Introducción al lenguaje de programación R

N.Ibáñez-Escriche

Genètica i Millora Animal. IRTA-Lleida

Valencia-2009

Outline

1 Introducción al R
- Introducción

2 Números y Vectores

3 Arrays y matrices

4 Listas y data frames

5 Lectura de datos desde fichero

6 Programación en R

7 Estadística básica y gráficos

Software

- Es gratuito
- R esta disponible para Windows, Linux y Unix
- R posee muchas funciones para tratamiento de datos, análisis estadísticos y gráficos

Características principales

- Flexibilidad
- Lenguaje “Orientado a Objetos”
- No se compila
- Se ejecuta mediante envío de instrucciones en la línea de comandos

Instalación de R

- Ir a la página web http://cran.r-project.org
- Download
- Escoger servidor
- Bajar programa para windows
- Ejecutar

Funciones de R

- Se agrupan por paquetes
- Las Funciones más habituales se incluyen por defecto en R
- El comando search() muestra las librerias cargadas
- Las librerias se pueden añadir fácilmente
  package install $\longrightarrow$ select cran $\longrightarrow$ install package
  comando library("package")
- Comando library() muestra librerias disponibles para ser cargadas
- Comando ls(número) muestra las funciones del paquete

Ayuda en R

- R dispone de ayuda en línea
- ¿Como utilizarla?
  - Manera clásica:
    help(lm)o ?lm
  - Consulta caracteres no-convencionales (entre comillas):
    help("for") o ?"for"
  - Ayuda en formato html:
    help(start)
  - Busqueda de términos:
    help.search("anova")
  - Mostrar ejemplos:
    example(quantile)

Primeras nociones

Creación de objetos

- R permite expresiones y asignaciones
- Expresión se muestra en pantalla pero no se guarda ejem: (20+5)*40
- Asignación, se realiza mediante $<-$ o $->$
  a<-5
  3->b
- Distingue entre mayúsculas y minúsculas
  N<-2
  n<-5
- También se pueden asignar carácteres
- nombre<-"Blasco"
- Se pueden escribir varios comandos en una misma línea
- n<-100; N<-5

Primeras nociones

Listado de objetos

- La funcion ls() lista los objetos en memoria
- Se pueden listar objetos con algún caracteristica ls(pat=n)
- La función rm() borra objetos de la memoria rm(n)
- Borra todos los objetos de la memoria rm(list=ls())

Interacción archivos externos

Cambiar directorio

- R busca los archivos en el directorio de trabajo
- Por defecto el directorio de trabajo de R está en la carpeta del programa R
- El directorio de trabajo puede ser cambiado:
  menú File $\longrightarrow$ Change dir ...
- O por código:
  setwd("c:/mis documentos/curso R")
  setwd("c:\\mis documentos\\curso R")
- Borra todos los objetos de la memoria

Interacción archivos externos

Ejecutar

- Guarda workspace modificado
  save.image("workspace1") (guarda objetos no librerias)
- Ejecutar comandos desde un archivo de texto
  source("ejer1.R")
- Guardar el resultado de nuestros comandos
- Incio volcado
  sink("result1")
- Comandos
  a<-1; b<-156*a
- Fin del volcado
  sink()

Ejercicios 1

1. Cargar la librería “datasets” o “nlme”. Mirar que funciones tiene y para que sirven. Usar el help para alguna de sus funciones.
2. Realizar asignaciones sencillas como A<-5: N<-6; a<-A*N/2. Comprobar los objetos que hay actualmente en R. Borrar un el objeto A y volver a comprobar los objetos.
3. Crear un archivo “Ejercicio1.R” y copiar los comandos nombrados anteriormente. Cambiar el directorio de trabajo de R al directorio donde se encuentra el archivo “Ejercicio1.R". Ejecutar los comandos mediante el archivo de texto.
4. Guardar directamente el resultado de las expresiones 50*30 y log(20) en un archivo llamado “result”. Después, anular el envío de la salida al fichero.
5. Guardar el workspace con el nombre “ejer1”. Cerrar R y cargarlo de nuevo mediante el menú o load.

