Stat 8054 Lecture Notes: R Standalone Library

Charles J. Geyer

April 26, 2020

1 License

2 R

- The version of R used to make this document is 4.0.0.
- The version of the `rmarkdown` package used to make this document is 2.1.

3 Reading

- Section 9 The standalone Rmath library in the book *R Installation and Administration* that can be found on CRAN or in your current installation of R. For the latter type `help.start()` in R and navigate in the web browser that pops up to this section of this book.
- Section 6.17 Using these functions in your own C code in the book *Writing R Extensions* that can be found on CRAN or in your current installation of R. For the latter type `help.start()` in R and navigate in the web browser that pops up to this section of this book.
- A Makefile Tutorial for those who want to learn about the unix program `make` and makefiles (which is not necessary for understanding this handout).

4 The Standalone Library

This handout is about using R functions from C code (or C++) that is *not* called from R. We just have a C main program. The R interpreter is not running. No R expressions are evaluated.

What we do have is some C functions that R uses to do computations are available in a C library and can be called from our C code.

We will just do some examples. The full API can be found in the header file `Rmath.h` found in the `include` subdirectory of the directory where R is installed, which is two directories up from what `R RHOME` says. That is, on unix, the command

```
ls `R RHOME`/../../include/Rmath.h
```

```
/home/geyer/local/current/lib/R/../../include/Rmath.h
```

shows where this header file is on the computer this Rmarkdown document was processed on.

More information is found in the reading cited above.

5 An Example

For an example we will call the function `pgamma` that calculates the distribution function (DF) of the gamma distribution.

To call this we have to look up the source code for this R function in the R source tree. If you have the R standalone library, then you have probably built R from source. But readers who have not done that can look in the R source tree on-line https://svn.r-project.org/R/branches/R-4-0-branch/src/nmath/pgamma.c (make suitable changes to get to the current version of R, this was the current version when this handout was written).

There we see

- This code is licensed under the GPL version 2 or later versions. So your code, if distributed to others, must have a GPL-compatible license.
- The code includes the following comment (part of voluminous comment at the top of the file)

```
 *  SYNOPSIS
 *
 *  #include <Rmath.h>
 *
 *  double pgamma (double x, double alph, double scale,
 *              int lower_tail, int log_p)
```

which is clear enough. It evaluates the DF of the gamma distribution, and it has five arguments that correspond to the arguments of R function `pgamma`. The only difference is that, unlike the R function, this function does not vectorize. Every argument is scalar. Note that the second parameter of the gamma distribution (here called `scale`) is the scale parameter for R function `pgamma` which is by default `1/rate` but here we do not have defaults (C functions don't) and we don't have `rate`.

Here is a C main program

```
cat my_pgamma.c
```

```
// my_pgamma - evaluate the distribution function of the gamma distribution
//             a demo of the R standalone math library
//
// Written in 2020 by Charles J. Geyer <geyer@umn.edu>
//
// To the extent possible under law, the author(s) have dedicated all
// copyright and related and neighboring rights to this software to
// the public domain worldwide. This software is distributed without
// any warranty.
//
// See the CC0 Public Domain Dedication
// <http://creativecommons.org/publicdomain/zero/1.0/>.

#define MATHLIB_STANDALONE
#include <Rmath.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
```

```
int main(int argc, char *argv[])
{
    if (argc != 4) {
        fputs("Usage: my_pgamma x alpha beta\n"
            "where x is a real number and alpha and beta"
            " are postive real numbers\n"
            "the shape and scale parameters (respectively)\n", stderr);
        return EXIT_FAILURE;
    }

    double x, alpha, beta;

    if (sscanf(argv[1], "%lg", &x) != 1) {
        fputs("failed to read x\n", stderr);
        return EXIT_FAILURE;
    }

    if (sscanf(argv[2], "%lg", &alpha) != 1) {
        fputs("failed to read alpha\n", stderr);
        return EXIT_FAILURE;
    }

    if (sscanf(argv[3], "%lg", &beta) != 1) {
        fputs("failed to read beta\n", stderr);
        return EXIT_FAILURE;
    }

    double result = pgamma(x, alpha, beta, true, false);

    printf("pgamma(x, alpha, beta) = %g\n", result);

    return EXIT_SUCCESS;
}
```

We make the executable with `make`

```
make --always-make --no-print-directory my_pgamma
```

```
cc -I/home/geyer/local/current/include -Wall -Wextra -c my_pgamma.c
cc -static -L/home/geyer/local/current/lib my_pgamma.o -lRmath -lm -o my_pgamma
```

Here

- the `--always-make` flag tells the program `make` to make my program whether this is needed or not (normally make does not make the program if the result will not be any different from what is already there, that is, when the exectuable `my_pgamma` is newer than the source code `my_pgamma.c`). We need to make it so we can see the commands that make it.
- the `--no-print-directory` just shuts up some irrelevant blather.
- the `-I` flag tells the C compiler where to look for include files (besides the usual places), in this case where to find `Rmath.h`.
- the `-Wall` and `-Wextra` enable more warnings than the default (but the code is clean, the compiler warns about nothing; it did warn about multiple things during development of the program).
- the `-L` flag tells the linker where to look for libraries (besides the usual places), in this case where to

find `libRmath.a`.

