Entrega 4 y FINAL. Problemas y Talleres MATIII Estadística grado informática 2019-2020

Ricardo Alberich

13-05-2020

Contenidos

1 Entrega 4 Problemas: Estadística Inferencial 2

Contestad en GRUPOS del proyecto a los siguientes problemas y cuestiones en un fichero Rmd y su salida en html o pdf.

Cambien podéis incluir capturas de problemas hechos en papel. Cada pregunta vale lo mismo y se reparte la nota entre sus apartados.

1.1 Problema 1: Contraste de proporciones de dos muestras independientes.

Queremos comparar las proporciones de aciertos de dos redes neuronales que detectan tipos si una foto con un móvil de una avispa es una [avispa velutina o asiática] (https://es.wikipedia.org/wiki/Vespa_velutina). Esta avispa en una especie invasora y peligrosa por el veneno de su picadura. Para ello disponemos de una muestra de 1000 imágenes de insectos etiquetadas como avispa velutina y no velutina.

Aquí tenéis el acceso a los datos. Cada uno está en fichero los aciertos están codificados con 1 y los fallos con 0.

Se pide:

1. Cargad los datos desde el servidos y calcular el tamaño de las muestras y la proporción de aciertos de cada muestra.
2. Contrastad si hay evidencia de que las las proporciones de aciertos del algoritmo 1 son mayores que las del algoritmo 2. Definid bien las hipótesis y las condiciones del contraste. Tenéis que hacer el contraste con funciones de `R` y resolver el contrate con el p-valor.(**vale doble**)
3. Calculad e interpretar los intervalos de confianza para la diferencia de proporciones asociados al test anterior, con funciones de R.

1.1.1 Pre Solución

```
algoritmo1=read.table("http://bioinfo.uib.es/~recerca/MATIIIGINF/velutina/algoritmo1.csv")
algoritmo2=read.table("http://bioinfo.uib.es/~recerca/MATIIIGINF/velutina/algoritmo2.csv")
```

Proporción aciertos de cada algoritmo

```
n1=dim(algoritmo1)[1]
n1
```

```
## [1] 500
```

```
n1=length(algoritmo1$V1)
n1
```

```
## [1] 500
```

```
n2=length(algoritmo2$V1)
n2
```

```
## [1] 500
```

```
aciertos_absolutos_algorimo1=table(algoritmo1)["1"]
aciertos_absolutos_algorimo1
```

```
##   1
## 396
```

```
p1=prop.table(table(algoritmo1))["1"]
p1
```

```
##     1
## 0.792
```

```
aciertos_absolutos_algorimo2=table(algoritmo2)["1"]
aciertos_absolutos_algorimo2
```

```
##   1
## 437
```

```
p2=prop.table(table(algoritmo2))["1"]
p2
```

```
##     1
## 0.874
```

```r
x=matrix(c(aciertos_absolutos_algorimo1,n1-aciertos_absolutos_algorimo1,aciertos_absolutos_algorimo2,n2
x
```

```
##      [,1] [,2]
## [1,]  396  437
## [2,]  104   63
```

```r
fisher.test(x,alternative="greater",conf.level=0.95)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  x
## p-value = 0.9998
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  0.4056457       Inf
## sample estimates:
## odds ratio
##  0.5492712
```

```r
prop.test(c(aciertos_absolutos_algorimo1,aciertos_absolutos_algorimo2), c(n1,n2),alternative="greater",
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(aciertos_absolutos_algorimo1, aciertos_absolutos_algorimo2) out of c(n1, n2)
## X-squared = 11.502, df = 1, p-value = 0.9997
## alternative hypothesis: greater
## 95 percent confidence interval:
##  -0.1225654  1.0000000
## sample estimates:
## prop 1 prop 2
##  0.792  0.874
```

```r
fisher.test(x,alternative="greater",conf.level=0.95)$conf.int
```

```
## [1] 0.4056457       Inf
## attr(,"conf.level")
## [1] 0.95
```

```r
prop.test(c(aciertos_absolutos_algorimo1,aciertos_absolutos_algorimo2), c(n1,n2),alternative="greater",
```

```
## [1] -0.1225654  1.0000000
## attr(,"conf.level")
## [1] 0.95
```

1.2 Problema 2 : Contraste de proporciones de dos muestras emparejadas.

En el problema anterior hemos decidido quedarnos con el mejor de los algoritmos y mejorarlo. Pasamos las mismas 1000 imágenes a la version_beta del algoritmo y a la version_alpha. Aquí tenéis el acceso a los datos en el mismo orden para las 1000 imágenes. Cada uno está en fichero los aciertos están codificados con 1 y los fallos con 0.

