The Data: Vectors and Matrices

Susan Holmes (c)

Data structures: vectors

We have already seen examples of vectors we created using the `c()` that combines values:

```
fib <- c(0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987)
class(fib)
```

```
## [1] "numeric"
```

```
is.vector(fib)
```

```
## [1] TRUE
```

How many elements do you think there are in `c(fib,fib)` ?
We can also combine elements in the middle:

```
fib5fib <- c(fib,5,fib)
fib5fib
```

```
##  [1]   0   1   1   2   3   5   8  13  21  34  55  89 144 233 377 610 987
## [18]   5   0   1   1   2   3   5   8  13  21  34  55  89 144 233 377 610
## [35] 987
```

Let's try some operations on fib:

```
fib+10
```

```
##  [1]  10  11  11  12  13  15  18  23  31  44  65  99 154 243 387 620 997
```

```
fib*10
```

```
##  [1]    0   10   10   20   30   50   80  130  210  340  550  890 1440 2330
## [15] 3770 6100 9870
```

Try these commands out:

```
fib5fib+c(1,10,100,1000,10000)
```

```
##  [1]     1    11   101  1002 10003     6    18   113  1021 10034    56
## [12]    99   244  1233 10377   611   997   105  1000 10001     2    12
## [23]   103  1005 10008    14    31   134  1055 10089   145   243   477
## [34]  1610 10987
```

```
fib+c(1,100)
```

```
## Warning in fib + c(1, 100): longer object length is not a multiple of
## shorter object length
```

```
##  [1]   1 101   2 102   4 105   9 113  22 134  56 189 145 333 378 710 988
```

Question: Can you explain how R is trying to add vectors of different lengths?

Answer: In fact to do operation between unequal lengthed vectors, R tries to recycle values to make the operations work, this can cause confusion when it goes ahead and does things when you really made a typing error.

Question: Try out different operations on vectors: - "/","+","^2","log","exp","cos",…

Indexing vectors

We saw that the number [1] appears as the first index of the vector on the left. We use the indexing to reach certain elements of the vector. R indices start at 1.

```
fib[1]
```

```
## [1] 0
```

```
fib[4]
```

```
## [1] 2
```

```
fib[3:5]
```

```
## [1] 1 2 3
```

We can access only certain elements given by indices in their own vector, for instance c(1,3,5)

```
fib[c(1,3,5)]
```

```
## [1] 0 1 3
```

A negative index means take out that value from the vector:

```
fib[-2]
```

```
##  [1]   0   1   2   3   5   8  13  21  34  55  89 144 233 377 610 987
```

Taking a random subset of a vector

```
vec4 <- 4:32
length(vec4)
```

```
## [1] 29
```

```
sample(vec4,size=10)
```

```
##  [1] 17 23  7 22  6 31  5 29 28 16
```

```
sample(vec4,size=10)
```

```
##  [1]  5 15 13 23 25 26 14 29 28  6
```

`sample` takes a random subset of the input, here the vector `vec4`

Question Why do the two calls to the same function with the same input and arguments give two different answers?

Many variables measured on individuals: matrices

R was created so we can easily manipulate, summarize and visualize data. The first structure that allows us to group together several measurements on the same people/animals/samples are matrices.

(Here we have interjected a few comments in the code using the # character.) We create a matrix with the `matrix` function.

```
  A <- matrix(
   c(4,2,0,3,1,7,2,8,4,5),    # the data elements
   nrow=2,                  # number of rows
   ncol=5)                  # number of columns
A
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    4    0    1    2    4
## [2,]    2    3    7    8    5
```

You can check what type of object A is by asking R

```
class(A)
```

```
## [1] "matrix"
```

```
mode(A)
```

```
## [1] "numeric"
```

```
is.matrix(A)
```

```
## [1] TRUE
```

```
is.vector(A)
```

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

```
is.numeric(A)
```

```
## [1] TRUE
```

Matrices have to have all its entries of the same mode.

