Chapter 2 advanced R

Brigid McDermott

May 9, 2017

Initial attempt at opening quiz.

Trying to answer these before I read the chapter

1. What is the result of subsetting a vector with positive integers, negative integers, a logical vector, or a character vector? – You get a vector with the same type of elements from which you took the subset.{Whoops this is not what this question meant. See the first part of chapter on subsetting.}
2. What's the difference between [, [[, and $ when applied to a list? – I have never been really clear on this, but something like [gives you an element of a list as a list, [[gives you the element as the underlying structure (vector, db etc) and $ gives you named elements of the list.
3. When should you use drop = FALSE? – when you are subsetting from a factor, say for example taking all the men from from men/women factor but you want to keep all the levels in the new subsetted factor.
4. If x is a matrix, what does x[] <- 0 do? How is it different to x <- 0? – I don't know;let me try. Ok the first produces the structure matrix with 0 for all elements, while the second produces an atomic vector with the type double, value 0
5. How can you use a named vector to relabel categorical variables? –this one I knew

```
animals<-c( c="cat",d="dog", m="mouse", r="rabbit")
x<-rep(c("c","d","m","r"),c(4,3,2,1))
x
```

```
##  [1] "c" "c" "c" "c" "d" "d" "d" "m" "m" "r"
```

```
animals[x]
```

```
##        c        c        c        c        d        d        d        m
##    "cat"    "cat"    "cat"    "cat"    "dog"    "dog"    "dog"  "mouse"
##        m        r
##  "mouse" "rabbit"
```

```
animals<-c( c="cougar",d="dolphin", m="moose", r="raptor")
animals[x]
```

```
##         c         c         c         c         d         d         d
##  "cougar"  "cougar"  "cougar"  "cougar" "dolphin" "dolphin" "dolphin"
##         m         m         r
##   "moose"   "moose"  "raptor"
```

```
unname(animals[x]) #### nice, I didn't know unname
```

```
##  [1] "cougar"  "cougar"  "cougar"  "cougar"  "dolphin" "dolphin" "dolphin"
##  [8] "moose"   "moose"   "raptor"
```

note to myself about "outer" function

I looked at http://www.endmemo.com/program/R/outer.php

```
x <- c(1,2.3,2,3,4,8,12,43)
y<- c(2,4)
outer(x,y)
```

```
##      [,1]  [,2]
## [1,]  2.0   4.0
## [2,]  4.6   9.2
## [3,]  4.0   8.0
## [4,]  6.0  12.0
## [5,]  8.0  16.0
## [6,] 16.0  32.0
## [7,] 24.0  48.0
## [8,] 86.0 172.0
```

```
### same thing as
x%o%y
```

```
##      [,1]  [,2]
## [1,]  2.0   4.0
## [2,]  4.6   9.2
## [3,]  4.0   8.0
## [4,]  6.0  12.0
## [5,]  8.0  16.0
## [6,] 16.0  32.0
## [7,] 24.0  48.0
## [8,] 86.0 172.0
```

So if you have x with a elements and y with b elements then outer(x,y) will result in a matrix with a rows and b columns where element [1,1] in matrix is x[1]*y[1] and element [3,4] is x[3]*y[4] etc. You can define the function another function instead of the default *, i.e + .

```
outer(x,y,FUN="+")
```

```
##      [,1] [,2]
## [1,]  3.0  5.0
## [2,]  4.3  6.3
## [3,]  4.0  6.0
## [4,]  5.0  7.0
## [5,]  6.0  8.0
## [6,] 10.0 12.0
## [7,] 14.0 16.0
## [8,] 45.0 47.0
```

```
## or from the Advanced R book
vals <- outer(1:5, 1:5, FUN = "paste", sep = ",")
vals
```

```
##      [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] "1,1" "1,2" "1,3" "1,4" "1,5"
## [2,] "2,1" "2,2" "2,3" "2,4" "2,5"
## [3,] "3,1" "3,2" "3,3" "3,4" "3,5"
## [4,] "4,1" "4,2" "4,3" "4,4" "4,5"
## [5,] "5,1" "5,2" "5,3" "5,4" "5,5"
```

Exercises

1.Fix each of the following common data frame subsetting errors: mtcars[mtcars$cyl== 4,] ## double equal sign for equality logical mtcars[-1:-4,] ## cannot mix negative and positive indexes, this will remove 1 to 4 first rows mtcars[mtcars$cyl <= 5,]##logical identifies rows so need a comma to make this clearly row identifier mtcars[mtcars$cyl == 4|mtcars$cyl==6,] ## syntax requires it this way

2. Why does x <- 1:5; x[NA] yield five missing values? (Hint: why is it different from x[NA_real_]?) I believe it is because in the case of x[NA] it is asking if each element is a missing logical and in each case the answer is NA because the logical cannot be compared with the double. In the second case it asks where in x is an element matched as NA_real and the answer is nowhere, ie. NA.

