Tema 7:

Análisis estadísticos elementales

Recogemos algunos análisis estadísticos habituales que pueden servir de ejemplo de cómo realizar un análisis con R.

7.1. Funciones de densidad y distribución

```
Dibuja<-function(n=30,alfa=.001,fil=1,col=1)
{
#Dibuja la funcion de densidad y la de distribucion de la normal
n<-as.integer(n)
if (n<10) stop("Hacen falta mas numeros")
a<-qnorm(alfa)
b<-qnorm(1-alfa)
win.graph()
par(mfrow=c(fil,col))
for (i in 1:(fil*col))
{
x<-seq(a,b,,i*n)
plot(x,dnorm(x))
}
win.graph()
par(mfrow=c(fil,col))
for (i in 1:(fil*col))
{
x<-seq(a,b,,i*n)
plot(x,pnorm(x))
}
}
```

7.2. Estadística descriptiva

Los datos correspondientes al ejercicio 3.1 de se encuentran en el archivo ej3-1.txt. El análisis descriptivo de los mismos es el siguiente:

```
> read.table(file="c:/datos/ej3-1.txt",header=T)
Peso Altura Edad Sexo
1 75 175 21 H
2 81 178 22 H
3 56 162 22 M
4 68 180 21 M
5 79 182 24 H
6 89 185 22 H
7 62 157 21 M
8 59 165 22 M
9 83 180 23 H
10 55 160 22 M
11 72 174 21 H
12 56 161 23 M
```

13 84 182 22 H
14 64 163 24 M
15 76 172 22 H
16 68 169 21 M
17 58 162 24 H
18 78 178 22 H
19 62 157 21 M
20 77 165 21 H
21 71 182 22 M
22 55 163 24 M
23 78 174 25 H
24 77 181 23 H
25 81 181 22 H
26 75 174 23 H
27 60 167 21 M
28 68 178 21 H
29 68 182 22 M
30 89 185 22 H
31 82 190 21 H
32 59 155 22 M
33 63 173 21 M
34 75 177 24 H
35 62 174 21 M
36 81 183 22 H

```
> win.graph()
> Ej3.1<-read.table(file="c:/datos/ej3-1.txt",header=T)
> summary(Ej3.1)
```

```
      Peso Altura Edad Sexo
      Min. :55.00 Min. :155.0 Min. :21.00 H:20
      1st Qu.:62.00 1st Qu.:164.5 1st Qu.:21.00 M:16
      Median :71.50 Median :174.0 Median :22.00
      Mean :70.72 Mean :172.9 Mean :22.14
      3rd Qu.:78.25 3rd Qu.:181.0 3rd Qu.:23.00
      Max. :89.00 Max. :190.0 Max. :25.00
```

```
> par(mfrow=c(2,2))
> plot(Ej3.1)
> win.graph()
> pie(table(Ej3.1[,4]))
> win.graph()
> par(mfrow=c(1,3))
> for (i in 1:3) {boxplot(Ej3.1[,i]) ;title(names(Ej3.1)[i])} win.graph()
par(mfrow=c(1,2))
boxplot(Ej3.1[,1])
title(main=names(Ej3.1)[1], sub="Sin muescas")
boxplot(Ej3.1[,2],notch=T)
```

```
title(main=names(Ej3.1)[2], sub="Con muescas")
hist2d(Ej3.1[,1:2])
win.graph()
persp(hist2d(Ej3.1[,1:2]))
win.graph()
hist(Ej3.1[,1])
> var(Ej3.1[,1:3])
```