Outline

1 Introducción al R

2 Números y Vectores
- Objetos: Vectores numéricos
- Vectores lógicos
- Valores missing
- Vectores de caracteres
- Vectores índices

3 Arrays y matrices

4 Listas y data frames

5 Lectura de datos desde fichero

Objetos: Vectores numéricos

- R trabaja con objetos. El vector es un objeto básico en R.
- Los vectores tienen dos argumentos el “mode” (modo) y “length” (longitud).
- Los vectores pueden ser de modo numérico, lógico y caracter.
- El vector numérico es la estructura más simple de datos.

Ejemplos

a<-5

a es un vector de longitud 1

Una manera de asignar valores a un vector es mediante la función y<-c(4,5.6,7,8.2,4) (y es un vector numérico de longitud 5)

Otras formas menos utlizadas de asignar valores son:
assign(“y”,c(4,5.6,7,8.2,4))
c(4,5.6,7,8.2,4)->y

Para obtener el valor del vector *y* o get(“y”)

Operaciones con vectores númericos

- R tiene aritmética vectorial.
- Los vectores pueden ser usados en expresiones aritméticas.
- Muy útil en programación
- Los operadores más habituales en R son $+, -, *, /$ y ^ para elevar a una potencia.
- Las funciones por defecto aplicables en vectores numéricos son, log, exp, sin , cos, tan, sqrt, max, min, sum, prod, mean, median, sd, var, sort.

Operaciones con vectores númericos

Ejemplos

- Crear vectores usando otros vectores
  k<-c(y,0,y)
- R permite operar vectores con diferente longitud, el vector menor se "recicla" hasta alcanzar la longitud de vector mayor
  v<-2*y+k+1
- En este caso y aumenta hasta convertirse en un vector de longitud 11.
- podemos calcular la media de un vector
  mean(v)
- ordenarlo
  sort(v)
- Calcular la suma de sus valores
  sum(v)

Generación de secuencias regulares

- R permite generar secuencias de números fácilmente
- Si queremos generar un vector que vaya desde 1 a 20 simplemente
  a<-1:20
- Al revés
  a<-20:1
- En las expresiones el operador tiene la máxima preferencia
  n<-10
  a<-1:n-1
  a
  0 1 2 3 4 5 6 7 8 9
  a<-2*1:15
  a
  2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

Generación de secuencias regulares

- Para generar secuencias también podemos usar las funciones seq y rep
- Si queremos generar un vector que vaya desde 1 a 20 simplemente
  a<-seq(1,20)
- Si queremos generar un vector que vaya desde 1 a 20 de dos en dos
  a<-seq(1,20,by=2)
- Si queremos fijar la longitud de la secuencia en vez del intervalo entre elemnetos
  a<-seq(1,20,length=12)
- La funcion rep sirve para generar repeticiones de objetos
- Si queremos repetir un vector *n* veces
  a<-1:5; a5<-rep(a,times=5)
- Si queremos repetir cada elemento 5 veces
  a<-1:5; a5<-rep(a,each=5)

Vectores lógicos

- R permite manipular cantidades lógicas
- Los valores de un vector lógico pueden ser TRUE o T, FALSE o F y NA/NaN
- Los vectores lógicos siempre se generan mediante condiciones con operadores lógicos
- Los operadores que se usan en los vectores lógicos son == (igualdad) != (desigualdad), <, <=, >, >=, & (y lógica), | (o lógica), ! (negación lógica)
- Los vectores lógicos siempre se generan mediante condiciones
- Los vectores lógicos se pueden transformar en ceros y unos para ser utilizados en operaciones aritméticas

Vectores lógicos

Ejemplos

Crear un vector lógico cuya condición es que los valores sean mayores de 18
x->10:22
adult1<-x>18

Mediante and se puede generar un vector que cumpla dos condiciones
adult2<-x>16
adult3<-adult1&adult2

Se puede realizar una negación de un vector lógico
! adult1

Valores missing

- R indica los valores desconocidos(missing) mediante el valor NA
- R indica los valores númericos desconocidos mediante NaN (0/0)
- Cualquier operación con un valor NA o NaN te devolvera un valor NA o Nan
- Una manera de saber si un vector tiene valores desconocidos en mediante la función is.na()

Ejemplo:

- Generamos un vector con un valor desconocido
  z<-c(1:3,NA)
- Miramos que valores del vector son desconocidos
  is.na(z)

Vectores de caracteres

- Los caracteres en R deben estar rodeados de comillas
- Los vectores de caracteres se utilizan normalmente para guardar etiquetas
- Un vector de caracteres
  nombres<-c(“Maria”, “Jesus”)
- Para crear sequencias de caracteres y concatenarlos se puede utilizar la función paste
  paste(c(“Maria”, “Jesus”),1:2,sep"")
  paste(c(“Maria”, “Jesus”),1:4,sep".")

