R split numeric vector at position

splitAt <- function(x, pos){</pre>

 $a \leftarrow c(1, 2, 2, 3)$

> splitAt(a, 4)

list(x[1:pos-1], x[pos:length(x)])

Asked 7 years ago Active 3 years, 10 months ago Viewed 7k times



I am wondering about the simple task of splitting a vector into two at a certain index:

```
20
```

13

[1] 1 2 2 [[2]]

[[1]]

[1] 3

My question: There must be some existing function for this, but I can't find it? Is maybe split a possibility? My naive implementation also does not work if pos=0 or pos>length(a).

```
vector
           split
```

asked May 3 '13 at 11:33 user1981275 10.9k 5

3 Answers





An improvement would be:

27

splitAt <- function(x, pos) unname(split(x, cumsum(seq_along(x) %in% pos)))</pre>



which can now take a vector of positions:



```
splitAt(a, c(2, 4))
# [[1]]
# [1] 1
# [[2]]
#[1] 2 2
# [[3]]
# [1] 3
```

And it does behave properly (subjective) if $pos \le 0$ or pos >= length(x) in the sense that it returns the whole original vector in a single list item. If you'd like it to error out instead, use stopifnot at the top of the function.

answered May 3 '13 at 11:41



This function is very slow with very large x, probably due to the $seq_along(x)$ that creates a very long vector and then the %in% that has to match this very long vector. — Calimo Oct 9 '13 at 13:42

@Calimo: no, if you profile it, you'll see that most of the time is spent inside the slowish split. You can certainly avoid it but you'll lose a lot in terms of readability and code compactness. – flodel Oct 10 '13 at 0:00



I tried to use <u>flodel's answer</u>, but it was too slow in my case with a very large \times (and the function has to be called repeatedly). So I created the following function that is much faster, but also very ugly and doesn't behave properly. In particular, it doesn't check anything and will return buggy results at least for pos >= length(x) or pos <= 0 (you can add those checks yourself if you're unsure about your inputs and not too concerned about speed), and perhaps some other cases as well, so be careful.



```
splitAt2 <- function(x, pos) {
   out <- list()
   pos2 <- c(1, pos, length(x)+1)
   for (i in seq_along(pos2[-1])) {
      out[[i]] <- x[pos2[i]:(pos2[i+1]-1)]
   }
   return(out)
}</pre>
```

However, splitAt2 runs about 20 times faster with an x of length 10⁶:

edited May 23 '17 at 12:18

Community ◆

1 1

answered Oct 9 '13 at 14:08



Thanks! Also with the simple example from above, splitAt2 performs better. - user1981275 Oct 9 '13 at 14:47

4 +1-a somewhat pretty rewrite could be: function(x, pos) {pos <- c(1L, pos, length(x) + 1L); Map(function(x, i, j) x[i:j], list(x), head(pos, -1L), tail(pos, -1L) - 1L)}. It also seems a bit faster as the number of splits increases, not sure why. — flodel Oct 10 '13 at 0:33

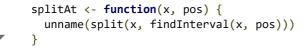
@user1981275 define "better". If better = faster I agree, but as a general purpose function robustness is key, in which case flodel's version is better. — Calimo Oct 10 '13 at 7:15

@flodel indeed your rewrite is faster with a very large number of splits. Can't explain why either. – Calimo Oct 10 '13 at 7:16



Another alternative that might be faster and/or more readable/elegant than flodel's solution:













Joshua Ulrich 154k 29 302 385