Q: A vector also has to have homogeneous entries, it's not always obvious, try typing `c(letters[3],4,6,letters[7])` what do you notice?

```
matlet <- matrix(letters,ncol=26,nrow=5)
matlet
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,] "a"  "f"  "k"  "p"  "u"  "z"  "e"  "j"  "o"  "t"   "y"   "d"   "i"
## [2,] "b"  "g"  "l"  "q"  "v"  "a"  "f"  "k"  "p"  "u"   "z"   "e"   "j"
## [3,] "c"  "h"  "m"  "r"  "w"  "b"  "g"  "l"  "q"  "v"   "a"   "f"   "k"
## [4,] "d"  "i"  "n"  "s"  "x"  "c"  "h"  "m"  "r"  "w"   "b"   "g"   "l"
## [5,] "e"  "j"  "o"  "t"  "y"  "d"  "i"  "n"  "s"  "x"   "c"   "h"   "m"
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,] "n"   "s"   "x"   "c"   "h"   "m"   "r"   "w"   "b"   "g"   "l"
## [2,] "o"   "t"   "y"   "d"   "i"   "n"   "s"   "x"   "c"   "h"   "m"
## [3,] "p"   "u"   "z"   "e"   "j"   "o"   "t"   "y"   "d"   "i"   "n"
## [4,] "q"   "v"   "a"   "f"   "k"   "p"   "u"   "z"   "e"   "j"   "o"
## [5,] "r"   "w"   "b"   "g"   "l"   "q"   "v"   "a"   "f"   "k"   "p"
##      [,25] [,26]
## [1,] "q"   "v"
## [2,] "r"   "w"
## [3,] "s"   "x"
## [4,] "t"   "y"
## [5,] "u"   "z"
```

```r
matlet <- matrix(letters,ncol=26,nrow=5,byrow=TRUE)
matlet
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"
## [2,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"
## [3,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"
## [4,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"
## [5,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,] "n"   "o"   "p"   "q"   "r"   "s"   "t"   "u"   "v"   "w"   "x"
## [2,] "n"   "o"   "p"   "q"   "r"   "s"   "t"   "u"   "v"   "w"   "x"
## [3,] "n"   "o"   "p"   "q"   "r"   "s"   "t"   "u"   "v"   "w"   "x"
## [4,] "n"   "o"   "p"   "q"   "r"   "s"   "t"   "u"   "v"   "w"   "x"
## [5,] "n"   "o"   "p"   "q"   "r"   "s"   "t"   "u"   "v"   "w"   "x"
##      [,25] [,26]
## [1,] "y"   "z"
## [2,] "y"   "z"
## [3,] "y"   "z"
## [4,] "y"   "z"
## [5,] "y"   "z"
```

```r
dim(matlet)
```

```
## [1]  5 26
```

```r
nrow(matlet)
```

```
## [1] 5
```

```r
ncol(matlet)
```

```
## [1] 26
```

You see that by default the function matrix takes a vector and fills in the data column by column., in order to change that you have put a special byrow=TRUE argument.

Now is a good time to revist the help to understand how to read the default arguments.

```
?matrix
```

Q: Where do you see the default value of the argument byrow ?

Accessing Matrix elements

Matrices are sometimes called two dimensional arrays.
The rows of a matrix are the first index, the columns are the second.
Now suppose we want to replace the forth column of A by two 1's.
First take a look at the current values:

```
A[,4]
```

```
## [1] 2 8
```

Strangely the column has become a row ? This is because all vectors appear as row vectors to save space.

Now we replace the values by 1:

```
A[,4] <- 1
A
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    4    0    1    1    4
## [2,]    2    3    7    1    5
```

Transposition

```
t(A)
```

```
##      [,1] [,2]
## [1,]    4    2
## [2,]    0    3
```

```
## [3,]    1    7
## [4,]    1    1
## [5,]    4    5
```

The transpose of the matrix A has the number of rows equal to the number of columns of A, so what do you think `dim(t(A))` will be?