	Peso	Altura	Edad
Peso	103.4063492	77.6984127	0.4111111
Altura	77.6984127	88.5111111	-0.3349206
Edad	0.4111111	-0.3349206	1.2658730

```
> quantile(Ej3.1[,1])
0% 25% 50% 75% 100%
55 62 71.5 78.25 89

> quantile(Ej3.1[,2])
0% 25% 50% 75% 100%
155 164.5 174 181 190


> quantile(Ej3.1[,3])
0% 25% 50% 75% 100%
21 21 22 23 25


> win.graph();histogram(Ej3.1[,1])
> stem(Ej3.1[,1])

N = 36 Median = 71.5
Quartiles = 62, 78.5

Decimal point is 1 place to the right of the colon

5 : 5566899
6 : 022234
6 : 8888
7 : 12
7 : 555677889
8 : 111234
8 : 99

> stem(Ej3.1[,1],depth=T)

N = 36 Median = 71.5
Quartiles = 62, 78.5

Decimal point is 1 place to the right of the colon
```

```
7 7 5 : 5566899
13 6 6 : 022234
17 4 6 : 8888
2 7 : 12
17 9 7 : 555677889
8 6 8 : 111234
2 2 8 : 99
```

> stem(Ej3.1[,1:3],depth=T)

Lo hace para los tres.

Sería mejor añadir la hoja de datos a la lista de búsqueda

> attach(Ej3.1)
> stem(Peso)

N = 36 Median = 71.5
Quartiles = 62, 78.5

Decimal point is 1 place to the right of the colon

```
5 : 5566899
6 : 022234
6 : 8888
7 : 12
7 : 555677889
8 : 111234
```

7.2.1. density

Esta función realiza una estimación no paramétrica de la función de densidad

> hist(Peso)
> (density(Peso,n=20))

```
$x:
[1] 52.93352 54.94052 56.94752 58.95452
[5] 60.96151 62.96851 64.97551 66.98251
[9] 68.98950 70.99650 73.00350 75.01050
[13] 77.01749 79.02449 81.03149 83.03849
[17] 85.04548 87.05248 89.05948 91.06648

$y:
[1] 0.0003583734 0.0420238934 0.0185183659
[4] 0.0434627049 0.0223812610 0.0393774733
[7] 0.0061812084 0.0216712467 0.0232208651
[10] 0.0217112936 0.0065417807 0.0546300411
[13] 0.0502519682 0.0280147661 0.0548217371
[16] 0.0279770494 0.0052941968 0.0005927378
[19] 0.0321395881 0.0003583734
```

```
> lines(density(Peso,n=20)$x,density(Peso,n=20)$y*length(Peso))
> win.graph()
> hist(Peso,prob=T)
> lines(density(Peso,n=20))
> lines(density(Peso,n=50))
```

7.3. Tablas de contingencia

Podemos crear tablas de contingencia con las funciones **table** y **crosstabs**. Es posible realizar acciones más complejas que el simple conteo de frecuencias con la función **tapply**. Si es necesario, una variable puede agruparse por intervalos con la función **cut**. Las tablas creadas son variables multiindexadas con tantos índices como argumentos se den a la función.

La sintaxis de **table** es

```
table(..., exclude=c(NA,NaN))
```

donde los argumentos son cualquier número de objetos, todos de la misma longitud, cada uno de los cuales se interpreta como una variable categórica. Además, **exclude** es un vector de objetos que deben eliminarse cuando se creen las catgorías a partir de los objetos que no lo son.

Por su parte, **crosstabs** crea la tabla a partir de un conjunto de factores. Su sintaxis es

```
crosstabs(formula, data=sys.parent(), margin=,
subset, na.action=na.fail, drop.unused.levels=T)
```

- **formula** un objeto **formula** . Cada término de la parte derecha de la fórmula debe ser un factor y en caso contrario será convertido a este tipo. Si hay término en la parte izquierda debe ser un vector de conteos, procedente posiblemente de una tabla anterior. Cuando el argumento **data** sea una hoja de datos, puede omitirse la fórmula y se realizarán todas las tablas cruzadas en la hoja de datos.
- **data** es una hoja de datos que contiene las variables utilizadas en la fórmula, en caso de que no estén en la lista de búsqueda.
- **subset** subconjunto de individuos a los que debe limitarse la construcción de la tabla. Si no se indica, se incluyen todos.
- **margin** lista de vectores de enteros que indican qué marginales deben calcularse. El valor predeterminado en el caso bidimensional es **list ("Row%"=1, "Col%"=2, "Total%"=integer(0))**
- **na.action** es una función que indica qué hacer cuando hay datos faltantes. El valor predeterminado es la función **na.fail** que detiene el proceso. También son de uso común **na.omit**, que los elimina, y **na.include**. que añade a cada factor el nivel NA en las tabulaciones, aunque puede definirse cualquier otra.
- **drop.unused.levels** valor lógico que indica si los niveles que no tienen observaciones deben eliminarse o incluirse en la tabla. En el último caso habrá líneas completas de ceros y, por tanto, los estadísticos de Pearson valdrán NA.