Vectores índices

- En R se pueden seleccionar elementos de un vector añadiendo al lado de su nombre un vector de índices entre corchetes ([]) Existen cuatro tipos de vectores índices: lógicos, números enteros, enteros negativos, y caracteres

Ejemplos

- Creamos un vector x<-c(1:5,NA,6:8,NA,9,10)
- Condición vector lógico k<- x[!is.na(x)]
- Vector enteros positivos x[1:5]
- Vector índice enteros negativos x[-(1:5)] Eliminamos los primeros 5 elementos
- Seleccionamos mediante caracteres y<-c(5,18,7); nombres(y)<-c(“Maria”, “Jesus”,”Pedro”) y[c(“Maria”,”Pedro”)]

Outline

1 Introducción al R

2 Números y Vectores

3 Arrays y matrices
- Arrays
- Matrices

4 Listas y data frames

5 Lectura de datos desde fichero

6 Programación en R

7 Estadística básica y gráficos

Arrays

- Conjunto de datos de k dimensiones
- En el caso de la matriz k=2
- Para cualquier array se debe especificar su dimension “dim”
- Por defecto el límite inferior es 1
  - Creamos un vector y lo transformamos en un array de 3 dimensiones
  - x<-1:24; dim(x)<-c(3,4,2)

Seleción arrays

- Al igual que los vectores los arrays puden ser seleccionados
  - Seleccionamos una la posición 1,2,3
    x[1,2,3]
  - Seleccionamos todos los elementos que se encuentran en las dos primeras dimensiones 1,2
    x[1,2,]
  - seleccionamos los elementos cuya primera dimension =1
    x[1,,]
- Los arrays se pueden crear directamente con la función array
  x<-array(1:12, dim=c(6,6))

Matrices

- La matriz es un caso particular de un array
- Se pueden crear con la función array o con la función matriz x<-array(1:12, dim=c(4,3)); x<-matrix(1:12,4,3)
- R posee funciones específicas para las matrices
- También se puede construir una matriz a partir de vectores utilizando las funciones rbind y cbind
  rbind(1:5, 11:15, 21:25)
- Averiguar el número de filas nrow(x) y columnas ncol(x)
- Multiplicar matrices $\%*\%$
- Transponerlas t(x)
- Invertirlas solve(x)
- Utilizar la diagonal diag(x)

Matrices

- R permite utilizar los operadores más comunes ($+, -, *, ..$) con matrices de las mismas dimensiones

Ejemplos

x<-matrix(1:12,4,3)
y<-matrix(13:24,4,3)
k<-y+x
k<-y-x
k<-y*x
k<-y% $*$ %x (multiplicación dimensiones diferentes)

Funciones apply

- Con apply podemos aplicar una función sobre las filas (1) o sobre las columnas (2)

Ejemplo

- Calcula la media por filas
  apply(x,1, mean)
- Calcula la media por columnas
  apply(x,2, mean)

Funciones tapply

- Con tapply podemos aplicar una función sobre grupos de array

Ejemplo

```
x<-array(1:24, dim=c(3,2,4))
tapply(1:24, x, sum)
n <- 17; fac <- factor(rep(1:3, length = n), levels = 1:5)
table(fac)
tapply(1:n, fac, sum)
```

Outline

1 Introducción al R

2 Números y Vectores

3 Arrays y matrices

4 Listas y data frames
- Listas
- Data frame

5 Lectura de datos desde fichero

6 Programación en R

7 Estadística básica y gráficos

Listas I

- En R una lista es un objeto compuesto de una colección de objetos conocidos como sus componentes
- Los componentes de una lista no tienen porque ser del mismo modo
  lista1<-list(padre="verraco01", madre="cerda01",
  num.lecho=3,peso.lecho=c(1,1.5,4))
  lista1
- Los elementos de una lista siempre estan numerados, se accede a ellos mediante esos números, seleccionando los objetos con doble corchete
  lista1[[1]]
- En caso de vector
  lista1[[4]][1]