1. Cargad los datos desde el servidos y calcular el tamaño de las muestras y la proporción de aciertos de cada muestra.
2. Contrastad si hay evidencia de que las las proporciones de aciertos del algoritmoalfa son iguales que las del algoritmo beta. Definid bien las hipótesis y las condiciones del contraste. Tenéis que hacer el contraste con funciones de `R` y resolver el contrate con el p-valor.

1.2.1 Pre Solución

```
algoritmoalfa=read.table("http://bioinfo.uib.es/~recerca/MATIIIGINF/velutina2/algoritmo_alpha.csv")
algoritmobeta=read.table("http://bioinfo.uib.es/~recerca/MATIIIGINF/velutina2/algoritmo_beta.csv")
```

```
X=table(algoritmoalfa$V1,algoritmobeta$V1)
X
```

```
##
##       0   1
##   0  15 110
##   1  88 787
```

```
mcnemar.test(X)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  X
## McNemar's chi-squared = 2.2273, df = 1, p-value = 0.1356
```

1.3 Problema 3 : ANOVA comparación media puntuaciones según fabricante.

Una vez mejorado nuestro algoritmo queremos saber su comportamiento bajo distintos tipos de móviles.

Seleccionamos 6 móviles de la misma gama de calidad de 6 fabricantes distintos. A los fabricantes los denotamos por F1, F2, F3, F4, F5 y F6.

Vamos a jugar no con la clasificación sino con el score que produce el algoritmo. Para ello seleccionamos 4 muestra aleatorias de fotos de insectos enviadas por los usuarios y la puntuación (*score*) que nos da el algoritmo que es una variable aleatoria continua de con rango de 0 a 100.

La idea es comprobar si la media de las puntuaciones del algoritmo es la misma para cada uno de los fabricantes.

Los datos los podéis descargar de esta dirección del servidor bioinfo.uib.es.

Antes de descargarlo, visualizar el fichero desde el navegador, para saber cómo descargarlo.

1. ¿Podemos asegurar que la muestras son normales en cada grupo? ¿y son homocedásticas? Justificar la respuesta con el correspondiente código en R comentado.
2. Escribid formalmente la hipótesis nula y la alternativa. Calcular la tabla de ANOVA y resuelve el test de forma manual.
3. Calcular la tabla de ANOVA y resuelve el test con la función `aov` de R.
4. Haced una comparación de pares con la función adecuada de R para la corrección del holm al nivel de significación $\alpha = 0.1$. Interpreta el resultado.
5. Comparar por grupos con el test de Duncan del paquete `agricolae`. Interpreta el resultado.

