STAT40780 Data Programming with C (online)

Lab Sheet 11+12 (Solutions)

Dr Marie Galligan

Summer 2015

This lab sheet poses questions relating to lecture material covered during this module, particularly that covered in weeks 11 and 12.

1 Find the median

With the help of the Rcpp and the `nth_element` algorithm from the Standard Template Library (STL), write a C++ function that's callable from R, to find the median of a vector passed from R.

A possible solution can be found at:
`http://gallery.rcpp.org/articles/robust-estimators/`

2 Count values

With the help of the Rcpp and the `count` algorithm from the Standard Template Library (STL), write a C++ function that's callable from R, to count the number of missing values in an R vector.

A possible solution:

Count missing values

```
1
2  body_countMiss <- '
3   NumericVector xx(x);
4   LogicalVector y = is_na(xx);
5   int result = std::count(y.begin(),y.end(), true);
6   return wrap(result);
7  '
8
9  countMiss <- cxxfunction( signature(x = "numeric"),
10                           body = body_countMiss,
11                           plugin= "Rcpp")
12
13 x <- rnorm(100)
14 #set 10 values from x to zero
15 x[sample(1:100, size = 10, replace = FALSE)] <- NA
16 countMiss(x)
```

3 Merge and sort

Write a C++ function that is callable from R, that takes as input two numeric vectors from R, and merges these into a single sorted vector, which is returned to R. The function should not sort the original vectors passed from R.

Merge and sort two vectors

```
1  body_mergeTwo <- '
2   NumericVector xx = clone(x);
3   NumericVector yy = clone(y);
4   std::sort(xx.begin(), xx.end());
5   std::sort(yy.begin(), yy.end());
6   NumericVector out = NumericVector( xx.size() + yy.size());
7   std::merge(xx.begin(), xx.end(), yy.begin(), yy.end(), out.begin());
8   return wrap(out);
9  '
10
11 mergeTwo <- cxxfunction( signature(
12                           x = "numeric", y = "numeric"),
13                           body = body_mergeTwo,
14                           plugin= "Rcpp")
15
16
17 x <- rnorm(10)
18 y <- rnorm(10)
19 x
20 y
21 mergeTwo(x,y)
```

4 Simulate values from a multivariate normal density

Write a C++ function (callable from R) with the help of RcppArmadillo, to simulate n observations from a multivariate normal distribution, given a specified mean vector and covariance matrix (passed from R), and returns the simulated data in a matrix.

The following algorithm might help. It may be used to generate a single observation y (a vector) from a multivariate normal distribution with mean vector μ and covariance matrix Σ:

- First, use a Cholesky decomposition to factor the covariance matrix Σ into a left triangular matrix times its transpose

$$\mathbf{\Sigma = LL^T}$$

- Set p = number of columns (or equivalently number of rows) in Σ...the new observation will be of length p
- Fill a vector $\mathbf{x}$ with p values generated randomly from a standard normal distribution: N(0 , 1)
- A single observation $\mathbf{y}$ from a p-dimensional multivariate normal distribution with mean vector μ and covariance matrix Σ may obtained as:

$$\mathbf{y = Lx} + \mu$$

A proof of why this algorithm works may be found at:

To test your function, generate a matrix of 100 observations from a multivariate normal distribution with $\mu = (10, 5, -3)$ and

$$\Sigma = \begin{bmatrix} 1.0 & 0.9 & -0.3 \\ 0.9 & 1.0 & -0.4 \\ -0.3 & -0.4 & 1.0 \end{bmatrix}$$

A possible solution may be found at:

`http://gallery.rcpp.org/articles/simulate-multivariate-normal/`

5 Centering a matrix

Write a C++ function that takes as input a numeric matrix from R, and centers the data in the matrix, by calculating a vector of means (containing a mean for each column of the matrix), and subtracting the mean vector from each row of the matrix. The centered matrix should be returned to R. Compile using `cxxfunction()` and call the resulting R wrapper function `centerRcpp()`.

Run the following R code, which loads the `iris` dataset into R and extracts the first four columns (containing sepal length, sepal width, petal length and

petal width), converts to a matrix and names the matrix Y. Pass the matrix Y to `centerRcpp()` to obtained a centered dataset.

Note that the 5th column of `iris` contains the categorical variable `Species` and hence is removed before passing to C++.

```
data(iris)  #load iris data into R
names(iris) #extract names of variables in the data.frame
?iris  #get information on the iris dataset

#remove last column of the iris data.frame,
#which contains the categorical variable Species
Y <-  as.matrix( iris[,-5] )
```

A possible solution:

Centering a matrix

```
centerRcpp <- cxxfunction( signature(X="numeric"),
                    body = '
                      mat M = as<mat>(X);
                      //compute mean of each col
                      rowvec mu = mean(M, 0);
                      //subtract mean vector from each row
                      M.each_row() -= mu;
                      return wrap(M);
                     ',
                    include = 'using namespace arma;',
                    plugin = "RcppArmadillo")

data(iris)  #load iris data into R
names(iris) #extract names of variables in the data.frame
?iris  #get information on the iris dataset

#remove last column of the iris data.frame,
#which contains the categorical variable Species
Y <-  as.matrix( iris[,-5] )
#call function and save centered data as centY
centY <- centerRcpp( Y )
#view first few rows
head(centY)
#mean of each column should be 0 if centering succeeded
round( apply(centY,2,mean) , 10)
```

6 Outer product

Write a C++ functions (callable from R) that finds the outer product (i.e. cross-product) of two vectors $\mathbf{x}$ and $\mathbf{y}$.

Note that the outer product is equivalent to a matrix multiplication $\mathbf{xy^T}$ where both $\mathbf{x}$ and $\mathbf{y}$ are column vectors.

A possible solution:

Outer product of two vectors

```
library(Rcpp)
library(inline)

#Outer product of two vectors
op <- cxxfunction( signature(x="numeric", y = "numeric"),
                    body = '
```

```
                      vec v1 = as<vec>(x); //convert to C++ vec
                      vec v2 = as<vec>(y); //convert to C++ vec
                      mat M = v1 * v2.t(); //outer product
                      return wrap(M);
                     ',
                     include = 'using namespace arma;',
                     plugin = "RcppArmadillo")

#generate some data
x <- rnorm(5)
y <- rnorm(3)

#call the outer product function
op(x,y)
```