Listas II

- La función length, puede aplicarse a una lista y devuelve el número de componentes que contiene
  length(lista1)
- Para acceder a los valores de los componentes de una lista sin necesidad de saber su posición se utiliza el símbolo dolar ($) junto al nombre del componente
  lista1$madre
  lista1$peso.lecho
  Para acceder a un deperminado peso de lechón
  lista$peso.lecho[3]

Listas III

- Los nombres de las listas se pueden añadir después de haber creado la lista
- Se crea la lista sin nombres lista1<-list(”verraco01”, “cerda01”, 3,c(1,1.5,4))
- Se añaden los nombres names(lista1)<-c(“padre”, “madre”, “num.lecho”,”peso.lecho”)
- Una vez creadas las listas pueden ser modificadas fácilmente
- Se pueden extender lista1[[5]]<-c(40,35,32)
- Añadimos el nombre del elemento nuevo names(lista1)[5]<-”longitud.lecho” lista1$longitud.lecho<-c(40,35,32)

Listas IV

- La función lapply permite aplicar una función a cada elemento de la lista

Ejemplo

- Creamos una lista lista2<-list(a=1:10, norm=rnorm(7,1,2),logic=c(TRUE,FALSE,TRUE,FALSE))
- Calculamos la media para cada elemento lapply(lista2,mean)

Data frame

- Data frame es una lista con clase “data frame”
  - Los componentes deben ser vectores, factores, matrices numéricas, listas u otras data frames
  - Los vectores de la data frame deben tener todos la misma longitud
  - Las matrices deben tener la misma longitud de fila
  - Una data frame es similar a una matriz pero con algunos atributos diferentes
- Jaula<-data.frame(Animal=54,Días=25,Peso=9.0)

attach() and detach()

- A veces es incómodo la notación $ junto al nombre de la data frame y el componente del data frame
- Jaula$Animal
- La función attach() permite trabajar con los componentes del data frame sin necesidad de $
- Al mismo tiempo se puede operar con los componentes del data frame pero no son modificados en el data frame original
- Para dejar de referirse al data frame se aplica la función detach()

Ejemplo

- Aplicamos función attach
  attach(Jaula)
- Operamos con los componentes del data frame “Jaula”
  Peso*10
  Animal<-Días*Peso
  Animal;Jaula
- Cambiamos permanentemente el valor del Animal en el data frame
  Jaula$Animal<-Jaula$Animal
- Salimos de la función
  detach(Jaula)

Outline

1 Introducción al R

2 Números y Vectores

3 Arrays y matrices

4 Listas y data frames

5 Lectura de datos desde fichero

6 Programación en R

7 Estadística básica y gráficos

read.table()

- R permite leer datos desde ficheros externos directamente
- Las funciones más utilizadas son read.table() y scan()
- Se diferencian en que read.table devuelve siempre un data.frame mientras que scan devuelve un vector o lista

- read.table(file)
- Lee un fichero en el mismo directorio de trabajo
- Por defecto asume el fichero tabular separado por espacios
- Por defecto asume que las columnas no tienen cabecera

Opciones read.table

- read.table(file, header = FALSE, sep = "", quote = "¨", dec = ".", row.names, col.names, as.is = !stringsAsFactors, na.strings = "NA", colClasses = NA, nrows = -1, skip = 0, check.names = TRUE, fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip = TRUE, comment.char = "", allowEscapes = FALSE, flush = FALSE, stringsAsFactors = default.stringsAsFactors(), encoding = "unknown")

Funciones más importantes

- file: ruta del fichero que contiene los datos a leer
- header: indica si la primera fila del fichero contiene cabecera
- sep:carácter que separa los valores de las columnas (“,”;”,”\t”)
- Todas las funciones se encuentran en ?read.table

Opciones scan

- scan(file = "", what = double(0), nmax = -1, n = -1, sep = "", quote = if(identical(sep, "")) "" else "'¨", dec = ".", skip = 0, nlines = 0, na.strings = "NA", flush = FALSE, fill = FALSE, strip.white = FALSE, quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE, comment.char = "", allowEscapes = FALSE, encoding = "unknown")

