Calling C++ from R via Rcpp

The package `Rcpp` makes it easy to call C++ from R, which can speed up function calls, loops that can't be vectorized, and other code; and it provides data structures and algorithms from C++ libraries that aren't available in R.

To prepare to use Rcpp,

- `install.packages("Rcpp"); require("Rcpp")`
- install a C++ compiler:
  - Windows: Rtools (http://cran.r-project.org/bin/windows/Rtools)
  - Mac: Xcode (https://developer.apple.com/xcode/downloads)

To include a short C++ function within a "`.R`" file, use `cppFunction(code)`, where `code` is a character string containing the C++ function. e.g.

```
fibonacci = function(n) { # A recursive function needing > fibonacci(n) calls!
  if (n == 0) return(0)
  if (n == 1) return(1)
  return(fibonacci(n-1) + fibonacci(n-2))
}
system.time(f <- fibonacci(30))
print(f)

cppFunction("
  int Fibonacci(int n) { // This is a C++ version.
    if (n == 0) return 0;
    if (n == 1) return 1;
    return(Fibonacci(n-1) + Fibonacci(n-2));
  }
")
system.time(F <- Fibonacci(30))
stopifnot(f == F)
```

To use longer C++ code from R,

- put the C++ code in a "`.cpp`" file that begins with

  `#include <Rcpp.h>`

  `using namespace Rcpp;`
- make a C++ function visible in R by preceding it with

  `// [[Rcpp::export]]`
- call `sourceCpp(file)`, where `file` is the name of the "`.cpp`" file

e.g. See `escapeTime.cpp` and `mandelbrotRcpp.R`

Translating basic R to basic C++

Here are a few R programming constructs and the corresponding C++:

- conditional: `if,  if ... else,  if ... else if ... else  # R and C++ (no change)`
- loop:
  - loop through a vector:
    ```
    for (VARIABLE in SEQUENCE)                 { EXPRESSION }  #  R
    for (INITIALIZATION; CONDITION; INCREMENT) { EXPRESSION }  // C++
    ```
    e.g.
    ```
    for (i in seq_len(n))         { ... }  #  R   (indices 1 to n  )
    for (i = 0; i < n; i = i + 1) { ... }  // C++ (indices 0 to n-1)
    ```
  - repeat zero or more times:
    ```
    while (CONDITION) { EXPRESSION }  # R and C++ (no change)
    ```
  - repeat one or more times:
    ```
    repeat {         # R
      EXPRESSION
      if (CONDITION) {
        break
      }
    }
    ```
    ```
    do {           // C++
      EXPRESSION
    } while (CONDITION);
    ```
- function:
  ```
  f = function(PARAMETER.LIST) {
    BODY                     # R
  }
  ```
  ```
  RETURN_TYPE f(TYPE PARAMETER LIST) {
    BODY                            // C++
  }
  ```

Here are a few R types and their corresponding C++ types:

```
scalars:
  # R                           // C++
  x = logical(1) # TRUE, FALSE    bool   x; // true, false
  x = integer(1)                int    x;
  x = numeric(1)                double x;
  x = character(1)              String x;
vectors:
  x = logical(n)                LogicalVector   x(n); // omit "(n)" in parameter
  x = integer(n)                IntegerVector   x(n);
  x = numeric(n)                NumericVector   x(n);
  x = character(n)              CharacterVector x(n);

  n = length(x)                 n = x.size();
```

For more, see `http://adv-r.had.co.nz/Rcpp.html`