Data Matrices

In real situations matrices represent data where the rows are the observations and the columns are the variables.

```
observNames <- c("H1","H3","H4","H5","H7","H8","H9")
vecHapl <- c(14,12,4,12,3,10,8,10,1,4,15,13,0,1,1,15,13,4,13,3,9,8,10,1,4,13,12,0,1,1,
15,11,5,11,3,10,8,10,1,4,11,14,0,1,1,17,13,4,11,3,10,7,10,1,4,14,12,0,1,1,
13,12,5,12,3,11,8,11,1,4,14,14,0,1,1,16,11,5,12,3,10,8,10,1,4,11,15,0,1,1,
16,11,5,11,3,10,8,10,1,4,11,14,0,1,1)
matHap <- matrix(vecHapl,nrow=7,byrow=TRUE)
rownames(matHap) <- observNames
```

Suppose I also want to name the columns and type:

```
POSnames <- c(DYS19,DXYS156Y,DYS389m,DYS389n,DYS389p,DYS389q,DYS390m,
DYS390n,DYS390p,DYS390q,DYS392,DYS393,YAPbcbc,SRY1532bb,92R7bb)
```

Question: Why does this create an error?

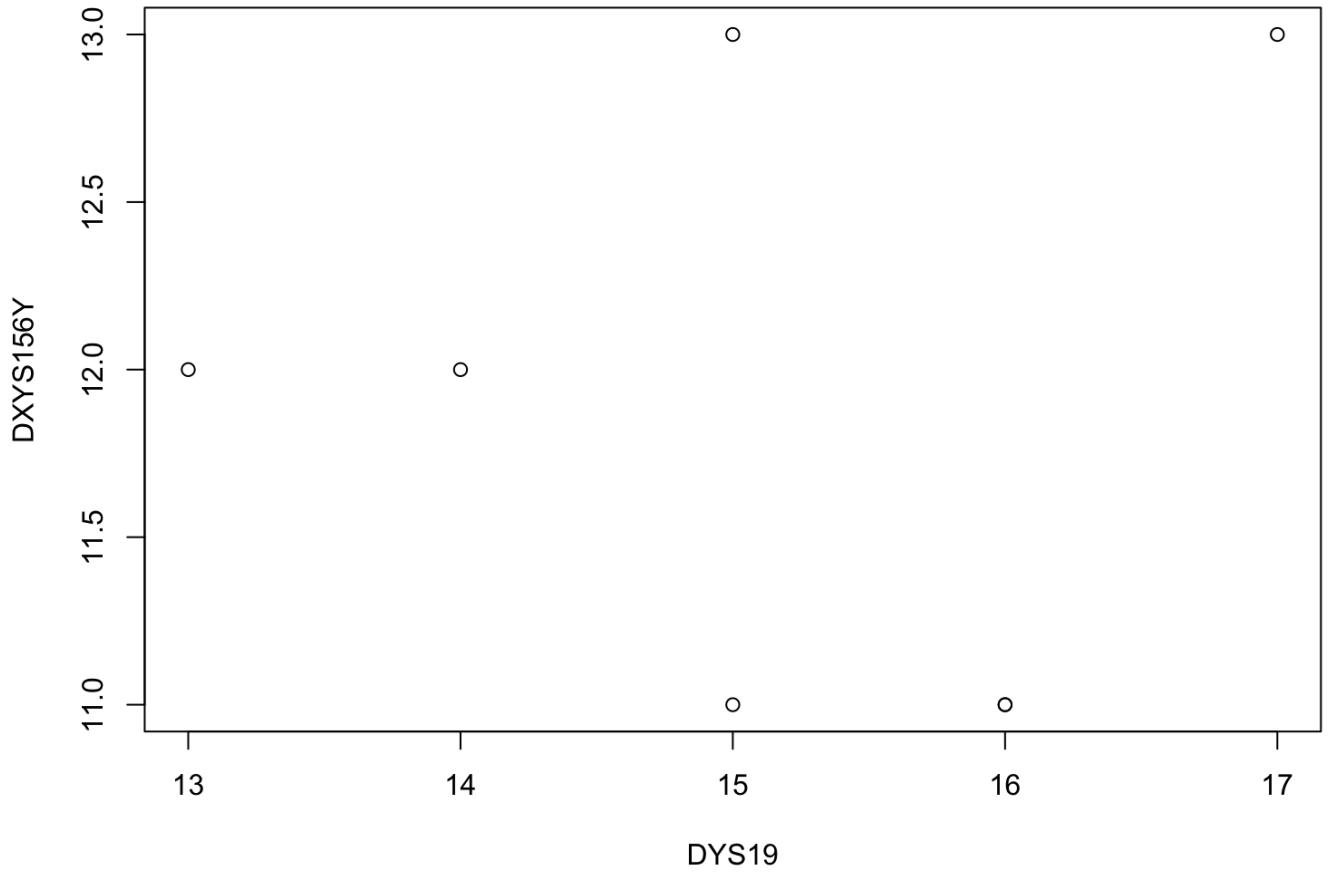
```
POSnames <- c("DYS19","DXYS156Y","DYS389m","DYS389n","DYS389p",
              "DYS389q","DYS390m","DYS390n","DYS390p","DYS390q",
              "DYS392","DYS393","YAPbcbc","SRY1532bb","92R7bb")
colnames(matHap) <- POSnames
matHap
```

```
##    DYS19 DXYS156Y DYS389m DYS389n DYS389p DYS389q DYS390m DYS390n DYS390p
## H1    14       12       4      12       3      10       8      10       1
## H3    15       13       4      13       3       9       8      10       1
## H4    15       11       5      11       3      10       8      10       1
## H5    17       13       4      11       3      10       7      10       1
## H7    13       12       5      12       3      11       8      11       1
## H8    16       11       5      12       3      10       8      10       1
## H9    16       11       5      11       3      10       8      10       1
##    DYS390q DYS392 DYS393 YAPbcbc SRY1532bb 92R7bb
## H1       4     15     13       0         1      1
## H3       4     13     12       0         1      1
## H4       4     11     14       0         1      1
```

```
## H5      4     14     12      0       1      1
## H7      4     14     14      0       1      1
## H8      4     11     15      0       1      1
## H9      4     11     14      0       1      1
```

