Multievent capture-recapture with Rcpp

Olivier Gimenez

August 11, 2017

Download the 'multievent.cpp' and 'titi2.txt' files. This code implements the matrix product by hand.

```
Rcpp::sourceCpp('multievent.cpp')
```

```
##
## > set.seed(1)
##
## > data = read.table("titis2.txt")
##
## > nh <- dim(data)[1]
##
## > k <- dim(data)[2]
##
## > km1 <- k - 1
##
## > eff <- rep(1, nh)
##
## > fc <- NULL
##
## > init.state <- NULL
##
## > for (i in 1:nh) {
## +     temp <- 1:k
## +     fc <- c(fc, min(which(data[i, ] != 0)))
## +     init.state <- c(init.state, data[i, fc[i]])
## + }
##
## > binit <- runif(9)
##
## > data <- t(data)
##
## > devMULTIEVENT <- function(b, data, eff, e, garb, nh,
## +     km1) {
## +     par = plogis(b)
## +     piNB <- par[1]
## +     phiNB <- par[2]
## +     phiB <- pa .... [TRUNCATED]
##
## > devMULTIEVENT(binit, data, eff, fc, init.state, nh,
## +     km1)
## [1] 4119.899
##
## > multieventrcpp(binit, data, fc, init.state)
## [1] 4119.899
##
## > deb = Sys.time()
##
## > tmpmin1 <- optim(binit, devMULTIEVENT, NULL, hessian = T,
```

```
## +     data, eff, fc, init.state, nh, km1, method = "BFGS")
##
## > fin = Sys.time()
##
## > res1 = fin - deb
##
## > deb = Sys.time()
##
## > tmpmin2 <- optim(binit, multieventrcpp, NULL, hessian = T,
## +     data, fc, init.state, method = "BFGS")
##
## > fin = Sys.time()
##
## > res2 = fin - deb
##
## > res1
## Time difference of 10.35423 secs
##
## > tmpmin1$par
## [1]  0.8674528  1.4764532  1.6396689 -1.2691570 -1.2283542  0.2600259
## [7]  0.3959696 -1.4643768  1.0333858
##
## > res2
## Time difference of 1.101992 secs
##
## > tmpmin2$par
## [1]  0.8674528  1.4764532  1.6396689 -1.2691570 -1.2283542  0.2600259
## [7]  0.3959696 -1.4643768  1.0333858
##
## > library(microbenchmark)
##
## > res = microbenchmark(optim(binit, devMULTIEVENT, NULL,
## +     hessian = T, data, eff, fc, init.state, nh, km1, method = "BFGS"),
## +     optim(binit, .... [TRUNCATED]
##
## > res2 = summary(res)
```

The Rcpp code is 9.6637593 times faster than basic R.

Following the advice of both Dirk Eddelbuettel and Romain Francois, I have switched to RcppArmadillo to rely on 'the matrix code by professionals and decades of testing (LAPACK and BLAS)'.

```
Rcpp::sourceCpp('multieventarmadillo.cpp')
```

```
##
## > set.seed(1)
##
## > data = read.table("titis2.txt")
##
## > nh <- dim(data)[1]
##
## > k <- dim(data)[2]
##
## > km1 <- k - 1
##
## > eff <- rep(1, nh)
```

```
##
## > fc <- NULL
##
## > init.state <- NULL
##
## > for (i in 1:nh) {
## +     temp <- 1:k
## +     fc <- c(fc, min(which(data[i, ] != 0)))
## +     init.state <- c(init.state, data[i, fc[i]])
## + }
##
## > binit <- runif(9)
##
## > data <- t(data)
##
## > devMULTIEVENT <- function(b, data, eff, e, garb, nh,
## +     km1) {
## +     par = plogis(b)
## +     piNB <- par[1]
## +     phiNB <- par[2]
## +     phiB <- pa .... [TRUNCATED]
##
## > devMULTIEVENT(binit, data, eff, fc, init.state, nh,
## +     km1)
## [1] 4119.899
##
## > multieventrcpp(binit, data, fc, init.state)
##          [,1]
## [1,] 4119.899
##
## > deb = Sys.time()
##
## > tmpmin1 <- optim(binit, devMULTIEVENT, NULL, hessian = T,
## +     data, eff, fc, init.state, nh, km1, method = "BFGS")
##
## > fin = Sys.time()
##
## > res1 = fin - deb
##
## > deb = Sys.time()
##
## > tmpmin2 <- optim(binit, multieventrcpp, NULL, hessian = T,
## +     data, fc, init.state, method = "BFGS")
##
## > fin = Sys.time()
##
## > res2 = fin - deb
##
## > res1
## Time difference of 10.25048 secs
##
## > tmpmin1$par
## [1]  0.8674528  1.4764532  1.6396689 -1.2691570 -1.2283542  0.2600259
## [7]  0.3959696 -1.4643768  1.0333858
```

```
##
## > res2
## Time difference of 0.20031 secs
##
## > tmpmin2$par
## [1]  0.8674528  1.4764532  1.6396689 -1.2691570 -1.2283542  0.2600259
## [7]  0.3959696 -1.4643768  1.0333858
##
## > library(microbenchmark)
##
## > res = microbenchmark(optim(binit, devMULTIEVENT, NULL,
## +     hessian = T, data, eff, fc, init.state, nh, km1, method = "BFGS"),
## +     optim(binit, .... [TRUNCATED]
##
## > res2 = summary(res)
```

Now the RcppArmadillo code is 48.4372229 times faster than basic R!!