USO DE "R" APLICADO A LAS CIENCIAS VETERINARIAS

Roberto Bustillos Huilca, MVZ, MSc.

Universidad Nacional de Loja

Carrera de Medicina Veterinaria y Zootecnia

Loja, Marzo 2019

Contenido

Características de los objetos en R: modos y atributos.

- Los objetos están compuestos de elementos. Los elementos más simples, las variables, puden ser:
  - Numeric: números reales con doble precisión. Los podemos escribir como enteros (2,-4), con fracción decimal (3.14) o con notación científica (1.2e-10).
  - Complex: números complejos de la forma a+bi.
  - Character: cadenas alfanuméricas de texto.
  - Logical: variables lógicas. Puede ser TRUE o FALSE.

Examples

```
patología <- "Varroasis"
macho <- FALSE
edad <- 10
peso <- 500.5
```

Datos especiales: NA´s.

- En algunos casos las componentes de un objeto pueden NO ser completamente conocidas.
- Cuando un elemento o valor es "not available" le asignamos el valor especial NA.
- En general una operación con elementos NA resulta NA, a no ser que mediante una opción de la función, podamos omitir o tratar los datos faltantes de forma especial.
- La opción por defecto en cualquier función es na.rm=FALSE (que indica que NO elimina los NA), que da como resultado NA cuando existe al menos un dato faltante.
- Con la opción na.rm=TRUE, la operación se efectúa con los datos válidos.

Datos especiales: NA´s.

Examples

- x <- NA
  Asignar NA a la variable x
- x + 1
  Observar que el resultado es también un NA
- y <- c(x,8,4,x)
  Asignamos al vector y dos NA y dos números
- mean(y)
  Calcula la media teniendo en cuenta los NA´s.
- mean(y,na.rm=TRUE)
  Calcula la media SIN tener en cuenta los NA´s.

Datos especiales: Inf y NaN´s.

- En la mayoría de los casos, no debemos preocuparnos de si los elementos de un objeto numérico son enteros, reales o incluso complejos. Los cálculos se realizarán internamente como números de doble precisión, reales o complejos según el caso.
- En determinadas ocasiones los cálculos realizados pueden llevar a respuestas con valor infinito positivo (Inf) o infinito negativo (-Inf).
- Es posible realizar y evaluar cálculos que involucren Inf.
- Sin embargo, a veces, determinados cálculos llevan a expresiones que no son números (representados por R como NaN´s, del inglés "not a number").

Datos especiales: Inf y NaN´s.

Examples

- sqrt(-8)
  Produce un "Warning message" avisando que es un NaN.
- sqrt(-8+0i)
  Calcula el resultado y devuelve un número complejo.
- a <- 5/0
  Asignamos al vector a un valor infinito.
- exp(-a)
  Devuelve el valor 0.
- exp(a)-exp(a)
  Devuelve NaN: es una indeterminación.

Operadores lógicos.

- Los elementos de tipo lógico tiene dos posibilidades, FALSE o TRUE. Se pueden abreviar en F y T respectivamente.
- Los objetivos lógicos son generalmente fruto de una comparación.

Examples

Si w <- 6

v <- w>10, produce un objeto lógico de longitud la de x con valor F o V.

z <- w==6

- Las comparaciones que dan un resultado lógico son: <, <=, >, >=, ==, !=.

Operadores lógicos.

- La aritmética entre objetos lógicos se puede llevar a cabo con los siguientes operadores lógicos.

Operador	Operación
$!x$	Negación de x. Los T los convierte en F y viceversa.
$x\&y$	Intersección, operador lógico y: T y T da T, otra comparación da F
$x\|y$	Unión, operador lógico o: F y F da F, otra comparación da T.
`xor(x,y)`	Exclusivo OR, `xor(T,F)==T`, otra comparación da F.
`all`	Para una secuencia de argumentos lógicos, `all` devuelve el valor lógico que indica si todos los elementos son TRUE.
`any`	Para una secuencia de argumentos lógicos, `any` devuelve el valor lógico que indica si algún elemento es TRUE.

Operadores lógicos.