1.4 Pre solución

```
df=read.table("http://bioinfo.uib.es/~recerca/MATIIIGINF/anova_score/score_manufacturer.csv")
head(df)
```

```
##      score manufacturer
## 1 69.32030           F1
## 2 66.93433           F1
## 3 67.70541           F1
## 4 63.47195           F1
## 5 65.58738           F1
## 6 65.47437           F1
```

```
df$manufacturer=as.factor(df$manufacturer)
```

```
table(df$manufacturer)
```

```
##
##  F1  F2  F3  F4  F5  F6
## 100 100 100 100 100 100
```

```
library(nortest)
lillie.test(df$score[df$manufacturer=="F1"])
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  df$score[df$manufacturer == "F1"]
## D = 0.091505, p-value = 0.03825
```

```
for(Fabricante in levels(df$manufacturer)){
print(lillie.test(df$score[df$manufacturer==Fabricante]))
}
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  df$score[df$manufacturer == Fabricante]
## D = 0.091505, p-value = 0.03825
##
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  df$score[df$manufacturer == Fabricante]
## D = 0.067758, p-value = 0.3121
##
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  df$score[df$manufacturer == Fabricante]
## D = 0.069567, p-value = 0.2744
##
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  df$score[df$manufacturer == Fabricante]
## D = 0.069567, p-value = 0.2744
##
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  df$score[df$manufacturer == Fabricante]
## D = 0.069567, p-value = 0.2744
##
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  df$score[df$manufacturer == Fabricante]
## D = 0.10632, p-value = 0.007255
```

```
sapply(levels(df$manufacturer), FUN=function(x) {lillie.test(df$score[df$manufacturer==x])})
```

```
##           F1
## statistic 0.09150477
## p.value   0.03825059
## method    "Lilliefors (Kolmogorov-Smirnov) normality test"
## data.name "df$score[df$manufacturer == x]"
##           F2
## statistic 0.06775771
## p.value   0.3121052
## method    "Lilliefors (Kolmogorov-Smirnov) normality test"
## data.name "df$score[df$manufacturer == x]"
##           F3
## statistic 0.06956719
## p.value   0.2743622
## method    "Lilliefors (Kolmogorov-Smirnov) normality test"
## data.name "df$score[df$manufacturer == x]"
##           F4
## statistic 0.06956719
## p.value   0.2743622
## method    "Lilliefors (Kolmogorov-Smirnov) normality test"
## data.name "df$score[df$manufacturer == x]"
##           F5
## statistic 0.06956719
```

```
## p.value   0.2743622
## method    "Lilliefors (Kolmogorov-Smirnov) normality test"
## data.name "df$score[df$manufacturer == x]"
##           F6
## statistic 0.1063201
## p.value   0.007255259
## method    "Lilliefors (Kolmogorov-Smirnov) normality test"
## data.name "df$score[df$manufacturer == x]"
```

```
summary(aov(df$score~df$manufacturer))
```

```
##                  Df Sum Sq Mean Sq F value              Pr(>F)
## df$manufacturer   5   9143  1828.5   17.54 0.000000000000000317 ***
## Residuals       594  61910   104.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pairwise.t.test(df$score,df$manufacturer,p.adjust.method = "holm")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  df$score and df$manufacturer
##
##    F1      F2            F3            F4      F5
## F2 1.00000 -             -             -       -
## F3 0.07729 0.44250       -             -       -
## F4 0.00011 0.00000297115 0.00000000017 -       -
## F5 0.00011 0.00000297115 0.00000000017 1.00000 -
## F6 0.00724 0.00040       0.00000014687 1.00000 1.00000
##
## P value adjustment method: holm
```

```
library(agricolae)
```

```
## Warning: package 'agricolae' was built under R version 3.6.3
```

```
resultado.anova=aov(df$score~df$manufacturer)
duncan.test(resultado.anova,"df$manufacturer",group=TRUE)$group
```

```
##    df$score groups
## F3 74.07499      a
## F2 71.61166     ab
## F1 70.47337      b
## F6 65.71299      c
## F4 64.07499      c
## F5 64.07499      c
```

1.5 Problema 4: Regresión lineal simple.

Consideremos los siguientes datos

```
x=c(-2,-1,2,0,1,2)
y=c(-7, -5,  5, -3,  3.0,  4)
summary(lm(y~x))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      1      2      3      4      5      6
##  0.675 -0.400  0.375 -1.475  1.450 -0.625
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5250     0.4872  -3.130 0.035176 *
## x             3.0750     0.3189   9.642 0.000647 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.165 on 4 degrees of freedom
## Multiple R-squared:  0.9587, Adjusted R-squared:  0.9484
## F-statistic: 92.96 on 1 and 4 DF,  p-value: 0.0006472
```

1. Calcular manualmente los coeficiente de la regresión lineal de y sobre x
2. Calcular los valores $\hat{y}_i = b_0 + b_1 \cdot x_1$ para los valores de la muestra y el error cometido.
3. Calcular la estimación de la varianza del error.
4. Resolver manualmente el contraste $\left\{\begin{array}{ll} H_0: & \beta_1 = 0 \\ H_1: & \beta_1 \neq 0 \end{array}\right.$, calculando el p-valor.
5. Calcular SST, SSR y SSE.
6. Calcular el coeficiente de regresión lineal r_{xy} y el coeficiente de determinación R^2. Interpretad el resultado en términos de la cantidad de varianza explicada por el modelo
7. Comprobar que los resultados son los mismos que los obtenidos con la función `summary(lm(y~x))`.

