Subsetting

Lecture 04

Dr. Colin Rundel

Subsetting in General

R has three subsetting operators (`[`, `[[`, and `$`). The behavior of these operators depends on the object (class) they are being used with.

In general there are 6 different types of subsetting that can be performed:

- Positive integer
- Negative integer
- Logical value
- Empty / NULL
- Zero valued
- Character value (names)

Positive Integer subsetting

Returns elements at the given location(s)

```
1 x = c(1,4,7)
```

```
1 x[1]
```

```
[1] 1
```

```
1 x[c(1,3)]
```

```
[1] 1 7
```

```
1 x[c(1,1)]
```

```
[1] 1 1
```

```
1 x[c(1.9,2.1)]
```

```
[1] 1 4
```

```
1 y = list(1,4,7)
```

```
1 str( y[1] )
```

```
List of 1
 $ : num 1
```

```
1 str( y[c(1,3)] )
```

```
List of 2
 $ : num 1
 $ : num 7
```

```
1 str( y[c(1,1)] )
```

```
List of 2
 $ : num 1
 $ : num 1
```

```
1 str( y[c(1.9,2.1)] )
```

```
List of 2
 $ : num 1
 $ : num 4
```

Negative Integer subsetting

Excludes elements at the given location(s)

```
1 x = c(1,4,7)
```

```
1 x[-1]
```

```
[1] 4 7
```

```
1 x[-c(1,3)]
```

```
[1] 4
```

```
1 x[c(-1,-1)]
```

```
[1] 4 7
```

```
1 y = list(1,4,7)
```

```
1 str( y[-1] )
```

```
List of 2
 $ : num 4
 $ : num 7
```

```
1 str( y[-c(1,3)] )
```

```
List of 1
 $ : num 4
```

```
1 x[c(-1,2)]
```

```
Error in x[c(-1, 2)]: only 0's may be mixed with negative subscripts
```

```
1 y[c(-1,2)]
```

```
Error in y[c(-1, 2)]: only 0's may be mixed with negative subscripts
```

Logical Value Subsetting

Returns elements that correspond to `TRUE` in the logical vector. Length of the logical vector is coerced to be the same as the vector being subsetted.

```
1 x = c(1,4,7,12)
```

```
1 x[c(TRUE,TRUE,FALSE,TRUE)]
```

```
[1]  1  4 12
```

```
1 x[c(TRUE,FALSE)]
```

```
[1] 1 7
```

```
1 x[x %% 2 == 0]
```

```
[1]  4 12
```

```
1 y = list(1,4,7,12)
```

```
1 str( y[c(TRUE,TRUE,FALSE,TRUE)] )
```

```
List of 3
 $ : num 1
 $ : num 4
 $ : num 12
```

```
1 str( y[c(TRUE,FALSE)] )
```

```
List of 2
 $ : num 1
 $ : num 7
```

```
1 str( y[y %% 2 == 0] )
```

```
Error in y%%2: non-numeric argument to bin
```

Empty Subsetting

Returns the original vector, this is not the same as subsetting with `NULL`

```
x = c(1,4,7)
```

```
x[]
```

```
[1] 1 4 7
```

```
x[NULL]
```

```
numeric(0)
```

```
y = list(1,4,7)
```

```
str(y[])
```

```
List of 3
 $ : num 1
 $ : num 4
 $ : num 7
```

```
str(y[NULL])
```

```
list()
```

Zero subsetting

Returns an empty vector (of the same type), this is the same as subsetting with `NULL`

```
x = c(1,4,7)
```

```
x[0]
```

```
numeric(0)
```

```
y = list(1,4,7)
str(y[0])
```

```
list()
```

`0`s can be mixed with either positive or negative integers for subsetting, but they are ignored in both cases.

```
x[c(0,1)]
```

```
[1] 1
```

```
y[c(0,1)]
```

```
[[1]]
[1] 1
```

```
x[c(0,-1)]
```

```
[1] 4 7
```

```
y[c(0,-1)]
```

```
[[1]]
[1] 4

[[2]]
[1] 7
```

Character subsetting

If the vector has names, selects elements whose names correspond to the values in the name vector.

```
1 x = c(a=1, b=4, c=7)
```

```
1 x["a"]
```

```
a
1
```

```
1 x[c("a","a")]
```

```
a a
1 1
```

```
1 x[c("b","c")]
```

```
b c
4 7
```

```
1 y = list(a=1,b=4,c=7)
```

```
1 str(y["a"])
```

```
List of 1
 $ a: num 1
```

```
1 str(y[c("a","a")])
```

```
List of 2
 $ a: num 1
 $ a: num 1
```

```
1 str(y[c("b","c")])
```

```
List of 2
 $ b: num 4
 $ c: num 7
```