Examples

x <- c(T,T,F,F)
y <- c(T,F,T,F)
x&y produce TRUE FALSE FALSE FALSE
x|y produce TRUE TRUE TRUE FALSE
any(x) produce TRUE
all(x) produce FALSE

Atributos de los objetos.

- Utilizando la función str(objeto) podemos obtener información sobre su estructura.
- Los atributos de un objeto suministran información específica sobre el propio objeto. TODOS los objetos tienen dos atributos intrínsecos: el modo y su longitud.
- Las funciones mode(objeto) y length(objeto) se pueden utilizar para obtener el modo y longitud de cualquier estructura.

Examples

código <- c(1,4,9,10)
mode(código) Devuelve el tipo numeric
length(código) Devuelve la longitud
asociación <- c("Riesgo","OR","RR")
mode(asociación) Devuelve el tipo character
length(asociación) Devuelve la longitud

Asignación. Coerción de tipos.

- Recordar que la función principal para definir un objeto es a través de sus componentes, con la función c( ), mediante el comando más importante en R que es <- el de la asignación.
- Las asignaciones pueden realizarse también con una flecha apuntando a la derecha, realizando el cambio obvio en la asignación.

Examples

h <- 200 es equivalente a 200 -> h pero no a h < - 200

- La asignación puede realizarse también mediante la función assign( ).

Examples

d <- c(4,8,12) es equivalente a assign("d", c(4,8,12))

- En muchos contextos el símbolo = puede también utilizarse indistintamente para asignar un objeto.

Coerción de tipos.

- La mayoría de las funciones producen un error cuando el tipo de datos que esperan no coincide con los que ponemos en los argumentos.
  - Comprobar el tipo de datos utilizando funciones is.algo( ), que nos responde con un valor lógico.
  - Forzar al tipo de datos deseado coercionando, para lo cual podemos utilizar funciones del tipo as.algo( ), que fuerzan el tipo de datos.
- Para conocer si un elemento es NA, la comparación lógica ==, no puede funcionar ya que NA es la nada. Utilizamos para ello la función is.na( ), que nos responde con un valor lógico: TRUE para los elementos del vector que son NA.
- De manera similar, podemos utilizar is.finite( ), is.nan( ), etc

Coerción de tipos.

Tipo	Comprobación	Coerción
array	is.array()	as.array()
character	is.character()	as.character()
complex	is.complex()	as.complex()
double	is.double()	as.double()
factor	is.factor()	as.factor()
integer	is.integer()	as.integer()
list	is.list()	as.list()
logical	is.logical()	as.logical()
matrix	is.matrix()	as.matrix()
NA	is.na()	-
NaN	is.nan()	-
NULL	is.null()	as.null()
numeric	is.numeric()	as.numeric()
ts	is.ts()	as.ts()
vector	is.vector()	as.vector()

Examples

Crear el vector: secuencia <- 1:15
is.numeric(secuencia) Resulta en TRUE
is.vector(secuencia) Resulta en TRUE
is.complex(secuencia) Resulta en FALSE
is.character(secuencia) Resulta en FALSE
caracter <- as.character(secuencia) Fuerza el vector example a tomar los valores ("1","3","4"..."15")

Factores-Variables categóricas.

- Un factor es un vector utilizado para especificar una clasificación discreta de los elementos de otro vector de igual longitud.
- En R existen dos tipos de factores.
  - No ordenados (nominales): No existe jerarquía entre ellos (p.e., colores).
  - Ordenados (ordinales): Existe jerarquía entre ellos (p.e., grupos de edad).
- Se pueden crear a partir de un vector numérico con las funciones as.factor( ), as.ordered( ), gl( ).
- También a partir de un vector de caracteres utilizando factor( ).
- Las etiquetas se asignan con levels( ).

Examples

ejemplo <- as.factor(c(4,5,6,5,6,6,5,4,4)) Factor con 3 categorías.
levels(ejemplo) <- c("bajo","medio","alto")
ejemplood <- as.ordered(ejemplo) Factor ordenado
age <- gl(4,5,15,labels=c("Terneros","Vaconas","Vacas","Toros"))
age

Vectores de caracteres: la función paste( ).