En análisis más complejos, puede utilizarse la función **tapply** que aplica una función a cada

uno de los elementos de una matriz que cumplen la condición de tener los mismos niveles en diferentes ategorías. La sintaxis es

tapply(X, INDICES, FUN=, ..., simplify=T)

- **X** es un vector de datos que serán agrupados mediante índices.
- **INDICES** es una lista cuyos componentes se interpretan como variables categóricas, de la misma longitud que X, que definen la posición en una variable multiindexada de la observación
  concreta y que, salvo que se utilice **simplify**, define el resultado.
- **FUN** es una función (o el nombre una función) que debe aplicarse a cada celdilla. Si no se indica, se devuelve un vector que puede usarse para indexar la variable que se produce.
- **...** son argumentos opcionales para la función.
- **simplify** valor lógico que indica si debe devolverse un vector (T) o una variable multiindexada (F).

La función **cut** crea una variable categórica a partir de una numérica, dividiendo en intervalos, bien mediante puntos de corte, bien indicando el total de intervalos a obtener. La sintaxis es

cut(x, breaks, labels=<<see below>>, include.lowest = F)

- **x** vector de datos.
- **breaks** bien un vector de puntos de corte (ordenados), bien el número de clases.
- **labels** vector de etiquetas de los intervalos. El valor predeterminado es inferior + thru superior.
- **include.lowest** valor lógico que indica si cada intervalo es cerrado por la izquierda (T) o por la derecha (F). El valor predeterminado es F.

la función devuelve un vector de igual longitud que el de entrada pero que es categórico. Los valores inferiores al primer punto de corte o superiores al último se consideran NA.

7.4. Modelos loglineales

La función **loglin** realiza el análisis loglineal de tablas de contingencia. La sintaxis es

loglin(table, margin, start=, fit=F, eps=0.1, iter=20, param=F, print=T)

- **table** es la tabla de contingencia a la que se desea ajustar el modelo.
- **margin** es una lista de vectores que describen las marginales a ajustar.
- **start** estimación inicial para la tabla ajustada. Si aparecen ceros estructurales en la tabla, deben incluirse en los lugares correspondientes de este parámetro y deben ponerse unos en el resto de sitios.
- **fit** valor lógico que indica si el ajuste estimado debe ser devuelto.
- **eps** valor máximo permitido para una diferencia entre un valor observado y uno ajustado.

- **iter** número máximo de iteraciones.
- **param** valor lógico que indica si deben devolverse los valores de los parámetros.
- **print** valor lógico que indica si deben devolverse la desviación final y el número de iteraciones.

La función devuelve una lista con los siguientes componentes:

- **lrt** estadístico del cociente de verosimilitudes.
- **pearson** estadístico de Pearson.
- **df** grados de libertad. No se hace ajuste por la existencia de ceros.
- **margin** lista de las marginales ajustadas.
- **fit** variable multidimensional con los mismos índices que **table** y que contiene los valores ajustados.
- **param** parámetros estimados el modelo.

Debe tenerse en cuenta que un término incluye todos los términos de orden inferior.

7.5. El problema de las dos muestras

La comparación de medias de dos poblaciones normales, para dos muestras independientes o relacionadas (apareadas), se obtiene haciendo uso de la función **t.test** del package **stats**.

