

C for R users

Ella Kaye

useR! 2024, Salzburg

Background

- Research Software Engineer at University of Warwick
- Sustainability and EDI in the R Project (with Heather Turner)

Fostering a larger, more diverse
community of contributors to base R

This talk

What I'll do

- Encourage you to learn C
- Show you some C code in base R
- Encourage you to contribute to base R

What I won't do

- Assume you know any C
- Try to teach you any C

What is C and how does it relate to R?

- C is a low-level, high-performance, compiled programming language
- It provides fine-grained control over memory and hardware
- Much of base R is written in C
- R provides interfaces to compiled code
- R has a C API to deal with R objects in C

Why C?

As R users/developers

- Write efficient, portable code
- Encounter C code when debugging

As R contributors

- Find root cause of bug
- Propose a patch to the C code to fix a bug

Writing high-performance code

Limits of R

Sometimes you reach the limits of R:

- Your code is still slow despite optimizing the computational approach and the R implementation
- You *could* speed up the R code, but it results in very obscure, convoluted code

In these cases it can make sense to code parts in C or C++ and call it from R.

Typical scenarios

- Loops that can't be vectorized because iterations depend on previous results
- Recursive functions, or problems which involve calling functions millions of times
- Problems that require advanced data structures and algorithms that R doesn't provide

You *almost certainly* want C++ with Rcpp

- Protects you from many of the historical idiosyncracies of R's C API
- Takes care of memory management
- Provides many useful helper methods

But you *might* want/need C

- Portability (e.g. can also call from Python)
- Building on other people's C code

Digging into a bug

Irregularity in `stem()` display

https://bugs.r-project.org/show_bug.cgi?id=8934

```
1 a <- c(8.48, 9.58, 9.96)
2 stem(a)
```

The decimal point is at the |

```
8 |
8 | 5
9 |
9 | 6
10 | 0
```

```
1 stem(2)
2 stem(c(2, 2))
```

The decimal point is at the |

```
2 | 00
```

Check the code

```
1 stem
1 function (x, scale = 1, width = 80, atom = 1e-08)
2 {
3     if (!is.numeric(x))
4         stop("'x' must be numeric")
5     x <- x[is.finite(x)]
6     n <- as.integer(length(x))
7     if (is.na(n))
8         stop("invalid length(x)")
9     if (n == 0)
10         stop("no finite and non-missing values")
11     if (scale <= 0)
12         stop("'scale' must be positive")
13     .Call(C_StemLeaf, as.double(x), scale, width, atom)
14     invisible(NULL)
15 }
```

There's C!

```
.Call(C_StemLeaf, as.double(x), scale, width, atom)
```

```
1 C_StemLeaf
```

```
Error in eval(expr, envir, enclos): object 'C_StemLeaf' not found
```

Where's C?

github.com/r-devel/r-svn

Q repo:r-devel/r-svn C_StemLeaf

Search in this repository

Q org:r-devel C_StemLeaf

Search in this organization

Q C_StemLeaf

Search all of GitHub

Code

<>

function C_StemLeaf

r-devel/r-svn · src/library/graphics/src/graphics.h

Jump to

<>

function C_StemLeaf

r-devel/r-svn · src/library/graphics/src/stem.c

Jump to

Search syntax tips

Give feedback

C_StemLeaf()

R's C API

```
1 SEXP C_StemLeaf(SEXP x, SEXP scale, SEXP swidth, SEXP atom)
2 {
3     if (typeof(x) != REALSXP || typeof(scale) != REALSXP)
4         error("invalid input");
5     #ifdef LONG_VECTOR_SUPPORT
6         if (IS_LONG_VEC(x))
7             error(_("long vector '%s' is not supported"), "x");
8     #endif
9     int width = asInteger(swidth), n = LENGTH(x);
10    if (n == NA_INTEGER) error(_("invalid '%s' argument"), "x");
11    if (width == NA_INTEGER) error(_("invalid '%s' argument"), "width");
12    double sc = asReal(scale), sa = asReal(atom);
13    if (!R_FINITE(sc)) error(_("invalid '%s' argument"), "scale");
14    if (!R_FINITE(sa)) error(_("invalid '%s' argument"), "atom");
15    stem_leaf(REAL(x), n, sc, width, sa);
16    return R_NilValue;
17 }
```


stem_leaf()

```
1 static Rboolean
2 stem_leaf(double *x, int n, double scale, int width, double atom)
3 {
4     // <initialise variables>
5
6     R_rsort(x,n);
7
8     if (n <= 1) return FALSE;
9
10    //<more code here>
11
12    /* Find the print width of the stem. */
13
14    lo = floor(x[0]*c/mu)*mu;
15    hi = floor(x[n-1]*c/mu)*mu;
16    ldigits = (lo < 0) ? (int) floor(log10(-(double)lo)) + 1 : 0;
17    hdigits = (hi > 0) ? (int) floor(log10((double)hi)): 0;
18    ndigits = (ldigits < hdigits) ? hdigits : ldigits;
```

A note about interfaces

We've seen `.Call()`

In base R, there's also `.Internal()` and `.Primitive()`

e.g. the source code for `tabulate` includes:

```
1 .Internal(tabulate(bin, nbins))
```

We can find the underlying code on GitHub with

```
1 pryr::show_c_source(.Internal(tabulate(bin, nbins)))
```

Contributing

3-digit hex case-study

The original idea

Want, e.g. "#ba1" to be interpreted as "#bbaa11".