1.5.1 Pre solución

```
x=c(-2,-1,2,0,1,2)
y=c(-7, -5,  5, -3,  3.0,  4)
sol_lm=(lm(y~x))
summary(sol_lm)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      1      2      3      4      5      6
##  0.675 -0.400  0.375 -1.475  1.450 -0.625
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5250     0.4872  -3.130 0.035176 *
## x             3.0750     0.3189   9.642 0.000647 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.165 on 4 degrees of freedom
## Multiple R-squared:  0.9587, Adjusted R-squared:  0.9484
## F-statistic: 92.96 on 1 and 4 DF,  p-value: 0.0006472
```

```
mediay=mean(y)
mediax=mean(x)
sdx=sd(x)
sdy=sd(y)
sxy=cov(x,y)
b1=sxy/sdx^2
b1
```

```
## [1] 3.075
```

```
b0=mediay-b1*mediax
b0
```

```
## [1] -1.525
```

```
c(b0,b1)==sol_lm$coefficients# dan distintos errores de redondeo
```

```
## (Intercept)           x
##       FALSE        TRUE
```

```
near(c(b0,b1),sol_lm$coefficients)# opcional
```

```
## (Intercept)           x
##        TRUE        TRUE
```

```
recta=function(x) b0+b1*x
y_est=recta(x)
predict(sol_lm,newdata = data.frame(x))
```

```
##      1      2      3      4      5      6
## -7.675 -4.600  4.625 -1.525  1.550  4.625
```

```
e=y-y_est
e
```

```
## [1]  0.675 -0.400  0.375 -1.475  1.450 -0.625
```

```r
sol_lm$residuals
```

```
##      1      2      3      4      5      6
##  0.675 -0.400  0.375 -1.475  1.450 -0.625
```

```r
mean(e) # es cero, pero por eerror de redondeo no da exacto.
```

```
## [1] -0.0000000000000002220446
```

```r
SSE=sum(e^2)
SSE
```

```
## [1] 5.425
```

```r
n=length(x)
n
```

```
## [1] 6
```

```r
S2=SSE/(n-2)#estimacion_var_error
S2
```

```
## [1] 1.35625
```

```r
S=sqrt(S2)# Residual standard error: 1.165
S
```

```
## [1] 1.164581
```

```r
 # contraste beta1=0
t0=b1/(S/(sdx*sqrt(n-1)))
t0
```

```
## [1] 9.6415
```

```r
2*pt(abs(t0),n-2,lower.tail = FALSE)
```

```
## [1] 0.0006472191
```

```r
2*(1-pt(abs(t0),n-2,lower.tail = TRUE))
```

```
## [1] 0.0006472191
```

```r
2*(1-pt(abs(t0),n-2))
```

```
## [1] 0.0006472191
```

comparar con

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.5250     0.4872  -3.130 0.035176 *
x             3.0750     0.3189   9.642 0.000647 ***
```

```r
SST=sum((y-mean(y))^2)
SST
```

```
## [1] 131.5
```

```r
mean(y_est)==mean(y)#media estimados regresion igual a media y
```

```
## [1] FALSE
```

```r
SSR=sum((y_est-mean(y))^2)
SSR
```

```
## [1] 126.075
```

```r
SSE# ya lo había a calculado
```

```
## [1] 5.425
```

```r
SST-SSR# da lo mismo pues SST=SSR+SSE
```

```
## [1] 5.425
```

```r
R2=SSR/SST
R2
```

```
## [1] 0.9587452
```

```r
cor(x,y)
```

```
## [1] 0.9791554
```

```r
cor(x,y)^2# en el caso regre simp`le R2=cor(xy)^2
```

```
## [1] 0.9587452
```

1.6 Problema 5: Distribución de los grados de un grafo de contactos.

En el artículo de A. Broder et al., Graph structure in the Web. Computer Networks 33, 309 (2000).

Se recopiló el número de enlaces a sitios web encontrados en un rastreo web de 1997 de aproximadamente 200 millones de páginas web,

Con el se construyó una tabla con la frecuencia de sitios por número de enlaces. El código siguiente carga del enlace que han puesto los autores del artículo

```r
data_links=read.table("http://tuvalu.santafe.edu/~aaronc/powerlaws/data/weblinks.hist",header=TRUE)
head(data_links)
```

