lilliefors (*K-S-L*). Se trata de una variante del test K-S, y se puede realizar aplicando a la muestra la función `lillie.test` del paquete `nortest`.

Vamos a usar el test K-S-L para contrastar si las longitudes de los sépalos de las iris siguen una ley normal.

```
> #Instalamos y cargamos el paquete "nortest"
...
> iris.sl=iris$Sepal.Length
> lillie.test(iris.sl)

     Lilliefors (Kolmogorov-Smirnov) normality test

data:  iris.sl
D = 0.0887, p-value = 0.005788
```

El p-valor es muy pequeño, y nos permite rechazar que la muestra provenga de una población normal.

La ventaja del test K-S-L es que es muy conocido, ya que es una variante del K-S (incluso usa el mismo estadístico), pero tiene un inconveniente: aunque es muy sensible a las diferencias entre la muestra y la distribución teórica alrededor de sus valores medios, le cuesta detectar diferencias prominentes en un extremo u otro de la distribución. Esto afecta su potencia. Por ejemplo, sabemos que una t de Student se parece bastante a una normal estándar, pero su densidad es algo más aplanada y hace que en los dos extremos esté por encima de la de la normal. Al test K-S-L le cuesta detectar esta discrepancia, como podemos ver en el siguiente ejemplo:

```
> set.seed(100)
> x=rt(50,3)   #Una muestra de una t de student con 3 g.l.
> lillie.test(x)

     Lilliefors (Kolmogorov-Smirnov) normality test

data:  x
D = 0.1033, p-value = 0.2013
```

Este inconveniente del test K-S-L lo resuelve el *test de normalidad de Anderson-Darling* (A-D). Para realizarlo podemos usar la función `ad.test` del paquete `nortest`. Los detalles del estadístico que usa los encontraréis en el `help` de la función.

```
> ad.test(iris.sl)

     Anderson-Darling normality test

data:  iris.sl
A = 0.8892, p-value = 0.02251
```

De nuevo obtenemos un p-valor muy pequeño. Veamos ahora que este test sí que distingue que la muestra anterior de una t de student con 3 grados de libertad no es normal:

```
> set.seed(100)
> x=rt(50,3)
> ad.test(x)

     Anderson-Darling normality test

data:  x
A = 1.1657, p-value = 0.004334
```

Un inconveniente común a los tests K-S-L y A-D es que, si bien pueden usarse con muestras pequeñas (pongamos de más de 5 elementos), se comportan mal con muestras grandes, de varios miles de elementos. En muestras de este tamaño, cualquier pequeña divergencia de la normalidad se magnifica y en estos dos tests aumenta la probabilidad de errores de tipo I. Un test que resuelve este problema es el de *Shapiro-Wilk* (*S-W*), implementado en la función `shapiro.test` de la instalación básica de `R`. Este test es importante, porque un experimento reciente ha mostrado evidencia significativa de que su potencia es mayor que la de los tests anteriores.[6] De nuevo, los detalles del estadístico que usa los encontraréis en el `help` de la función.

```
> shapiro.test(iris.sl)

  Shapiro-Wilk normality test

data:  iris.sl
W = 0.9761, p-value = 0.01018
> set.seed(100)
> x=rt(50,3)
> shapiro.test(x)

  Shapiro-Wilk normality test

data:  x
W = 0.8949, p-value = 0.0003285
```

Un último inconveniente que afecta a todos los tests explicados hasta ahora es el de los empates. Sus estadísticos tienen las distribuciones que se usan para calcular los p-valores cuando la muestra no tiene datos repetidos, y por lo tanto, si hay muchos, el p-valor puede no tener ningún significado. De los tres, el menos sensible a repeticiones es el S-W, pero si hay muchas es conveniente usar un test que no sea sensible a éstas, como por ejemplo el *test omnibus de D'Agostino-Pearson*. Este test se encuentra implementado en la función `dagoTest` del paquete `fBasics`, y lo que hace es cuantificar lo diferentes que son

[6] Véase N. M. Razali, Y. B. Wah, "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests." *S. Stat. Model. Anal.* 2 (2011), pp. 21–33.

la asimetría y la curtosis de la muestra (dos parámetros estadísticos relacionados con la forma de una distribución) respecto de los esperados en una distribución normal, y resume esta discrepancia en un p-valor con el significado usual.

```
> #Instalamos y cargamos el paquete "fBasics"
...
> dagoTest(iris.sl)

Title:
 D'Agostino Normality Test

Test Results:
  STATISTIC:
    Chi2 | Omnibus: 5.7356
    Z3   | Skewness: 1.5963
    Z4   | Kurtosis: -1.7853
  P VALUE:
    Omnibus  Test: 0.05682
    Skewness Test: 0.1104
    Kurtosis Test: 0.07421
```

El p-valor relevante es el del «Omnibus test», en este caso 0.05682, cae en la zona de penumbra.