- R tiene algunos vectores de caracteres predefinidos: LETTERS, letters, month.name y month.abb.
- La función paste( ) une todos los vectores de caracteres que se le suministran y construye una sola cadena de caracteres. Muy útil en gráficas.
- También admite argumentos numéricos, que convierte inmediatamente en cadenas de caracteres.
- En su forma predeterminada, en la cadena final, cada argumento original se separa del siguiente por un espacio en blanco, aunque ello puede cambiarse utilizando el argumento sep="cadena", que sustituye el espacio en blanco por cadena, la cual podría ser incluso vacía.

Examples

```
ABC <- LETTERS
abc <- letters
MESES <- month.name
meses <- month.abb
etiquetas <- paste(c("X","Y"),1:12,sep="")
```

Generación de secuencias regulares.

- El operador más usual es : que genera una secuencia desde: hasta en incrementos (o decremento si hasta es menor que desde) de uno.
- La función seq( ) permite generar sucesiones más complejas. Dispone de varios argumentos (from, to, by, length). Existen funciones similares.
- Una función relacionada es rep( ), que permite duplicar un objeto de formas diversas. Su forma más sencilla es rep(x, times=5).

Examples

pi:10
seq(0,1,length=20)
seq(5); 1:5; seq(1,5)
seq(1,10,by=0.5)
rep(1:6,2)
rep(1:4,c(2,3,1,2))
rep(1:4,c(2,2)) Nos da un error
sequence(c(2,5))

Matrices. Operaciones con matrices.

- Una matriz es un objeto con un atributo adicional el cual a su vez es un vector numérico de longitud 2, que define el número de filas y columnas de la matriz.
- Todos los elementos deben ser del mismo tipo.
- Una matriz se define con el comando matrix( ) especificando el número de filas y columnas o asignando la dim a un vector.
- Recordar que la matriz se crea por columnas, aunque con la opción byrow=TRUE lo hace por filas.
- Podemos asignar nombres a las filas y columnas con el atributo dimnames.
- Las funciones is.matrix( ) y as.matrix( ) comprueban o fuerzan el carácter de matriz de un objeto.

Examples

```
a <- c(1:6)
dim(a) <- c(2,3)
dimnames(a) <- list(c("F1","F2"),c("C1","C2","C3"))
Ej <- ma-
trix(1:12,ncol=3,byrow=T,dimnames=list(letters[1:4],LETTERS[1:3]))
Ej[1,1]
Ej[,c(2,3)]
Ej[,c(-1,-3),drop=F]
s <- matrix(rep(c(T,F),6),4,3)
```

Los data.frame

- Las bases de datos en estadística son, habitualmente de la forma:

		Covariables		
individuo	anyos	implante	edad	sexo
1	1.3	2	22	H
2	0.4	2	21	M
3	1.1	2	34	H
4	2.3	1	42	H
5	3.1	3	17	M
6	1.3	1	43	H

- R organiza este tipo de información en objetos del tipo data.frame, un caso particular de lista.
- Los data.frame son apropiados para describir "matrices de datos" donde cada fila representa a un individuo y cada columna una variable, cuyas variables pueden ser numéricas o categóricas.

Los data.frame u hojas de datos.

- Los data.frame se crean con la función data.frame( )

Examples

```
dataimp <- data.frame(anios=c(1,3,5,7,9,7),clase=c(2,3,3,2,2,1),
edad=c(12,24,36,48,56,48),sexo=c("M","H","M","H","H","M"))
```

- Los componentes deben ser vectores (numéricos, carácter o lógicos), factores, matrices numéricas u otros data.frames.
- Puesto que un vector representará una variable del banco de datos y las columnas de una matriz representarán varias variables de ese mismo banco, la longitud de los vectores debe ser la misma y coincidir con el número de filas de las matrices.
- Los datos que no son numéricos, la estructura de data.frame los considera factores, con tantos niveles como valores distintos encuentre, tal y como ocurre en el ejemplo que estamos considerando.

Las funciones attach( ) y detach( ).