3.What does upper.tri() return? How does subsetting a matrix with it work? Do we need any additional subsetting rules to describe its behaviour? The function upper.tri() returns a logical matrix that gives true values on the positions above the diagnoal of the matrix. I am not sure how this would work on a non-square matrix. I checked as below and see that the diagonal starts at the upper left value and ends before the lower right value. The default of this command is not to include the values in the diagonal but the option is to change it to "diag=TRUE"

```
x<-matrix(1:24,nrow=4)
upper.tri(x)
```

```
##       [,1]  [,2]  [,3]  [,4] [,5] [,6]
## [1,] FALSE  TRUE  TRUE  TRUE TRUE TRUE
## [2,] FALSE FALSE  TRUE  TRUE TRUE TRUE
## [3,] FALSE FALSE FALSE  TRUE TRUE TRUE
## [4,] FALSE FALSE FALSE FALSE TRUE TRUE
```

4.Why does mtcars[1:20] return an error? How does it differ from the similar mtcars[1:20,]? The first notation attempts to use column indexing on a data.frame but there are only 20 columan is mtcar so the command cannot be executed. Note that mtcar[1:5] will work. The second notation asks for the first 20 rows of the data.frame mtcars as indicated my the comma in the subsetting brackets[].

5.Implement your own function that extracts the diagonal entries from a matrix (it should behave like diag(x) where x is a matrix).

```
x<-matrix(letters[1:16],nrow=4)
x
```

```
##      [,1] [,2] [,3] [,4]
## [1,] "a"  "e"  "i"  "m"
## [2,] "b"  "f"  "j"  "n"
## [3,] "c"  "g"  "k"  "o"
## [4,] "d"  "h"  "l"  "p"
```

```
y<-rep(NA,nrow(x))
for(i in 1:nrow(x)){y[i]<-x[i,i]}
y
```

```
## [1] "a" "f" "k" "p"
```

6. What does df[is.na(df)] <- 0 do? How does it work? This command looks like it will index the data.fram where it contains missing values and replaces them with zeros. Let me check because I am not sure is.na() will work across the data.frame in this way. No I checked below and it did replace all missing with 0.

```
x<-matrix(letters[1:16],nrow=4)
diag(x)<-NA
```

```
x<-as.data.frame(x, stringsAsFactors=F)
str(x)
```

```
## 'data.frame':    4 obs. of  4 variables:
##  $ V1: chr  NA "b" "c" "d"
##  $ V2: chr  "e" NA "g" "h"
##  $ V3: chr  "i" "j" NA "l"
##  $ V4: chr  "m" "n" "o" NA
```

```
x[is.na(x)]<-0
x
```

```
##   V1 V2 V3 V4
## 1  0  e  i  m
## 2  b  0  j  n
## 3  c  g  0  o
## 4  d  h  l  0
```

```
x[x==0]<-NA
x
```

```
##     V1   V2   V3   V4
## 1 <NA>    e    i    m
## 2    b <NA>    j    n
## 3    c    g <NA>    o
## 4    d    h    l <NA>
```

```
complete.cases(x)### logical for rows of data.frame
```

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

Symplifying versus Preserving Subsetting.

I have never understood this well and have got into trouble over it. Mostly when a one column data.frame or matrix converts to a vector.

Exercises

1.Given a linear model, e.g., mod <- lm(mpg ~ wt, data = mtcars), extract the residual degrees of freedom. Extract the R squared from the model summary (summary(mod))

```
mod<-lm(mpg~wt,data=mtcars)
out<-summary(mod)
out$r.squared
```

```
## [1] 0.7528328
```

```
out[[8]]
```

```
## [1] 0.7528328
```

```
out["r.squared"]
```

```
## $r.squared
## [1] 0.7528328
```

```
str(out["r.squared"]) ### List of 1
```

```
## List of 1
##  $ r.squared: num 0.753
```

```
str(out[[8]])### number
```

```
##  num 0.753
```

The function "which" gives the indexes of a logical object where that logical object equals true

```
y<-1:10<6
y ## so y is a logical object
```

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

```
which(y)
```

```
## [1] 1 2 3 4 5
```

```
alpha<-letters[1:10]
alpha[which(y)]
```

```
## [1] "a" "b" "c" "d" "e"
```

```
### wanted to examine this
unwhich <- function(x, n) {
  out <- rep_len(FALSE, n)
  out[x] <- TRUE
  out
}

unwhich(which(y), 10) ### ok easy, create logical vector as long as n with all false
```

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

```
## then use the which(x) to id the locations (indices) where the logical vector should be true.
```

Exercises

1. How would you randomly permute the columns of a data frame? (This is an important technique in random forests.) Can you simultaneously permute the rows and columns in one step? Answer: Knowing that there are 32 rows and 11 columns you could simultaneously permutate the rows and columns as mtcars[sample(1:32,32),sample(1:11,11)]
2. How would you select a random sample of m rows from a data frame? What if the sample had to be contiguous (i.e., with an initial row, a final row, and every row in between)? Answer: I think what they are asking here is how do you take a sample of a block of m rows from a data.frame. This boils down to sampling from 1 to nrow(df)-m, then using this as the starting point and taking the next m-1 records in the file y<-sample(1:(nrow(df)-m),1) myMsample<-df[y:y+(m-1),]

3.How could you put the columns in a data frame in alphabetical order? Answer: mtcars[,order(names(mtcars))]