Funciones más importantes

- Funciones similares a las de read.table excepto “what”
- what: lista con los tipos de variables
- Más eficiente para leer grandes matrices
- Dos maneras de leer un mismo fichero:
  p<-scan(file=”airqualty”,what=list(Ozono=0,Mes="",viento="")
  ,sep=”\t”,skip=1)
  p<-read.table(file=”airqualty”,header=T)

Opciones write.table

- write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ", eol = "", na = "NA", dec = ".", row.names = TRUE, col.names = TRUE, qmethod = c("escape", "double"))

Funciones más importantes

- x:elementos que queremos escribir
- file= ruta del fichero dode vamos a escribir
- sep: separación entre las columnas del fichero
- col.names, row.names: indica si las columnas o filas tienen nombre
- Todas las funciones se encuentran en ?write.table

Opción cat

- cat(... , file = "", sep = " ", fill = FALSE, labels = NULL, append = FALSE)

Funciones más importantes

- Permite escribir cualquier dato en un fichero ASCII
- Si se quiere escribir repetidamente en un fichero se puede abrir una conexión con file y después cerrarla
- Ejemplo
  conec<-file("ej1.txt","x")
  cat("x1","x2","x3",file=conec,sep="\n")
  close(conec)
- Todas las funciones se pueden encontrar en ?cat

Outline

1 Introducción al R

2 Números y Vectores

3 Arrays y matrices

4 Listas y data frames

5 Lectura de datos desde fichero

6 Programación en R
- Estructuras de control
- Crear funciones propias
- Distribuciones de probabilidad
- Gráficos

Estructuras de control

- R es un lenguaje de expresiones
- Únicos comandos que acepta: funciones o expresiones que retornan un resultado
- Los comandos pueden ser agrupados colacándolos entre $\{expr_1 : \ldots , expr_2\}$
- Las instrucciones de control permiten alterar la ejecución secuencial y única de un conjunto de instrucciones tipos de estructuras
  - Ejecucuión secuencial
  - Ejecución condicional
  - Ejecución repetitiva

Ejecución condicional: if

- La condición de construye de la forma:
  $if(expr_1)expr_2[\text{else}expr_3]$
- La expresión $expr_1$ se evalua de forma lógica dando como resultado si es verdadero la $expr_2$ y si no la $expr_3$
- Ejemplo
  k<-5
  if(k<5)b<-k*3 else b<-b/3
- Los operadores && (and) y || (or) son también utilizados en expresiones condicionales
- R dispone de la versión vectorizada del if/else (condición,a,b)
- Devuelve un vector de la misma longitud que la condición

Ejecución repetitiva (bucles): for

- R ofrece diferentes alternativas:
- for(nombre in $expr_1$)$expr_2$
- nombre es el nombre de la variable del bucle, $expr_1$ es el número de repeticiones (normalmente 1:5) que se ejecutara $expr_2$ que normalmente depende de la variable el bucle (nombre) de $expr_2$

Ejemplo

x<-5:15
y<-7:17

- creamos e inicializamos el vector resultado z<-rep(0,length(x))

for(i n 1:10){
z[i]<-x[i]+y[i];
}

Ejecución repetitiva (bucles): while

- while(condición) *expr*
- *expr* se ejecuta mientras la condición sea cierta (TRUE)

Ejemplo

```
x<-5:15
y<-7:17
```

- creamos e inicializamos el vector resultado z<-rep(0,length(x))

```
while(i <= 1:10){
z[i]<-x[i]+y[i];
i<-i+1;
}
```

Ejecución repetitiva (bucles): repeat

- repeat *expr*
- *expr* se ejecuta mientras nosotros no interrumpamos el bucle mediante la instrucción break

Ejemplo

```
i<-1
repeat { if (i<=10)
{
z[i]<-x[i]+y[i];
i<-i+1;
}
else break;
}
```

Crear funciones

- Una función es un conjunto de instrucciones parametrizadas bajo un nombre determinado (ejem: mean())
- las funciones se definene mediante el siguiente patrón: nombre<-function($arg_1, arg_2, .., arg_n$)$expr$
- *expr* suele ser una expresión agrupada que usa los argumentos arg_i para calular un valor quese vuelve a la función
- Las funciones suelen tener la siguiente forma: nombre($expr_1, expr_2, ...$)