Queremos hacer una última advertencia en esta sección. Aunque los tests que hemos explicado se pueden aplicar a muestras pequeñas, es muy difícil rechazar la normalidad de una muestra muy pequeña. Por ejemplo, una muestra de 10 valores escogidos con distribución uniforme entre 0 y 5 pasa holgadamente todos los tests de normalidad (salvo el de D'Agostino-Pearson, que requiere una muestra de al menos 20 elementos):

```
> set.seed(100)
> x=runif(10,0,5)
> lillie.test(x)

     Lilliefors (Kolmogorov-Smirnov) normality test

data:  x
D = 0.1459, p-value = 0.7903

> ad.test(x)

     Anderson-Darling normality test

data:  x
A = 0.1663, p-value = 0.9119

> shapiro.test(x)
```

```
  Shapiro-Wilk normality test

data:  x
W = 0.9803, p-value = 0.9666

> dagoTest(x)
Error in .omnibus.test(x) : sample size must be at least 20
```

5.8. Guía rápida

- `fitdistr`, del paquete `MASS`, sirve para calcular los estimadores máximo verosímiles de los parámetros de una distribución a partir de una muestra. Sus parámetros principales son:
  - `densfun`: el nombre de la familia de distribuciones, entre comillas.
  - `start`: permite fijar el valor inicial del algoritmo numérico para calcular el estimador, si la función lo requiere.

  El resultado es una `list` que incluye los objetos `estimate` (el valor del estimador) y `sd` (la desviación estándar de la estimación).
- `qqplot` sirve para dibujar un QQ-plot de dos muestras. Admite, entre otros, los parámetros usuales de `plot`.
- `qqPlot`, del paquete `car`, sirve para dibujar un QQ-plot de una muestra contra una distribución teórica. Sus parámetros principales son:
  - `distribution`: el nombre de la familia de distribuciones, entre comillas.
  - Los parámetros de la distribución: `mean` para la media, `sd` para la desviación típica, `df` para los grados de libertad, etc.
  - Los parámetros usuales de `plot`.
- `chisq.test` sirve para realizar tests χ^2 de bondad de ajuste. Sus parámetros principales son:
  - `p`: el vector de probabilidades teóricas.
  - `rescale.p`: igualado a `TRUE`, indica que los valores de `p` no son probabilidades, sino sólo proporcionales a las probabilidades.
  - `simulate.p.value`: igualado a `TRUE`, `R` calcula el p-valor mediante simulaciones.
  - `B`: en este último caso, permite especificar el número de simulaciones.

  El resultado es una `list` formada por, entre otros, los objetos siguientes: `statistic` (el valor del estadístico X^2), `parameter` (los grados de libertad) y `p.value` (el p-valor).

- `read.GenBank("accession_number", as.character=TRUE)[[1]]`, del paquete `ape`, importa la secuencia de genes de la base de datos *GenBank* indicada por el número de acceso, en forma de un vector de letras.
- `ks.test` realiza el test de Kolmogorov-Smirnov. Tiene dos tipos de uso:
  - `ks.test(x,y)`: contrasta si los vectores x e y han sido generados por la misma distribución continua.
  - `ks.test(x, "distribución", parámetros)`: contrasta si el vector x ha sido generado por la distribución especificada, que se ha de indicar con el nombre de la función de distribución de `R` (la que empieza con p).
- `lillie.test`, del paquete `nortest`, realiza el test de normalidad de Kolmogorov-Smirnov-Lilliefors.
- `ad.test`, del paquete `nortest`, realiza el test de normalidad de Anderson-Darling.
- `shapiro.test`, del paquete `nortest`, realiza el test de normalidad de Shapiro-Wilk.
- `dagoTest`, del paquete `fBasics`, realiza el test omnibus de D'Agostino-Pearson.

Modelo de test

(1) Las distribuciones de Weibull tienen dos parámetros, forma (`shape`) y escala (`scale`). Supongamos que los datos siguientes siguen una distribución de Weibull: 2.46, 2.28, 1.7, 0.62, 0.87, 2.81, 2.35, 2.08, 2.11, 1.72. Calculad el estimador máximo verosímil del parámetro de escala de esta distribución, redondeado a 3 cifras decimales. Tenéis que dar el resultado (sin ceros innecesarios a la derecha), no cómo lo habéis calculado.

(2) Un determinado experimento tiene cinco resultados posibles: A, B, C, D, E. Lo repetimos un cierto número de veces y obtenemos 65 veces el resultado A, 95 veces el resultado B, 87 veces el resultado C, 70 veces el resultado D y 193 veces el resultado E. Realizad un test χ^2 para contrastar si los resultados A, B, C y D tienen la misma probabilidad y E tiene el doble de probabilidad que cada uno de los otros resultados. Dad el p-valor del contraste (redondeado a 3 cifras decimales, sin ceros innecesarios a la derecha) y decid (contestando SI, en mayúsculas y sin acento, o NO) si tendríamos que rechazar la hipótesis nula con un nivel de significación $\alpha = 0.05$. Tenéis que dar las respuestas en este orden y separadas por un único espacio en blanco.