Each row of matHap corresponds to a person, whose ID starts with ‘H’ and each columns represents a special position on the Y chromosome where repeats can occur, the numbers are the number of repeats, so they are integers.

```
plot(matHap)
```



By default `plot` makes a scatter plot of the first two columns of matHap.

Saving matrices

We can save our data to a simple text file for later use in various ways:

As an R object:

```
save(matHap,file="matHap.RData")
```

As a plain text file:

```
write.table(matHap, file="matHap.txt")
# Take a look at the file
file.show("matHap.txt")
```

You can't look at an `.RData` file because they are not text files but compressed binary versions of the information, so humanly unreadable, although later we will be able to load the data just by typing

```
load("matHap.RData")
```

Question: Why do we need quotes within the brackets here?

Summary of this Session:

- We have learned how to create vectors both with `c()` and with `:` and how some operators work.
- We saw that they are one dimensional with one index that we can use to access particular elements, we can also make a subvector by leaving out some elements using the `-`.
- We saw how to take a random sample of elements from a vector.
- We then saw how to create and use Matrices, which have all their cells of the same mode or type (we saw all numeric and all character, they could also all be logical).
- We saw how to name rows and columns and how to access certain elements with the double indices.
- The `plot` function understands what to do with a matrix.
- Matrices are two dimensional arrays (row and columns), there are actually higher dimensional arrays in R.

Questions:

-1 List three functions we introduced this session and look at their help documentation.

-2 Restart R, clear the workspace using `rm(list=ls())` and load the matHap data again from the file `matHap.RData`.