La sintaxis de la función es:

t.test (x, y = NULL, alternative = c("two.sided", "less", "greater"), mu = 0, pai
= FALSE, var.equal = FALSE, conf.level = 0.95, ...)

cuyos argumentos son:

- **x** vector numérico conteniendo los datos de la muestra
- **y** vector numérico opcional (test de dos muestras) conteniendo los datos de la segunda muestra
- **alternative** especifica un test bilateral, "**two.sided**" (defecto), o unilateral, "**greater**" o "**less**"
- **mu** valor de la media (o diferencia de medias) contrastado
- **paired** TRUE o FALSE indicando si se trata de muestras relacionadas
- **var.equal** TRUE o FALSE indicando si las varianzas son iguales
- **conf.level** nivel de confianza del intervalo

Ejemplos.

```
h.datos2[h.datos2[,5]=="M",2]
[1] 77 58 55 47 60 54 65
h.datos2[h.datos2[,5]=="M",2]
[1] 77 58 55 47 60 54 65
h.datos2[h.datos2[,5]=="H",2]
```

```
[1] 89 58 75 65 82 85 75

Peso.Mujer<-h.datos2[h.datos2[,5]=="M",2]

Peso.Hombre<-h.datos2[h.datos2[,5]=="H",2]

Peso.Mujer
[1] 77 58 55 47 60 54 65
Peso.Hombre
[1] 89 58 75 65 82 85 75

t.test(Peso.Hombre)

One-sample t-Test

data: Peso.Hombre
t = 18.1055, df = 6, p-value = 0
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
65.35817 85.78469
sample estimates:
mean of x
75.57143

t.test(Peso.Hombre)[[1]]
t
18.10554
t.test(Peso.Hombre)[[3]]
[1] 1.827059e-006
t.test(Peso.Hombre,Peso.Mujer)

Standard Two-Sample t-Test

data: Peso.Hombre and Peso.Mujer
t = 2.9271, df = 12, p-value = 0.0127
alternative hypothesis: true difference in means
is not equal to 0
95 percent confidence interval:
4.126756 28.158959
sample estimates:
mean of x mean of y
75.57143 59.42857
t.test(Peso.Hombre,Peso.Mujer,var.equal=F)

Welch Modified Two-Sample t-Test

data: Peso.Hombre and Peso.Mujer
t = 2.9271, df = 11.751, p-value = 0.0129
alternative hypothesis: true difference in means
is not equal to 0
95 percent confidence interval:
```

4.09844 28.18727
sample estimates:
mean of x mean of y
75.57143 59.42857

La función **var.test** dentro del paquete **stats**, permite la comparación de las varianzas de dos muestras de poblaciones normales. La hipótesis nula es que el cociente de varianza de las poblaciones desde las que se tomaron las muestras es igual al valor del argumento **ratio**.

La sintaxis de la función es

var.test(x, y, ratio = 1, alternative = c("two.sided", "less", "greater"), conf.leve 0.95, ...)

con argumentos

- **x, y** vectores numéricos de datos.
- **ratio** el valor contrastado para el cociente de varianzas (x entre y).
- **alternative** una cada de caracteres especificando la hipótesis alternativa. El valor debe ser uno de "two.sided" (defecto), "greater" o "less". Se puede escribir sólo la primera letra de la cadena.
- **conf.level** nivel de confianza para el intervalo calculado.

Para ver más detalles se puede escribir **help(var.test)**.