(3) Queremos contrastar si una determinada variable sigue una distribución de Poisson. Hemos efectuado algunas observaciones y hemos obtenido 10 veces el resultado 0, 32 veces el resultado 1, 18 veces el resultado 2, 19 veces el resultado 3 y 6 veces el resultado 4. Tenéis que calcular el estimador máximo verosímil del parámetro λ de una variable de Poisson que haya generado estas observaciones (redondeado a 3 cifras decimales, y sin ceros innecesarios a la derecha), calcular el p-valor (redondeado a 3 cifras decimales, sin ceros innecesarios a la derecha) del test χ^2 para determinar si la muestra sigue

alguna distribución de Poisson habiendo estimado como su parámetro λ este valor, y decir (contestando SI o NO) si, con un nivel de significación $\alpha = 0.1$, tendríamos que rechazar la hipótesis nula de que esta muestra proviene de una variable aleatoria de Poisson. Dad las tres respuestas en este orden y separadas por un único espacio en blanco.

(4) Queremos contrastar si una cierta variable sigue una distribución normal. Hemos efectuado 150 observaciones y hemos obtenido 9 veces un valor dentro de $]0, 3]$, 27 veces un valor dentro de $]3, 6]$, 51 veces un valor dentro de $]6, 9]$, 46 veces un valor dentro de $]9, 12]$ y 17 veces un valor dentro de $]12, 15]$. Tenéis que: calcular los estimadores máximo verosímiles del parámetro μ y del parámetro σ de una variable normal que haya generado estas observaciones (ambos redondeados a 2 cifras decimales y sin ceros innecesarios a la derecha): calcular el p-valor (redondeado a 3 cifras decimales, sin ceros innecesarios a la derecha) del test χ^2 para contrastar si la muestra sigue *alguna* distribución normal, empleando estos valores estimados redondeados de los parámetros que habéis dado para especificar la distribución teórica; y decir (contestando SI o NO) si, con un nivel de significación $\alpha = 0.05$, tendríamos que rechazar la hipótesis nula de que esta muestra proviene de una variable aleatoria normal. Dad las cuatro respuestas en este orden y separadas por un único espacio en blanco.

(5) Generad una muestra aleatoria x de 25 valores de una distribución χ^2 con 10 grados de libertad, fijando antes `set.seed(2014)`, y aplicad el test de Kolmogorov-Smirnov para contrastar si x proviene de una distribución $N(10, 3.16)$. Dad el p-valor del test (redondeado a 3 cifras decimales, sin ceros innecesarios a la derecha) y decid (contestando SI o NO) si, con un nivel de significación $\alpha = 0.05$, tendríamos que rechazar la hipótesis nula de que esta muestra proviene de esta distribución. Tenéis que dar las dos respuestas en este orden y separadas por un único espacio en blanco.

(6) Queremos contrastar si la muestra siguiente sigue una distribución normal: 4.6, 0.97, 0.3, 1.11, 2.16, 15.52, 1.13, 0.17, 0.64, 2.00. Dad el p-valor (redondeado a 3 cifras decimales y sin ceros innecesarios a la derecha) del test de Kolmogorov-Smirnov-Lilliefors para esta muestra y decid (contestando SI o NO) si, con un nivel de significación $\alpha = 0.05$, tendríamos que rechazar la hipótesis nula de que esta muestra sigue una distribución normal. Tenéis que dar las dos respuestas en este orden y separadas por un único espacio en blanco.

(7) Generad una muestra aleatoria x de 15 valores de una distribución normal con $\mu = 2$ y $\sigma = 0.8$ fijando antes `set.seed(2014)`, y una muestra aleatoria y de 25 valores de una distribución exponencial de parámetro $1/\lambda = 0.5$ fijando antes `set.seed(1007)`. Aplicad el test de Kolmogorov-Smirnov para contrastar si x e y provienen de una misma distribución continua. Tenéis que dar el p-valor del test (redondeado a 3 cifras decimales, sin ceros innecesarios a la derecha) y decir (contestando SI o NO) si, con un nivel de significación $\alpha = 0.05$, tendríamos que rechazar la hipótesis nula de que estas dos muestras provienen de la misma distribución continua. Dad las dos respuestas en este orden y separadas por un único espacio en blanco.

Respuestas

(1) `2.116`

(2) `0.02 SI`

(3) `1.753 0.064 SI`

(4) `8.2 3.18 0.692 NO`

(5) `0.313 NO`

(6) `0 SI`

(7) `0.117 NO`