[coolbutuseless](#)

@coolbutuseless@fosstodon.org

Free [#RStats](#) internals idea:

Add support for 3-digit hex colours e.g. "[#ba1](#)"

File to change: src/library/grDevices/src/color.c

Need to

- * add a "case 4:" to the switch statement
- * Write 4 lines of code. (including a break statement)
- * Change Line:1365 to include case when strlen = 4

Anyone else annoyed by this?

```
18      /* #RRGGBB[AA] String to Internal Color Code */
1349     static rcolor rgb2col(const char *rgb)
1350     {
```

The existing code

```
1  /* #RRGGBB[AA] String to Internal Color Code */
2  static rcolor rgb2col(const char *rgb)
3  {
4      unsigned int r = 0, g = 0, b = 0, a = 0; /* -Wall */
5      if(rgb[0] != '#')
6          error(_("invalid RGB specification"));
7      switch (strlen(rgb)) {
8      case 9:
9          a = 16 * hexdigit(rgb[7]) + hexdigit(rgb[8]);
10     case 7:
11         r = 16 * hexdigit(rgb[1]) + hexdigit(rgb[2]);
12         g = 16 * hexdigit(rgb[3]) + hexdigit(rgb[4]);
13         b = 16 * hexdigit(rgb[5]) + hexdigit(rgb[6]);
14         break;
15     default:
16         error(_("invalid RGB specification"));
17     }
18
19     if(strlen(rgb) == 7)
20         return R_RGB(r, g, b);
21     else
22         return R_RGBA(r, g, b, a);
23 }
```

The fix: part 1

```
1  switch (strlen(rgb)) {
2  case 9:
3      a = 16 * hexdigit(rgb[7]) + hexdigit(rgb[8]);
4  case 7:
5      r = 16 * hexdigit(rgb[1]) + hexdigit(rgb[2]);
6      g = 16 * hexdigit(rgb[3]) + hexdigit(rgb[4]);
7      b = 16 * hexdigit(rgb[5]) + hexdigit(rgb[6]);
8      break;
9  case 5:
10     // Equivalent to 16 * hexdigit(rgb[4]) + hexdigit(rgb[4]);
11     a = (16 + 1) * hexdigit(rgb[4]);
12 case 4:
13     r = (16 + 1) * hexdigit(rgb[1]);
14     g = (16 + 1) * hexdigit(rgb[2]);
15     b = (16 + 1) * hexdigit(rgb[3]);
16     break;
17 default:
18     error(_("invalid RGB specification"));
19 }
```

The fix: part 2

From

```
1  if(strlen(rgb) == 7)
2      return R_RGB(r, g, b);
3  else
4      return R_RGBA(r, g, b, a);
```

to

```
1  switch(strlen(rgb)) {
2  case 7:
3  case 4:
4      return R_RGB(r, g, b);
5  default:
6      return R_RGBA(r, g, b, a);
7  }
```


Learning more

C study group

<https://contributor.r-project.org/events/c-study-group-2024/>

- Will run again January–June 2025, details TBC
- Monthly meetings, weekly suggestions
- Work through sessions 1-5 of Harvard's CS50 course
cs50.harvard.edu/x
- R's C API
- Run by R Contribution Working Group (RCWG)

RCWG

Fosters a larger, more diverse community of contributors to base R.

- contributor.r-project.org
- meetup.com/r-contributors
- hachyderm.io/@R_Contributors

Resources: R's C API

- Deep R: <https://deepr.gagolewski.com/chapter/310-compiled.html>
- Advanced R, Hadley Wickham, (1st edn): <http://adv-r.had.co.nz/C-interface.html>
- Now You C Me, Davis Vaughan: <https://blog.davisvaughan.com/posts/2019-03-02-now-you-c-me>
- Writing R Extensions (Ch 5 and 6): <https://cran.stat.auckland.ac.nz/doc/manuals/r-devel/R-exts.html>
- R internals (Hadley Wickham): <https://github.com/hadley/r-internals>
- R internals (R Core): <https://cran.stat.auckland.ac.nz/doc/manuals/r-devel/R-ints.html>

Thank you! Questions?

ella.m.kaye@warwick.ac.uk

ellakaye.github.io/c-for-r-users