Ejemplos.

var.test(Peso.Hombre,Peso.Mujer)

F test for variance equality

data: Peso.Hombre and Peso.Mujer
F = 1.3408, num df = 6, denom df = 6, p-value
= 0.7308
alternative hypothesis: true ratio of variances
is not equal to 1
95 percent confidence interval:
0.2303941 7.8033490
sample estimates:
variance of x variance of y
121.9524 90.95238
var.test(Peso.Hombre,Peso.Mujer)[[3]]
[1] 0.730824

7.6. Regresión

La función **lm** se usa para ajustar modelos lineales. Se puede utilizar para realizar un estudio de regresión, un análisis de la varianza y la covarianza (aunque para esto último es más conveniente el uso de otras funciones como **aov.**

La sintaxis de la función es

**lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x
FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)**

cuyos argumentos principales son:

- **formula** es un objeto de la clase "formula", consiste en una descripción simbólica del modelo ajustado.
- **data** es un marco de datos (data frame) o lista opcional almacenando las variables del modelo. Si no se especifica las variables se toman del objeto **formula**.
- **subset** es un vector opcional que permite especificar un subconjunto de observaciones para el ajuste.
- **weights** permite especificar un vector numérico con pesos para un ajuste por mínimos cuadrados ponderados.

Ejemplos.

```
plot(h.datos2[,"Altura"],h.datos2[,"Peso"])

plot(h.datos2[,"Altura"],h.datos2[,"Peso"],xlab="Altura")

plot(h.datos2[,"Altura"],h.datos2[,"Peso"],\verb|xlab="Altura",ylab="Peso")

lm(Altura~Peso,h.datos2)

Call:
lm(formula = Altura ~ Peso, data = h.datos2)

Coefficients:
(Intercept) Peso
1.341264 0.005293446

Degrees of freedom: 14 total; 12 residual
Residual standard error: 0.04745207

sexo<-factor(h.datos2[,"Sexo"])

sexo

[1] M M H M M M M H H H H H H M|

codes(sexo)

[1] 2 2 1 2 2 2 2 1 1 1 1 1 1 2

lm(Peso~Altura,h.datos2)
Call:
lm(formula = Peso ~ Altura, data = h.datos2)
```

Coefficients:
(Intercept) Altura
-155.2693 131.151

Degrees of freedom: 14 total; 12 residual
Residual standard error: 7.469159

regresion<- lm(Peso~Altura,h.datos2)

regresion
Call:
lm(formula = Peso ~ Altura, data = h.datos2)

Coefficients:
(Intercept) Altura
-155.2693 131.151

Degrees of freedom: 14 total; 12 residual
Residual standard error: 7.469159

summary(regresion)
Call: lm(formula = Peso ~ Altura, data = h.datos2)
Residuals:
Min 1Q Median 3Q Max
-13.69 -3.019 -0.1219 1.96 18.49

Coefficients:
Value Std. Error t value Pr(>\vert t\vert)
(Intercept) -155.2693 42.7242 -3.6342 0.0034
Altura 131.1510 25.1256 5.2198 0.0002

Residual standard error: 7.469 on 12
degrees of freedom
Multiple R-Squared: 0.6942
F-statistic: 27.25 on 1 and 12 degrees of
freedom, the p-value is 0.0002148

Correlation of Coefficients:
(Intercept)
Altura -0.9989
residuals(regresion)
1 2 3 4 5 6
18.49321 -0.5067895 1.639994 -2.19528 -7.57226 1.49321
7 8 9 10 11
-13.68736 -3.129809 -3.179437 -2.687359 5.132072
12 13 14
0.2630133 2.066602 3.870191

Se puuede hacer un gráfico identificando los hombres y las mujeres

Mujer<-h.datos2[,5]=="M"

```
plot(h.datos2[,Altura],h.datos2[,Peso])
points(h.datos2[Mujer,Altura],h.datos2[Mujer,Peso],type="p",pch=1)
points(h.datos2[!Mujer,Altura],h.datos2[!Mujer,Peso], type="p",pch=5)

predict(regresion)
1 2 3 4 5 6
58.50679 58.50679 87.36001 57.19528 54.57226 58.50679
7 8 9 10 11 12
67.68736 61.12981 78.17944 67.68736 76.86793 84.73699
13 14
72.9334 61.12981

plot(predict(regresion),h.datos2[,"Peso"])

abline(h=0,lty=2)

abline(h=83,lty=2)

abline(h=0,lty=2)

?abline

abline(0,1)

regresion$coef
(Intercept) Altura
-155.2693 131.15
```