- Para trabajar con las variables del banco de datos, podemos utilizar la notación estándar de las listas, $nombre o [[ ]], pero resulta más natural emplear simplemente el nombre de la columna.
- El problema es que R guarda el nombre del data.frame pero no sus variables. Para poderlas utilizar con su nombre como vectores, tenemos que utilizar la función attach(nombre de data.frame). La operación inversa se realiza con la función detach( ).
- En el caso de los data.frames es posible seleccionar filas y columnas deseadas, y crear con ellas otros data.frames con las características deseadas con la función subset( ).

Examples

```
dataimp <- data.frame(anios=c(1,3,5,7,9,7),clase=c(2,3,3,2,2,1),
edad=c(12,24,36,48,56,48),sexo=c("M","H","M","H","H","M"))
anios
attach(dataimp); anios
detach(dataimp); anios
data.hembra.filtrada <- dataimp$anios[dataimp$sexo=="H"]
mas.peq <- subset(dataimp,anios>1,select=c(edad,sexo))
```

Transformaciones.

- Podemos modificar o manipular cualquier objeto con operadores (aritméticos, lógicos y comparativos) y funciones. Así, por ejemplo, si una variable no es normal, puedo transformarla para conseguir dicha normalidad.
- !Cuidado! Cualquier modificación que hagamos con las variables de un banco de datos "adjuntando" (attached) no afectará al propio data.frame.
- Para modificar el contenido interno de los data.frames podemos utilizar la función transform( ). En concreto podemos:
  - Transformar variables existentes (para conseguir normalidad).
  - Crear nuevas variables mediante la transformación de variables existentes.

Recodificaciones.

- Mediante la función recode( ) de la librería car es posible llevar a cabo recodificaciones de otras variables.
- Debemos especificar cuál es la variable a recodificar, cuáles son los criterios de recodificación y si deseamos que la nueva variable resultante de la recodificación sea un factor o no.
- !Cuidado! Las recodificaciones tampoco afectarán al propio data.frame.

Examples

attach(dataimp)
edad.final <- edad + anios; No afecta al data.frame
dataimp.1 <- transform(dataimp, edad.final=edad+anios) Si afecta
dataimp.2 <- transform(dataimp.1,edad=edad+1) Altera la variable
install.packages("car"); library(car) Instalar y cargar paquete.
edc <- recode(edad," 10:25='joven';
26:65='adulto'",as.factor=TRUE)
sexo.cod <- recode(sexo,"'H'=0; 'M'=1",as.factor=TRUE)

Lectura de ficheros de datos.

- Hasta ahora hemos visto cómo introducir datos interactivamente en forma de objetos, aunque habitualmente los datos nos vienen proporcionados en archivos externos.
- Para asignar un banco de datos a un data.frame se utiliza la función read.table( ). La forma más sencilla es con un fichero en el que:
  - La primera línea del archivo debe contener el nombre de cada variable de la hoja de datos.
  - En cada una de las siguientes líneas, el primer elemento es la etiqueta de la fila y a continuación deben aparecer los valores de cada variable. Si no R, asigna automáticamente etiquetas a las filas.

Examples

```
rats <- read.table(file="C:/Datos/ratas.txt",header=T)
```

Examples

```
library(readr)
NOTAS <- read_delim("C:/Users/Roberto
Bustillos/Documents/UNL/PROYECTO DE
TITULACION/Notas.csv", ";", escape_double = FALSE, locale =
locale(decimal_mark = ","), trim_ws = TRUE)
```

Tarea

Agregar al data.frame otra variable que se llame CATEGORIA, esta debe tener tres niveles: Bueno si la nota está entre 1 y 1.50; Muy bueno si la nota está entre 1.51 y 2.0; y Excelente si la nota está entre 2.01 y 2.50.

Importar datos de otros programas.

- La gran aceptación de R está haciendo cada vez más fácil el importar ficheros de otros programas.
- La librería foreign contiene funciones para importar y exportar datos en el formato de otros programas.

Examples

library(foreign) Cargamos el paquete
datos <- read.spss(file="glucosa.sav",to.data.frame=TRUE)
str(datos)