|  |  |  |
| --- | --- | --- |
| Table 3. Comparison Between C++ and R | | |
| R Code (gompertz\_event2) | C++ Code (gompertz\_eventC) | Explanation |
| gompertz\_event2 <- function(  m\_ind\_traits,  m\_coef\_ukpds\_ind\_traits,  m\_coef\_ukpds\_other\_ind\_traits,  health\_outcome) { | // [[Rcpp::export]] auto gompertz\_eventC(  arma::mat& m\_ind\_traits,  const arma::mat& m\_coef\_ukpds\_ind\_traits,  const arma::mat& m\_coef\_ukpds\_other\_ind\_traits,  int health\_outcome\_index) { | The function is exported to R using Rcpp attributes. The input arguments are typed explicitly as Armadillo matrices. `int health\_outcome\_index` replaces R’s use of column names and is adjusted to 0-based indexing inside the function. It is worth mentioning that we are when adding the ‘&’ it is passing by reference instead of actually passing the matrix back and forth. |
|  | int n\_rows = m\_ind\_traits.n\_rows; | Retrieves the number of individuals (rows in matrix). This is needed to generate a matrix of random uniform values using `arma::randu` later. |
|  | int idx = health\_outcome\_index - 1; | Adjusts for 0-based indexing used in C++ (R is 1-based). This index is used to access columns in coefficient matrices. |
| patient\_factors <- (m\_ind\_traits %\*% m\_coef\_ukpds\_ind\_traits[, health\_outcome] +  as.vector(m\_coef\_ukpds\_other\_ind\_traits["lambda", health\_outcome])) | arma::vec coef = m\_coef\_ukpds\_ind\_traits.col(idx); double lambda = m\_coef\_ukpds\_other\_ind\_traits(0, idx); arma::vec patient\_factors = m\_ind\_traits \* coef; patient\_factors += lambda; | Performs matrix multiplication and adds the intercept. C++ breaks this into sequential steps with explicit types. `col(idx)` extracts a single column as a vector. Matrix multiplication is performed using `\*`. Scalars must be explicitly extracted. `+=` adds lambda to each element of the result vector. |
| cum\_hazard\_t <- (1 / m\_coef\_ukpds\_other\_ind\_traits["rho", health\_outcome]) \*  exp(patient\_factors) \*  (exp(m\_ind\_traits[, "age"] \* m\_coef\_ukpds\_other\_ind\_traits["rho", health\_outcome]) - 1) | double rho = m\_coef\_ukpds\_other\_ind\_traits(1, idx); double inv\_rho = 1.0 / rho; const arma::vec& age = m\_ind\_traits.col(0); arma::vec patient\_factors\_exp = arma::exp(patient\_factors); arma::mat p\_t0 = arma::exp(age \* rho) - 1.0; arma::mat cum\_hazard\_t = inv\_rho \* (patient\_factors\_exp % p\_t0); | Cumulative hazard at time t using Gompertz function. `col(0)` assumes age is the first column of m\_ind\_traits. Armadillo's `exp()` is element-wise. `%` denotes element-wise multiplication. Intermediate results are stored in named variables for clarity and performance. |
| cum\_hazard\_t1 <- (1 / m\_coef\_ukpds\_other\_ind\_traits["rho", health\_outcome]) \*  exp(patient\_factors) \*  (exp((m\_ind\_traits[, "age"] + 1) \* m\_coef\_ukpds\_other\_ind\_traits["rho", health\_outcome]) - 1) | arma::vec age1 = age + 1; arma::mat p\_t1 = arma::exp(age1 \* rho) - 1.0; arma::mat cum\_hazard\_t1 = inv\_rho \* (patient\_factors\_exp % p\_t1); | Computes cumulative hazard at t+1. The logic is the same but is broken into parts. |
| trans\_prob <- 1 - exp(cum\_hazard\_t - cum\_hazard\_t1) | arma::mat trans\_prob = 1 - arma::exp(cum\_hazard\_t - cum\_hazard\_t1); | Both versions calculate transition probabilities from the difference in cumulative hazard. `arma::exp` is element-wise exponential. |
| event <- (trans\_prob > runif(nrow(m\_ind\_traits))) \* 1 | arma::mat random\_numbers = arma::randu(n\_rows, 1); arma::umat event = trans\_prob > random\_numbers; | Generates uniform random draws and determines whether the event occurred. `arma::randu(n\_rows, 1)` generates a matrix of random uniform values. Logical comparison returns a `umat` (unsigned int matrix) where 1 indicates event occurrence. Armadillo handles this natively. |
| colnames(event) <- health\_outcome |  | Column names are omitted because Armadillo matrices do not store metadata like names. If needed, this must be added in R after returning the object. |
| return(event) | return event; | Returns the logical matrix indicating event occurrence for each individual. |