- the `-lRmath -lm` flags tell the linker to find functions to link in the following places: first `my_pgamma.o`, second `libRmath.a` (where it finds `pgamma`), and third `libmath.a` (where it finds any math functions in the C standard library called by `pgamma`). `my_pgamma.o -lRmath -lm` must appear in that order. It does not work when they don't.
- the `-static` flag tells the linker to use static linking, so all the code is in the executable. This is not necessary, but simplifies running the program.

Now we exercise the program

```
./my_pgamma 2.3, 4.1, 2.05
```

```
pgamma(x, alpha, beta) = 0.0236923
```

To check that that worked, do it in R.

```
pgamma(2.3, shape = 4.1, scale = 2.05)
```

```
##  [1] 0.02369231
```

Voilà!

See the Makefile and Source Code section below for the `Makefile` and `my_pgamma.c` files.

6 Another Example, This Time with Random Numbers

For an example we will call the function `rgamma` that generates random variates the gamma distribution, the source for which is in the R source tree https://svn.r-project.org/R/branches/R-4-0-branch/src/nmath/rgamma.c.

Same comment as in the previous example about licensing. The comment in the source says

```
 *  SYNOPSIS
 *
 *    #include <Rmath.h>
 *    double rgamma(double a, double scale);
```

which is clear enough. It generates a random variate from the gamma distribution, and it has two arguments that correspond to the arguments of R function `rgamma`. The only difference is that, unlike the R function, this function does not vectorize. Every argument is scalar, and the result is a single value of type `double` so this function does not need an `n` argument for the number of values to generate. As with the other example, note that the second parameter of the gamma distribution (here called `scale`) is the scale parameter for R function `rgamma` which is by default `1/rate` but here we do not have defaults (C functions don't) and we don't have `rate`.

Here is the C main program for this example

```
cat my_rgamma.c
```

```
// my_rgamma - generate random variates from the gamma distribution
//             a demo of the R standalone math library
//
// Written in 2020 by Charles J. Geyer <geyer@umn.edu>
//
// To the extent possible under law, the author(s) have dedicated all
// copyright and related and neighboring rights to this software to
// the public domain worldwide. This software is distributed without
// any warranty.
```

```
//
// See the CC0 Public Domain Dedication
// <http://creativecommons.org/publicdomain/zero/1.0/>.

#define MATHLIB_STANDALONE
#include <Rmath.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <sys/random.h>

// The immediately preceding include and the call to getrandom
// to initialize RNG are linux-specific
// Must do something else on other OS.

int main(int argc, char *argv[])
{
    if (argc != 4) {
        fputs("Usage: my_rgamma n alpha beta\n"
            "where n is a positive integer and alpha and beta"
            " are postive real numbers\n"
            "the shape and scale parameters (respectively)\n", stderr);
        return EXIT_FAILURE;
    }

    unsigned int n;
    double alpha, beta;

    if (sscanf(argv[1], "%u", &n) != 1) {
        fputs("failed to read x\n", stderr);
        return EXIT_FAILURE;
    }

    if (sscanf(argv[2], "%lg", &alpha) != 1) {
        fputs("failed to read alpha\n", stderr);
        return EXIT_FAILURE;
    }

    if (sscanf(argv[3], "%lg", &beta) != 1) {
        fputs("failed to read beta\n", stderr);
        return EXIT_FAILURE;
    }

    unsigned int seed1, seed2;
    size_t buflen = sizeof(unsigned int);

    if (getrandom(&seed1, buflen, GRND_RANDOM) != (ssize_t) buflen ||
        getrandom(&seed2, buflen, GRND_RANDOM) != (ssize_t) buflen) {
        fputs("failed to initialize random number generator\n", stderr);
        return EXIT_FAILURE;
    }

    set_seed(seed1, seed2);
```

```
    for (unsigned int i = 0; i < n; i++)
        printf("%12.10f\n", rgamma(alpha, beta));

    return EXIT_SUCCESS;
}
```

We make the executable with `make`

```
make --always-make --no-print-directory my_rgamma
```

```
cc -I/home/geyer/local/current/include -Wall -Wextra -c my_rgamma.c
cc -static -L/home/geyer/local/current/lib my_rgamma.o -lRmath -lm -o my_rgamma
```

And we have the same comments about this as in the preceding example.

Now we exercise the program

```
./my_rgamma 5 4.1, 2.05
```

```
12.5724333073
11.3069163731
3.2432522789
14.4144133381
7.7648363586
```

```
./my_rgamma 5 4.1, 2.05
```

```
9.3031578306
11.7637018166
7.4488889698
6.5722713414
8.0120609116
```

We see that, because the program initializes the random number generator (RNG) with random starting "seeds", we get different output each time.

Voilà!

There is no point in checking against R because R uses a different RNG by default and uses different initial seeds. And since the seeds we used for our program were very random, there is no way to duplicate them for use by R, even if we switched the RNG R is using to the same one the R standalone library is using.

7 Makefile and Source Code

We did not say what the `Makefile` that enables the `make` program to make my program is because it is rather complicated. It also makes this document and does other things.

That makefile and the two source code files are linked on the course notes page that links to this page. We also link them here.

- http://www.stat.umn.edu/geyer/8054/notes/Makefile
- http://www.stat.umn.edu/geyer/8054/notes/my_pgamma.c
- http://www.stat.umn.edu/geyer/8054/notes/my_rgamma.c