Out of bounds

```r
x = c(1,4,7)
```

```r
x[4]
```

```
[1] NA
```

```r
x[-4]
```

```
[1] 1 4 7
```

```r
x["a"]
```

```
[1] NA
```

```r
x[c(1,4)]
```

```
[1]  1 NA
```

```r
y = list(1,4,7)
```

```r
str(y[4])
```

```
List of 1
 $ : NULL
```

```r
str(y[-4])
```

```
List of 3
 $ : num 1
 $ : num 4
 $ : num 7
```

```r
str(y["a"])
```

```
List of 1
 $ : NULL
```

```r
str(y[c(1,4)])
```

```
List of 2
 $ : num 1
 $ : NULL
```

Missing values

```
1 x = c(1,4,7)
```

```
1 x[NA]
```

```
[1] NA NA NA
```

```
1 x[c(1,NA)]
```

```
[1]  1 NA
```

```
1 y = list(1,4,7)
```

```
1 str(y[NA])
```

```
List of 3
 $ : NULL
 $ : NULL
 $ : NULL
```

```
1 str(y[c(1,NA)])
```

```
List of 2
 $ : num 1
 $ : NULL
```

NULL and empty vectors (length 0)

This final type of subsetting follows the rules for length coercion with a 0-length vector (i.e. the vector being subset gets coerced to having length 0 if the subsetting vector has length 0)

```
1 x = c(1,4,7)
```

```
1 x[NULL]
```

```
numeric(0)
```

```
1 x[integer()]
```

```
numeric(0)
```

```
1 x[character()]
```

```
numeric(0)
```

```
1 y = list(1,4,7)
```

```
1 y[NULL]
```

```
list()
```

```
1 y[integer()]
```

```
list()
```

```
1 y[character()]
```

```
list()
```

Subsetting and assignment

Subsets can also be used with assignment to update specific values within an object (in-place).

```
1 x = c(1, 4, 7, 9, 10, 15)
```

```
1 x[2] = 2
2 x
```

```
[1]  1  2  7  9 10 15
```

```
1 x %% 2 != 0
```

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

```
1 x[x %% 2 != 0] = (x[x %% 2 != 0] + 1) / 2
2 x
```

```
[1]  1  2  4  5 10  8
```

```
x[c(1,1)] = c(2,3)
x
```

```
[1]  3  2  4  5 10  8
```

```
x = 1:6
```

```
x[c(2,NA)] = 1
x
```

```
[1] 1 1 3 4 5 6
```

```
x = 1:6
```

```
x[c(-1,-2)] = 3
x
```

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

```
x = 1:6
```

```
x[c(TRUE,NA)] = 1
x
```

```
[1] 1 2 1 4 1 6
```

```
x = 1:6
```

```
x[] = 1:3
x
```

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

The other subset operators

[[and $

Atomic vectors - [vs. [[

`[[` subsets like `[` except it can only subset for a *single* value

```
1 x = c(a=1,b=4,c=7)
```

```
1 x[1]
```

```
a
1
```

```
1 x[[1]]
```

```
[1] 1
```

```
1 x[["a"]]
```

```
[1] 1
```

```
1 x[[1:2]]
```

```
Error in x[[1:2]]: attempt to select more than one element in vectorIndex
```

```
1 x[[TRUE]]
```

```
[1] 1
```

Generic Vectors (lists) - [vs. [[

Subsets a single value, but returns the value - not a list containing that value. Multiple values are interpreted as nested subsetting.

```
1 y = list(a=1, b=4, c=7:9)
```

```
1 y[2]
```

```
$b
[1] 4
```

```
1 str( y[2] )
```

```
List of 1
 $ b: num 4
```

```
1 y[[2]]
```

```
[1] 4
```

```
1 y[["b"]]
```

```
[1] 4
```

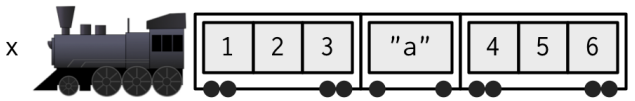
```
1 y[[1:2]]
```

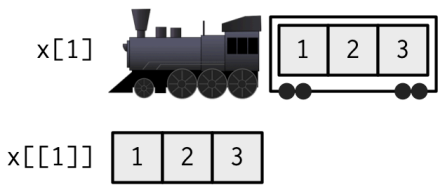
```
Error in y[[1:2]]: subscript out of bounds
```