```
##   degree frequency
## 1      0  35159835
## 2      1 106649769
## 3      2  40711748
## 4      3  22648832
## 5      4  12617832
## 6      5   8188854
```

```r
str(data_links)
```

```
## 'data.frame':    14480 obs. of  2 variables:
##  $ degree   : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ frequency: int  35159835 106649769 40711748 22648832 12617832 8188854 6438634 4690068 4954649 3731
```

```r
# eliminamos la páginas con menos de 8 enlaces  enlaces y las de más de 1000 enlaces
data_links_central=data_links[data_links$degree>8&data_links$degree<10^3,]
head(data_links_central)
```

```
##    degree frequency
## 10      9   3731928
## 11     10   3036333
## 12     11   2496648
## 13     12   2119312
## 14     13   1790068
## 15     14   1546579
```

```r
tail(data_links_central)
```

```
##      degree frequency
## 995     994       213
## 996     995       193
## 997     996       157
## 998     997       137
## 999     998       178
## 1000    999       153
```

El siguiente código calcula las regresiones exponecial, potencial y lineal (en algún orden) de las frecuencias (`frequency`) contra los enlaces (`degree`).

```r
sol1=lm(frequency~ degree,data=data_links_central)
summary(sol1)
```

```
##
## Call:
## lm(formula = frequency ~ degree, data = data_links_central)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
##  -96861  -69548  -25033   22374 3598744
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 134974.49   13778.17   9.796 <0.0000000000000002 ***
## degree        -198.98      23.77  -8.369 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 214100 on 989 degrees of freedom
## Multiple R-squared:  0.06614,    Adjusted R-squared:  0.06519
## F-statistic: 70.04 on 1 and 989 DF,  p-value: < 0.00000000000000022
```

```
sol2=lm(log10(frequency)~ degree,data=data_links_central)
summary(sol2)
```

```
##
## Call:
## lm(formula = log10(frequency) ~ degree, data = data_links_central)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43758 -0.26558 -0.07671  0.16681  2.13097
##
## Coefficients:
##                Estimate  Std. Error t value            Pr(>|t|)
## (Intercept)  4.46504979  0.02381018  187.53 <0.0000000000000002 ***
## degree      -0.00267658  0.00004109  -65.15 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.37 on 989 degrees of freedom
## Multiple R-squared:  0.811,  Adjusted R-squared:  0.8108
## F-statistic:  4244 on 1 and 989 DF,  p-value: < 0.00000000000000022
```

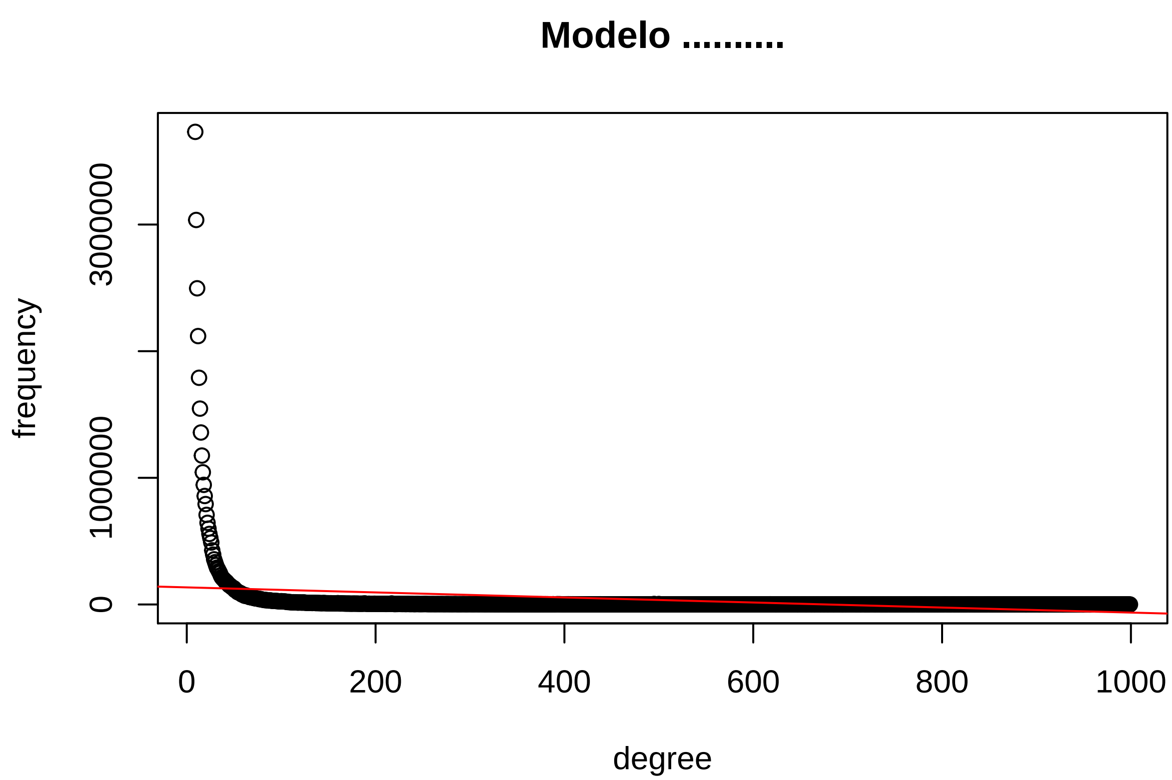
```
sol3=lm(log10(frequency)~ log10(degree),data=data_links_central)
summary(sol3)
```

