How to see the source code of R .Internal or .Primitive function?

Asked 7 years, 5 months ago Active 2 months ago Viewed 24k times

77

41

Neither of these show the source code of `pnorm` function,

```
stats:::pnorm
getAnywhere(pnorm)
```

How can i see the source code of `pnorm` ?

```
sum
 (..., na.rm = FALSE)  .Primitive("sum")
.Primitive("sum")
function (..., na.rm = FALSE)  .Primitive("sum")
methods(sum)
no methods were found
```

and, how can I see source code of the `sum` function?

r

edited Aug 17 '15 at 15:04
Mogsdad
37.7k 13 120 232

asked Dec 26 '12 at 2:49
Bqsj Sjbq
1,121 1 9 9

3 Answers

Active Oldest Votes

91

The R source code of `pnorm` is:

```
function (q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
.Call(C_pnorm, q, mean, sd, lower.tail, log.p)
```

So, technically speaking, typing "pnorm" *does* show you the source code. However, more usefully: The guts of `pnorm` are coded in C, so the advice in the previous question view source code in R is only peripherally useful (most of it concentrates on functions hidden in namespaces etc.).

Uwe Ligges's article in R news (p. 43) is a good general reference. From that document:

> When looking at R source code, sometimes calls to one of the following functions show up: .C(), .Call(), .Fortran(), .External(), or .Internal() and .Primitive(). These functions are calling entry points in compiled code such as shared objects, static libraries or dynamic link libraries. Therefore, it is necessary to look into the sources of the compiled code, if complete understanding of the code is required. ... The first step is to look up the entry point in file '$R HOME/src/main/names.c', if the calling R function is either .Primitive() or

Depending on how seriously you want to dig into the code, it may be worth downloading and unpacking the source code as Ligges suggests (for example, then you can use command-line tools such as `grep` to search through the source code). For more casual inspection, you can view the sources online via the R Subversion server or Winston Chang's github mirror (links here are specifically to `src/nmath/pnorm.c`). (Guessing the right place to look, `src/nmath/pnorm.c`, takes some familiarity with the structure of the R source code.)

`mean` and `sum` are both implemented in summary.c.

edited May 23 '17 at 12:02
Community ♦
1 1

answered Dec 26 '12 at 3:07
Ben Bolker
156k 18 276 354

1 it's in a different category from `pnorm`. Try `mean.default` for the R code, and github.com/wch/r-source/blob/trunk/src/main/summary.c for the C code. And do read Uwe Ligges's article linked above! – Ben Bolker Dec 26 '12 at 3:36

1 Just to follow up on this answer: might need to be careful about the exact function name in C or Fortran too. Example: I was trying to look up the source for `stl`, which calls this line : `z <- .Fortran(C_stl, x, n, as.integer(period), as.integer(s.window)`. So i searched the Github mirror linked above by the keyword `C_stl` to no avail. However when I search `stl` there is a file called `stl.f` which is what I want to find. The takeaway is the .c or .f file name might not be exactly the same as the function name that is being called. – yuqli Aug 14 '18 at 17:08

33

I know this post is more that 2 years old, but I thought this might be useful to some users browsing through this question.

I'm basically just copying my answer to this other similar question so that it can maybe prove useful to some R users who want to explore the C source files.

1. First, with **pryr** you can use the `show_c_source` function which will search on GitHub the relevant piece of code in the C source files. Works for .Internal and .Primitive functions.

```
body(match.call)

# .Internal(match.call(definition, call, expand.dots))

pryr::show_c_source(.Internal(match.call(definition, call, expand.dots)))
```

   Which takes you to this page, showing that `unique.c` contains the function do_matchcall.

2. I've put together this tab delimited file, building on the `names.c` file and using *find-in-files* to determine the location of the source code. There are some functions that have platform-specific files, and a handful of others for which there is more than one file with relevant source code. But for the rest the mapping is pretty well established, at least for the current version (3.1.2).

edited Mar 28 at 1:02

answered Apr 17 '15 at 10:49
Dominic Comtois
9,070 31 50

the tab-delimited file works. thanks. – zhanxw Sep 25 '19 at 21:29

6

```
> methods(mean)
[1] mean.data.frame mean.Date     mean.default   mean.difftime  mean.IDate*
[6] mean.POSIXct    mean.POSIXlt   mean.yearmon*  mean.yearqtr*

   Non-visible functions are asterisked
> mean.default
function (x, trim = 0, na.rm = FALSE, ...)
{
    if (!is.numeric(x) && !is.complex(x) && !is.logical(x)) {
        warning("argument is not numeric or logical: returning NA")
        return(NA_real_)
    }
    if (na.rm)
        x <- x[!is.na(x)]
    if (!is.numeric(trim) || length(trim) != 1L)
        stop("'trim' must be numeric of length one")
    n <- length(x)
    if (trim > 0 && n) {
        if (is.complex(x))
            stop("trimmed means are not defined for complex data")
        if (any(is.na(x)))
            return(NA_real_)
        if (trim >= 0.5)
            return(stats::median(x, na.rm = FALSE))
        lo <- floor(n * trim) + 1
        hi <- n + 1 - lo
        x <- sort.int(x, partial = unique(c(lo, hi)))[lo:hi]
    }
    .Internal(mean(x))
}
<bytecode: 0x155ef58>
<environment: namespace:base>
```

answered Dec 26 '12 at 3:34

IRTFM
231k 18 306 432

This seems not to answer the OP's original question (about `pnorm`), but their comment below about `mean` -- and note that this falls through into C code as well, at the bottom (see my comment below). – Ben Bolker Dec 26 '12 at 3:37

2 Indeed. And the "correct answer "is the one you gave earlier ... read Uwe Ligges' article in RNews. – IRTFM Dec 26 '12 at 3:57