Cheatography

R - Subsetting Cheat Sheet

by bwaldo via cheatography.com/35095/cs/11019/

Basics

There are a number of operators that can be used to extract subsets of R objects.

* [- always returns an object of the same class as the original; can be used to select more than on elements (there is one exception)
* [[- is used to extract elements of a list or a data frame; it can only be used to extract a single element and the class of the returned object will not necessarily be a list or data frame
* $ - is used to extract elements of a list or data frame by name, semantics are similar to hat of [[.

x <- c("a","b","c", "c","d","a") * character vector called x

* > x[1]
* [1] "a"
* > x[2]
* [1] "b"
* > x[1:4]
* [1] "a" "b" "c" "c"

x[x>"a"] # subset that gets every element greater than "a"

[1] "b" "c" "c" "d"

u <- x > "a" # create a logical vector called "u"

> u
[1] FALSE TRUE TRUE TRUE TRUE FALSE

x[u] # subet the vector "x" with this u vector and get out all elements that are greater than "a"

[1] "b" "c" "c" "d"

Basics (cont)

So in conclusion there are 2 types of indices that where used above
* the first type with the numeric index
* the second type was the logical index

Removing missing values

A common task is to remove missing values (NAS)

create a logical vector which tells you where the NA's are so you can remove them

> x <- c(1,2,NA,4,NA,5) * here we have a vector x
> bad <- is.na(x) * tell me which elements are na and stores the in the bad vector
> x[!bad] * give me the elements that are NOT missing or NA
[1] 1 2 4 5

What if there are multiple things and you want to take the subset with no missing values?

> x <- c(1,2,NA,4,NA,5)
> y <- c("a","b", NA, "d", NA, "f")
> good <- complete.cases(x,y)
> good
[1] TRUE TRUE FALSE TRUE FALSE TRUE
> x[good] * subset x
[1] 1 2 4 5
> y[good] * subset y
[1] "a" "b" "d" "f"

Lists

The nice thing about being able to subset an element using its name is that you don't have to remember where it is in the list

> x <- list(foo = 1:4, bar = 0.6) # list of 2 elements foo and bar
> x
$foo
[1] 1 2 3 4
$bar
[1] 0.6

Lists (cont)

> x[1]* list that contained the sequence 1 thru 4
$foo
[1] 1 2 3 4

> x[[1]] # just the sequence
[1] 1 2 3 4

> x$bar # give me the element that is associated with the name bar
[1] 0.6

> x[["bar"]]
[1] 0.6

if you want to extract multiple elements of a list then you need to use the single bracket

> x <- list(foo = 1:4, bar = 0.6, baz = "hello")
> x[c(1,3)] # give me the 1st and 3rd element of the vector x
$foo
[1] 1 2 3 4
$baz
[1] "hello"

[[to index a list where the index itself was computed

> x <- list(foo = 1:4, bar = 0.6, baz = "hello")
> name <- "foo"
> x[[name]] * computed index for 'foo'
[1] 1 2 3 4
> x$name * element 'name' doesn't exist
NULL
> x$foo * element 'foo' does exist
[1] 1 2 3 4

The [[indicator can take an interger sequence

> x <- list(a = list(10,12,14), b = c(3.14,2.81))
* I want to exract 14, that is really the 3rd element of the 1st element so its the 3rd element of the list which happens to be the first element of the other list.
> x[[c(1,3)]]
[1] 14
> x[[1]][[3]]
[1] 14
> x[[c(2,1)]] * extract the first element of the second element by passing the vector 2,1
[1] 3.14

C

By **bwaldo**
cheatography.com/bwaldo/

Matrices

Matrices can be subsetted in the usual way with (i,j) type indices

> x <- matrix(1:6, 2, 3) # create a 2x3 matrix with the number sequence of 1 thru 6
> x[1,2] # give me the first row and second column
[1] 3
> x[2,1]
[1] 2

Indices can also be missing

> x[1,] * i want the entire first row
[1] 1 3 5
> x[,2] * i want just the second column
[1] 3 4

By default, when a single element of a matrix is retrieved, it is returned as a vector of length 1 rather than a 1x1 matrix. This behavior can be turned off by setting drop = FALSE.

> x <- matrix (1:6,2,3)
> x[1,2]
[1] 3
> x[1,2, drop = FALSE] * this preserves the dimension of the object
[,1]
[1,] 3

Similarly, subsetting a single column or a single row will give you a vector, not a matrix (by default)

> x <- matrix (1:6,2,3)
> x[1,]
[1] 1 3 5
> x[1,,drop=FALSE]
[,1] [,2] [,3]
[1,] 1 3 5

Partial Matching

Partial matching of names is allowed with [[and $.

> x <- list(aardvark = 1:5) # create a list x which as the element aardvark that is a seq 1 thru 5
> x$a * instead of typing aardvark everytime, search for a name in the list that matches 'a'
[1] 1 2 3 4 5
> x[["a"]] * the [[expects a name with an exact match, so no partial matching
NULL
> x[["a", exact = FALSE]] * specify exact = FALSE then the return will be below.
[1] 1 2 3 4 5

Vectorized operations

Many operations in R are vectorized making code more efficient, concise, and easier to read

> x <- 1:4; y <- 6:9 * 2 vectors
> x + y * add the 1st element of x to the 1st element of 2 etc (1+6, 2+7 etc)
[1] 7 9 11 13
> x > 2 [1] FALSE FALSE TRUE TRUE
> x >= 2
[1] FALSE TRUE TRUE TRUE
> y == 8
[1] FALSE FALSE TRUE FALSE
> x * y
[1] 6 14 24 36
> x / y
[1] 0.1666667 0.2857143 0.3750000 0.4444444

Similary you can do the same with matrices

> x <- matrix(1:4, 2, 2); y <- matrix(rep(10,4), 2, 2)
* x is a matrix 1 thru 4 so its a 2x2 matrix
* y is a matrix of all 10's its also a 2x2 matrix

Vectorized operations (cont)

> x *y * **element-wise multiplication***
[,1] [,2]
[1,] 10 30
[2,] 20 40
> x / y
[,1] [,2]
[1,] 0.1 0.3
[2,] 0.2 0.4
> x %% y * **true matrix multiplication**
[,1] [,2]
[1,] 40 40
[2,] 60 60

C

By **bwaldo**
cheatography.com/bwaldo/

Published 27th February, 2017.
Last updated 27th February, 2017.