```
##
## Call:
## lm(formula = log10(frequency) ~ log10(degree), data = data_links_central)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.21376 -0.04747 -0.01555  0.01958  0.73976
##
## Coefficients:
##                Estimate Std. Error t value            Pr(>|t|)
## (Intercept)    8.722036   0.020623   422.9 <0.0000000000000002 ***
## log10(degree) -2.170129   0.007894  -274.9 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

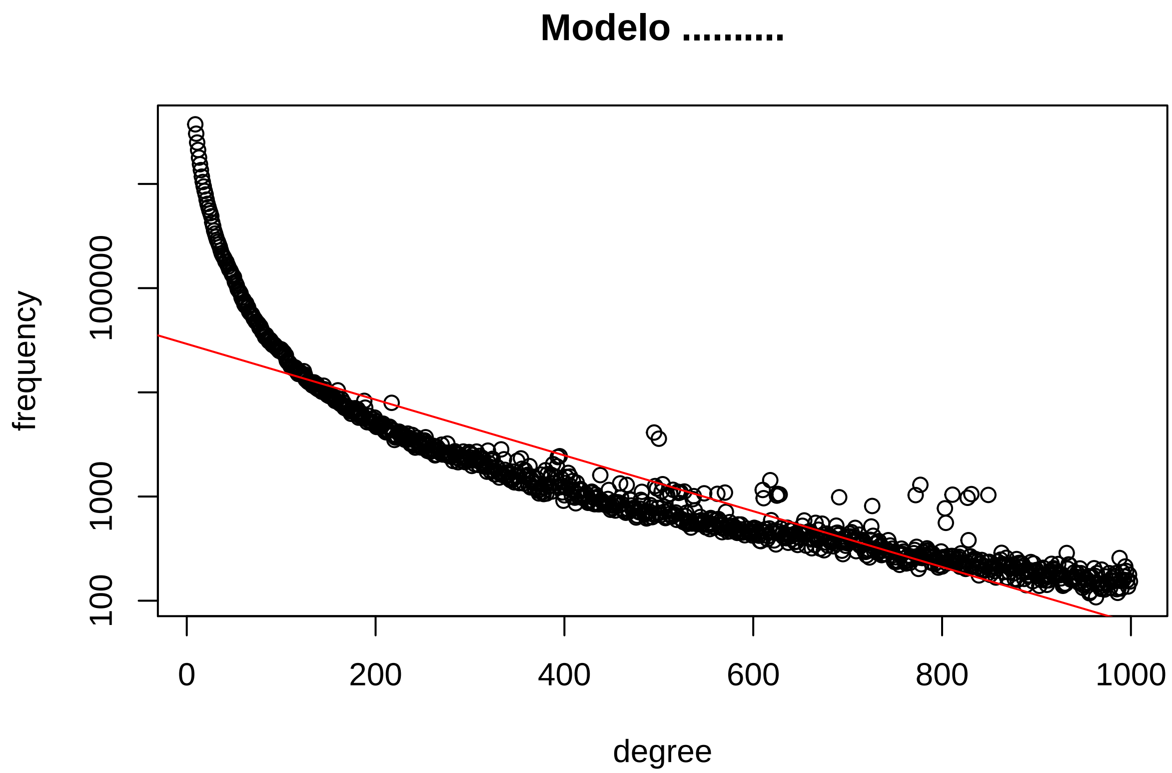
```
##
## Residual standard error: 0.09674 on 989 degrees of freedom
## Multiple R-squared:  0.9871, Adjusted R-squared:  0.9871
## F-statistic: 7.557e+04 on 1 and 989 DF,  p-value: < 0.00000000000000022
```

