Rcpp: A quick look

Stat 580: Statistical Computing

- Theme: Black - White
- Printable version

References

- "Advanced R", by Hadley Wickham.
- "Writing R Extensions", by R Core Team.
- "Rcpp: Seamless R and C++ Integration", by Dirk Eddelbuettel and Romain Francois, published in JSS 2011

Convolve example

```
#include <R.h>
#include <Rinternals.h>

/* from "Writing R Extensions" */

SEXP convolve2(SEXP a, SEXP b)
{
  int na, nb, nab;
  double *xa, *xb, *xab;
  SEXP ab;

  a = PROTECT(coerceVector(a, REALSXP));
  b = PROTECT(coerceVector(b, REALSXP));
  na = length(a); nb = length(b); nab = na + nb - 1;
  ab = PROTECT(allocVector(REALSXP, nab));
  xa = REAL(a); xb = REAL(b); xab = REAL(ab);
    for(int i = 0; i < nab; i++) xab[i] = 0.0;
    for(int i = 0; i < na; i++)
    for(int j = 0; j < nb; j++) xab[i + j] += xa[i] * xb[j];
    UNPROTECT(3);
    return ab;
}
```

Convolve example

```
#include <Rcpp.h>
/* from Rcpp: Seamless R and C++ Integration */
RcppExport SEXP convolve3cpp(SEXP a, SEXP b) {
  Rcpp::NumericVector xa(a);
  Rcpp::NumericVector xb(b);
  int n_xa = xa.size(), n_xb = xb.size();
  int nab = n_xa + n_xb - 1;
  Rcpp::NumericVector xab(nab);
  for (int i = 0; i < n_xa; i++)
    for (int j = 0; j < n_xb; j++)
      xab[i + j] += xa[i] * xb[j];
  return xab;
}
```

- Same tool: `R CMD SHLIB` and `.Call()` (load the package "Rcpp" before `dyn.load()`)
- Need to input some linker options:
  - via variables PKG_CXXFLAGS, PKG_LIBS

```
export PKG_CPPFLAGS='x' #replace x by output of Rscript -e "Rcpp:::CxxFlags()"
export PKG_LIBS='x' #replace x by output of Rscript -e "Rcpp:::LdFlags()"
```

Convolve example

- Only a single header file `Rcpp.h`
  - if you dig deep into `Rcpp.h`, it includes `Rcppcommon.h` (`->` `Rcpp/r/headers.h` `->` `R.h` and `Rinternals.h`)
- RcppExport is an alias to 'extern "C"' defined by Rcpp
  - C++ compiler mangles the name of the function and .Call can't find it
  - only useful when the function is intended to be called by `.Call()`
- the inputs are converted to C++ vector types provided by Rcpp
- `size()` is a member function of the class `NumericVector`

Convolve example

- the return conversion (`NumericVector` to `SEXP`) is automatic
  - you can use `Rcpp::wrap()` to implement the conversion manually
- in general, `Rcpp::as` can be used to convert R object to Rcpp object
- no protection needed

sourceCpp()

```
#include <Rcpp.h>

using namespace Rcpp;

// [[Rcpp::export]]

SEXP convolve3cpp(SEXP a, SEXP b) {
  NumericVector xa(a);
  NumericVector xb(b);
  int n_xa = xa.size(), n_xb = xb.size();
  int nab = n_xa + n_xb - 1;
  NumericVector xab(nab);
  for (int i = 0; i < n_xa; i++)
    for (int j = 0; j < n_xb; j++)
      xab[i + j] += xa[i] * xb[j];
  return xab;
}
```

sourceCpp()

```
#include <Rcpp.h>

using namespace Rcpp;

// [[Rcpp::export]]

NumericVector convolve3cpp(NumericVector xa, NumericVector xb) {
  int n_xa = xa.size(), n_xb = xb.size();
  int nab = n_xa + n_xb - 1;
  NumericVector xab(nab);
  for (int i = 0; i < n_xa; i++)
    for (int j = 0; j < n_xb; j++)
      xab[i + j] += xa[i] * xb[j];
  return xab;
}
```

Rcpp

- The `RObject` class is the basic class of Rcpp API
  - thin wrapper around a `SEXP` object (no copy)
  - manages the life cycle (protected from garbage collection while in scope)

```
SEXP ab;
PROTECT(ab = allocVector(REALSXP, 2));
REAL(ab)[0] = 123.45;
REAL(ab)[1] = 67.89;
UNPROTECT(1);
```

```
Rcpp::NumericVector ab(2);
ab[0] = 123.45;
ab[1] = 67.89;
```

Rcpp

- Useful examples of derived classes:

Rcpp class	*R typeof*
IntegerVector, IntegerMatrix	*integer*
NumericVector, NumericMatrix	*numeric*
LogicalVector, LogicalMatrix	*logical*
CharacterVector, CharacterMatrix	*character*
List	*list*
ExpresssionVector, ExpressionMatrix	*expression*
Environment	*environment*
Function	*function*
...	...

Constructor of NumericVector

```
// from SEXP
SEXP x;
NumericVector y(x);

// cloning (deep copy)
NumericVector z = clone<NumericVector>(y);

// given size (elements are initialized to 0.0)
NumericVector y(10);

// specifying a common value
NumericVector y(10, 2.0);

// with elements generated
NumericVector y(10, unif_rand);

// with given elements
NumericVector y = NumericVector::create(1.0, 2.0);
```

RcppArmadillo

- Armadillo is a state-of-art C++ linear algebra library.

```
#include <RcppArmadillo.h>
// [[Rcpp::depends(RcppArmadillo)]]

using namespace Rcpp;

// [[Rcpp::export]]
arma::mat mylm(arma::mat & X, arma::vec & y) {
  // for arma, with &: pass-by-reference
  arma::mat XtX = X.t() * X;
  return XtX.i() * X.t() * y;
}
```