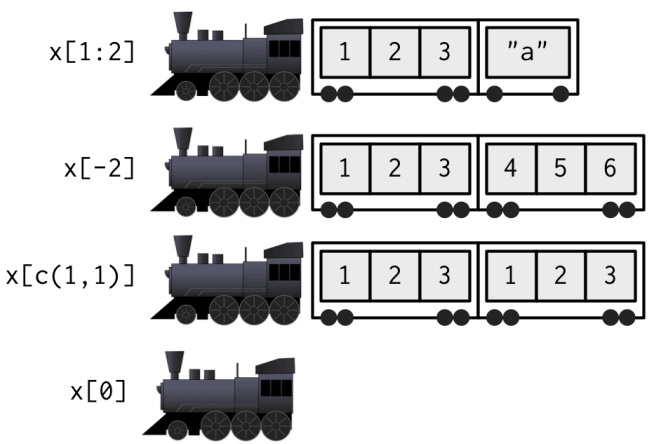
```
1 y[[2:1]]
```

```
[1] 4
```

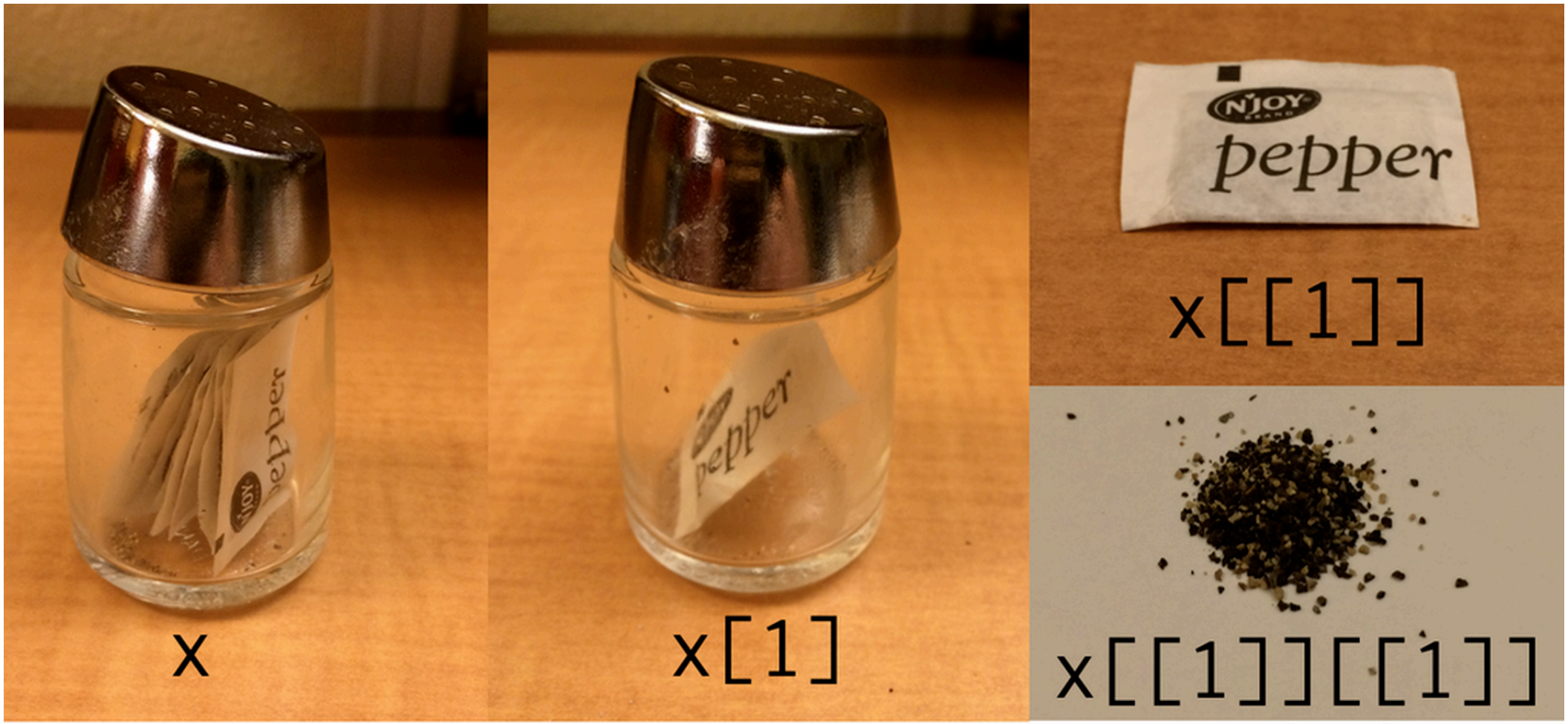
Hadley's Analogy (1)







Hadley's Analogy (2)



Hadley Wickham @hadleywickham · 6h

Indexing lists in #rstats. Inspired by the Residence Inn

273 370

[[vs. $

`$` is equivalent to `[[` but it only works for name based subsetting of *lists* (it also uses partial matching for names)

```
1 x = c("abc"=1, "def"=5)
```

```
1 x$abc
```

```
Error in x$abc: $ operator is invalid for atomic vectors
```

```
1 y = list("abc"=1, "def"=5)
```

```
1 y[["abc"]]
```

```
[1] 1
```

```
1 y$abc
```

```
[1] 1
```

```
1 y$d
```

```
[1] 5
```

A common error

Why does the following code not work?

```
x = list(abc = 1:10, def = 10:1)
y = "abc"
```

```
x[[y]]
```

```
x$y
```

```
 [1]  1  2  3  4  5  6  7  8  9 NULL
```

The expression `x$y` gets interpreted as `x[["y"]]` by R, note the inclusion of the `"`s, this is not the same as the expression `x[[y]]`.