Ejemplo: solución ecuaciones segundo grado $ax^2 + bx + c = 0$

```
ec-grado2<-function(a,b,c)
{
disc<-(b^2) -(4*a*c)
sol1<-(-b+sqrt(disc))/(2*a)
sol2<-(-b-sqrt(disc))/(2*a)
c(sol1,sol2)
}
```

Ejemplo: test de t

```
Dosmuest<-function(y1,y2)
{
n1<-length(y1);n2<-length(y2)
yb1<- mean(y1);yb2<- mean(y2)
s1<- var(y1);s2<- var(y2)
s<-((n1-1)*s1+(n2-1)*s2)/(n1+n2-2)
tst<-(yb1-yb2)/sqrt(s*(1/n1+1/n2))
tst
} <-Dosmuest(datMuest1, datMuest2)
```

Distribuciones de probabilidad

- R contiene un paquete estadístico llamado stats
- Este paquete proporciona funciones que permiten obtener:
  - La función de una densidad de probabilidad (dnorm(x,mean=0,sd=1))
  - La función de distribución de la probabilidad (pnorm(q,mean=0,sd=1))
  - Calcular cuantiles (qnorm(p,mean=0,sd=1))
  - Simular de la distribución (rnorm(n,mean=0,sd=1)

Distribuciones de probabilidad

Distribución	Nombre en R	Parámetros
bet	a beta	shape1, shape2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1, df2, ncp
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m, n, k
log-normal	lnorm	meanlog, sdlog
logistic	logis	location, scale
negative binomial	nbinom	size, prob
normal	norm	mean, sd
Poisson	pois	lambda
Student's t	t	df, ncp
uniform	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m, n

Ejemplo: Binomial

- Probabilidad de que una binomial(20,0.1) tome valor 1
  dbinom(1,size=20,prob=0.1)
- Que tome valor menor a 1
  pbinom(1,size=20,prob=0.1)
- Valor con una probabilidad acumulada de 0.7
  qbinom(0.7,size=20,prob=0.1)
- Generar 20 valores de esta distribución
  rbinom(20,size=20,prob=0.1)

Comandos gráficos

- En R existen tres tipos de comandos gráficos
  - Alto nivel: crean un nuevo gráficos, probablemente con ejes, etiquetas, títulos (por defecto)
  - Bajo nivel: añaden más información a un gráfico ya existente, como puntos, líneas leyendas, etc
  - Interactivos: permiten añadir o extraer información del gráfico de manera interactiva

Alto nivel

- Generan un gráfico completo de los datos
- Genera automáticamente ejes, etiquetas y títulos
- Siempre inicia un nuevo gráfico borrando el anterior
- Función más usada “plot”, pero hay más como barplot, boxplot, hist, qqplot
  - plot(1:10)
- El gráfico se puede copiar o guardar en varios formatos

Ejemplo: Plot

- Gráfico de puntos de x contra y
  plot(x=1:10,y=1:10)
- Si f es y factor e y un vector, plot produce un boxplot
  a<-1:100;f<-rep(1:5,20):f<-factor(f)
  plot(f,a)
- Si p es un data frame obtenemos gráficas de todos contra todos
  p <-data.frame(norm=rnorm(10),unif=runif(10),expo=rexp(10))
  plot(p)

Alto nivel

- En el help de R (?plot) podéis encontrar los comandos que se pueden utilizar en plot
- type: controla el tipo de gráfico (“p” puntos, “l” líneas, “b” puntos y líneas, ...)
- xlab, ylab: etiquetas de los ejes
- xlim, ylim: modifica ls límites de los ejes
- main: título del gráfico
- axes: activar o descativar el dibujo de los ejes

Bajo nivel

- points
- lines
- text
- abline
- legend
- title
- axis

Ejemplo: uso comandos bajo nivel

```
plot(1:10,1:10,axes=FALSE,xlab=" ", ylab=")
points(runif(5,1,10),runif(5,1,10),cex=1.5,col="green")
points(runif(5,1,10),runif(5,1,10)pch="+",cex=1.7,col="green")
lines(1:10,runif(10,1,10),lty=2,lwd=2)
abline(h=5,lty=4)
title("Gráfico 1")
axis(1,labels=c("uno","tres",'cinco","siete"," nueve"),
at=seq(1,10,by=2))
axis(2,labels=1:10,at=1:10)
legend(8,2.3,lty=c(1:3),col=rainbow(3),legend=c("uno","dos","tres"))
```

