Integrating R and C++

Sam Bowyer

2023-02-02

Ricker simulation in R

```r
rickerSimul <- function(n, nburn, r, y0 = 1){
  y <- numeric(n)
  yx <- y0

  # Burn in phase
  if(nburn > 0){
    for(ii in 1:nburn){
      yx <- r * yx * exp(-yx)
    }
  }

  # Simulating and storing
  for(ii in 1:n){
    yx <- r * yx * exp(-yx)
    y[ii] <- yx
  }

  return( y )
}
```

Question 1

Also written in C++

```
cat ./rickerSimul.c
```

```
## #include <R.h>
## #include <Rinternals.h>
## #include <Rmath.h>
##
## SEXP rickerSimul(SEXP num, SEXP numburn, SEXP rate, SEXP initialPop){
##     double *xys;
##     int n, nburn;
##     double r, y0;
##     SEXP ys;
##
##     n = INTEGER(num)[0];
##     ys = PROTECT(allocVector(REALSXP, n));
##     xys = REAL(ys);
##
##     nburn = INTEGER(numburn)[0];
##     r = REAL(rate)[0];
##     y0 = REAL(initialPop)[0];
##
##     double yx = y0;
##
##     // Burn in phase
##     if(nburn > 0){
##       for(int i = 0; i < nburn; i++){
##         yx = r * yx * exp(-yx);
##       }
##     }
##
##     // Simulating and storing
##     for(int i=0; i < n; i++){
##       yx = r * yx * exp(-yx);
##       xys[i] = yx;
##     }
##
##     UNPROTECT(1);
##
##     return ys;
##   }
```

So compile it

```
system("R CMD SHLIB rickerSimul.c")
```

(it’s made a .o and .so file)

```
ls rickerSimul.*
```

```
## rickerSimul.c
## rickerSimul.o
## rickerSimul.so
```

load the function into r

```
dyn.load("rickerSimul.so")
is.loaded("rickerSimul")
```

```
## [1] TRUE
```

Now call it with `.Call`

```
n = 25L
nburn=0L
r = 5
y0 = 4
.Call("rickerSimul", n, nburn, r, y0)
```

```
##  [1] 0.3663128 1.2697975 1.7833576 1.4986702 1.6742175 1.5692006 1.6336285
##  [8] 1.5945842 1.6184465 1.6039321 1.6127874 1.6073944 1.6106825 1.6086791
## [15] 1.6099002 1.6091561 1.6096096 1.6093333 1.6095017 1.6093990 1.6094616
## [22] 1.6094235 1.6094467 1.6094326 1.6094412
```

Same output as w/ R implementation

```
rickerSimul(n, nburn, r, y0)
```

```
##  [1] 0.3663128 1.2697975 1.7833576 1.4986702 1.6742175 1.5692006 1.6336285
##  [8] 1.5945842 1.6184465 1.6039321 1.6127874 1.6073944 1.6106825 1.6086791
## [15] 1.6099002 1.6091561 1.6096096 1.6093333 1.6095017 1.6093990 1.6094616
## [22] 1.6094235 1.6094467 1.6094326 1.6094412
```

Benchmark

```
rickerSimulR <- function() rickersimul(100000L, 10000L, r, y0)
rickerSimulC <- function() .Call("rickersimul", 100000L, 10000L, r, y0)

library(microbenchmark)
microbenchmark(rickerSimulR, rickerSimulC, times=1000)
```

```
## Unit: nanoseconds
##          expr min lq   mean median uq max neval
##  rickerSimulR  12 16 17.014     17 17 118  1000
##  rickerSimulC  15 16 17.356     17 17 583  1000
```