Ahora dibujamos los gráficos adecuados a cada modelo

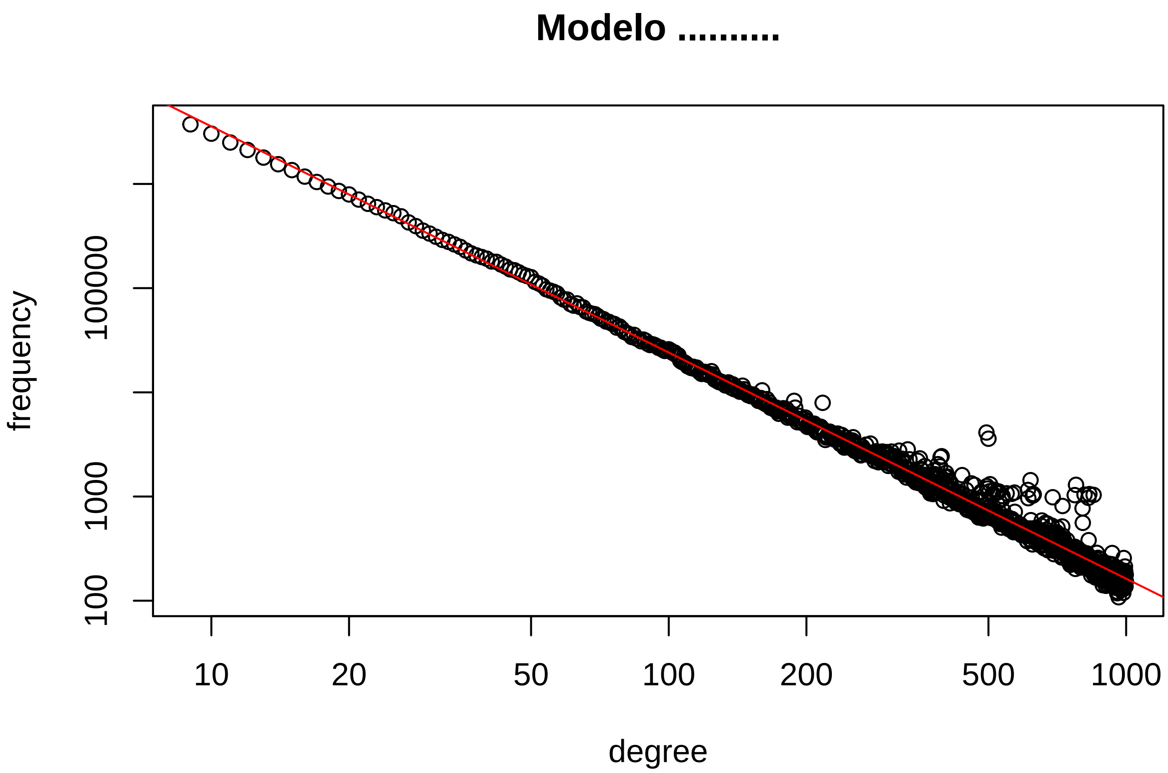
```r
plot(data_links_central,main="Modelo ..........")
abline(sol1,col="red")
```




```r
plot(data_links_central,main="Modelo ..........",log="y")
abline(sol2,col="red")
```




```r
plot(data_links_central,main="Modelo ..........",log="xy")
abline(sol3,col="red")
```



Se pide:

1. Explicad el modelo de regresión que calcula cada función `lm`
2. ¿Qué modelo y en función de qué parámetros es el mejor?
3. Para el mejor modelo calcular los coeficientes en las unidades originales y escribir la ecuación del modelos.