Outline

1 Introducción al R

2 Números y Vectores

3 Arrays y matrices

4 Listas y data frames

5 Lectura de datos desde fichero

6 Programación en R

7 Estadística básica y gráficos
- Modelos estadísticos
- Modelos lineares

Estadísticos resumen

- mean()
- var()
- sd()
- median()
- summary()
- quantile()
- Calcular todos estádisticos de a
- a<-rnorm(100)

Estadísticos resumen

- table(x,y): podemos calcular las frecuencias
- margin.table(): frecuencias marginales
- prop.table(): proporción

Definir modelos

- R permite realizar análisis estadísticos
- La manera de establecer los modelos estadísticos es muy simple
- En este apartado nos centraremos en análisis de regresión y de varianza
- Como establecer una formula:
  - $response \sim op_1\, term_1\, op_2\, term_2\, op_3\, term_3 ..$
  - $\sim$ es usado para definir la formula del modelo
  - *response* es un vector o matriz define la variable respuesra
  - op_i operador, + o -, que impica la inclusión o exclusión de un término
  - $term_i$ puede ser vector, matriz, factor, o una fórmula

Ejemplos

$y \sim x$, $y \sim 1 + x$ regresión linear de y en x con intercept
$y \sim 0 + x$, $y \sim -1 + x$,$y \sim x - 1$ regresión linear de y en x sin intercept
log(y)$\sim$x1+x2 regresión multiple de variable y transformada
$y \sim$poly(x,2) regresión polininómica de grado 2
$y \sim A$ (A es factor)análisis de la varianza de y con las clases determinadas por A
$y \sim A + x$ análisis de varianza con una covariada x

Ejemplo: lm()

- $fitted.model < -lm(formula, data = data.frame)$
  library(ISwR)
  data(thuesen)
  lm.velo$\leftarrow$ $lm(blood.glucose \sim$short.velocity, data=thuesen)
- resumen del análisis
  summary(lm.velo)
- Podemos hacer un gráfico de la recta de regresión
  attach(thuesen)
  plot(blood.glucose,short.velocity)
  abline(lm.velo)

Ejemplo: lm()

- Las funciones fitted y resid nos devuelven los valores ajustados y los residuos
  fitted(lm.velo)
  resid(lm.velo)
- Podemos trazar segmentos entre el valor observado y el ajustado
  segments(blood.glucose,fitted(lm.velo),blood.glucose,short.velocity)
- También se pueden hacer gráficos de diagnósis del modelo
  plot(fitted(lm.velo),resid(lm.velo))
  qqnorm(resid(lm.velo))

Ejemplo: Correlación

- Análisis de correlación entre variables
  cor(blood.glucose,short.velocity)
- Si tengo missing data
  cor(blood.glucose,short.velocity,use=" complete.obs")
- R usa diferentes métodos para testar una correlación
- Por defecto (pearson)
  cor.test(blood.glucose,short.velocity)
- Métodos no parámetricos
  cor.test(blood.glucose,short.velocity,method="spearman")
- Datos categóricos
  cor.test(blood.glucose,short.velocity,method="kendall")

Ejemplo: Análisis de la varianza

- El modelo anova compara variabilidad entre grupos y en el grupo
- *anova*(*lm*(*respuesta* $\sim$ *term*))

```
library(ISwR)
data.(red.cell.folate)
attach(red.cell.folate)
anova(lm(folate~ventilation))
```

Ejemplo: Análisis de la varianza

- Para hacer las comparaciones dos a dos se puden utilizar dos métodos:
- Por defecto (holm: menos conservador)
  pairwise.t.test(folate,ventilation)
- Especificándolo (Bonferroni)
  pairwise.t.test(folate,ventilation,p.adj="bonferroni")
- R permite asumir heterogenidad de varianzas
  oneway.test(folate∼ventilation)
  pairwise.t.test(folate,ventilation,pool.sd=F)

Bibliografía

- El curso esta basado principalmente en los manuales accesibles desde la web del proyecto (htt://cran.r-project.org/manuals.html)
- Otras referencias utilizadas son:
- E. Paradis. R for Beginners
- V. Moreno, X. Solé., J. Vallas. Curso de introducción a R. Septiembre, 2005. ICO