Question 2

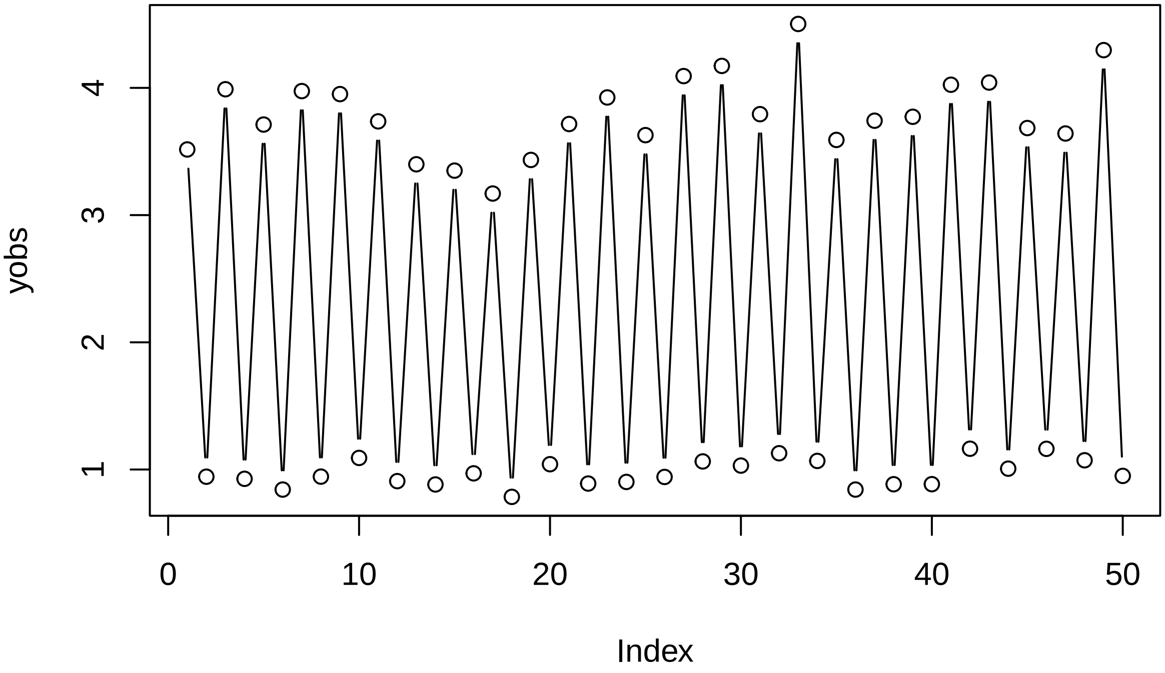
w/ noisy observations $z_t = y_t e^{\epsilon_t}$ where $\epsilon_t \sim N(0, \sigma^2)$.

```
nburn <- 100L
n <- 50L

y0_true <- 1
sig_true <- 0.1
r_true <- 10

Ntrue <- rickerSimul(n = n, nburn = nburn, r = r_true, y0 = y0_true)
yobs <- Ntrue * exp(rnorm(n, 0, sig_true))

plot(yobs, type = 'b')
```



Written a function to calculate the (log) likelihood of the data `rickerLLK.c`

```
cat rickerLLK.c
```

```
## #include <R.h>
## #include <Rinternals.h>
## #include <Rmath.h>
##
## SEXP rickerLLK(SEXP observed, SEXP simulated, SEXP sigma){
##     double *yobs, *ysim;
##     double sig;
##
##     SEXP LLK = PROTECT(allocVector(REALSXP, 1));
##
##     yobs = REAL(observed);
##     ysim = REAL(simulated);
##     sig  = REAL(sigma)[0];
##
##     int n = length(observed);
##
##     double result = 0;
##
##     for (int i = 0; i < n; i++){
##         result += dnorm(yobs - ysim, 0, sig, TRUE);
##     }
##
##     UNPROTECT(1);
##
##     REAL(LLK)[0] = result;
##
##     return LLK;
## }
```

```
system("R CMD SHLIB rickerLLK.c")
dyn.load("rickerLLK.so")
is.loaded("rickerLLK")
```

```
## [1] TRUE
```

Wrap the likelihood calculation in an R function:

```
myLikR <- function(logr, logsig, logy0, yobs, nburn){
  n <- length(yobs)
  r <- exp(logr)
  sig <- exp(logsig)
  y0 <- exp(logy0)

  ysim <- .Call("rickerSimul", n, nburn, r, y0)

  llk <- .Call("rickerLLK", yobs, ysim, sig)

  return( llk )
}

myLikR(log(r_true), log(sig_true), log(y0_true), yobs, nburn)
```

```
## [1] -1.505222e+16
```