Chapter 1

Basic Vector Operations in C++ and R

1.1 C++ code

Start with basic examples to gain an understanding of how to use vectors in C++. This example will demonstrate how to initialize a vector, assign values, add elements, remove elements, and print the vector.

```
#include <iostream>
#include <vector>

int main(){
        // initialize a vector of 5 elements initialized to 0
        std::vector<int> vec(5);

        // get the size (number of elements) of vec
        int vec_size = vec.size();

        std::cout << "number of elements in vec: " <<
                vec_size << std::endl;

        // print the vector elements to cout
        for(int i = 0; i < vec.size(); i++){
                std::cout << vec[i] << " ";
        }
        std::cout << std::endl; // this just adds a new line
```

```
    // assign values to the elements using the [] operator
    // the index starts at 0 in C++
    vec[0] = 6;
    vec[1] = 2;
    vec[2] = 5;
    vec[3] = 1;
    vec[4] = 8;

    // print the vector elements to cout
    for(int i = 0; i < vec.size(); i++){
            std::cout << vec[i] << " ";
    }
    std::cout << std::endl;

    // add elements to the end of the vector
    vec.push_back(5);
    vec.push_back(7);
    vec.push_back(3);

    // print the vector elements to cout
    for(int i = 0; i < vec.size(); i++){
            std::cout << vec[i] << " ";
    }
    std::cout << std::endl;

    // delete the last element
    vec.pop_back();

    // print the vector elements to cout
    for(int i = 0; i < vec.size(); i++){
            std::cout << vec[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

1.2 R code

Equivalent code in R

```
# initialize a vector of 5 elements initialized to 0
vec <- vector(mode="numeric", length=5)

# get the size of vec
vec_size <- length(vec)

cat("number of elements in vec:", vec_size, "\n")

# print the vector elements
cat(vec, "\n")

# assign values to the elements
# the index starts at 1 in R
vec[1] <- 6
vec[2] <- 2
vec[3] <- 5
vec[4] <- 1
vec[5] <- 8

# print the vector elements
cat(vec, "\n")

# add elements to the end of the vector
vec <- c(vec, 5)
vec <- c(vec, 7)
vec <- c(vec, 3)

cat(vec, "\n")

# delete the last element
vec <- head(vec, -1)

# print the vector elements
cat(vec, "\n")
```

1.3 Output

Output of ex1.cpp

```
$ make ex1
$ ./ex1
number of elements in vec: 5
0 0 0 0 0
6 2 5 1 8
6 2 5 1 8 5 7 3
6 2 5 1 8 5 7
```

Output of ex1.R

```
$ Rscript ex1.R
number of elements in vec: 5
0 0 0 0 0
6 2 5 1 8
6 2 5 1 8 5 7 3
6 2 5 1 8 5 7
```

Chapter 2

Computing the sum and mean of vectors in C++ and R

2.1 C++ code

The C++ example icontains 3 different functions to compute the sum of a vector. One thing to notice is the mean function calls the sum2 function.

```
// example to compute the sum and mean of a vector
// the example will consider a vector of ints

#include <iostream>
#include <vector>
#include <numeric>

// function to compute the sum of an int vector
int sum1(std::vector<int> v){
        int acc = 0;
        for(int i = 0; i < v.size(); i++){
                acc += v[i];
        }
        return acc;
}

// function to compute the sum of an int vector
// this function makes use of accumulate from
```

```
// the numeric header
int sum2(std::vector<int> v){
        return std::accumulate(v.begin(), v.end(), 0);
}

double mean(std::vector<int> v){
        double result = sum2(v) / (double)v.size();
        return result;
}

int main(){

        // initialize a vector
        std::vector<int> vec(10);

        // loop to assign values 1:10 to the vector
        for(int i = 0; i < vec.size(); i++){
                // vec[i] = i + 1;
                vec.at(i) = i + 1;
        }

        // print the vector elements to cout
        for(int i = 0; i < vec.size(); i++){
                std::cout << vec[i] << " ";
        }
        std::cout << std::endl;

        // compute the mean and sum of the vector
        int vec_sum1 = sum1(vec);
        int vec_sum2 = sum2(vec);
        double vec_mean = mean(vec);

        std::cout << "sum1 function using loop" << std::endl;
        std::cout << "The sum of the vector is " << vec_sum1 << std::e

        std::cout << "sum2 function using accumulate" << std::endl;
        std::cout << "The sum of the vector is " << vec_sum2 << std::e

        std::cout << "the mean of the vector is " << vec_mean << std::
```

```
        return 0;
}
```

2.2 R code

Equivalent code in R. Notice how R already has built in functions that are used to compute the sum and mean of a vector.

```
# initialize a vector and assign values 1:10 to the vector
vec <- c(1:10)

# print the elements of vector
cat(vec, "\n")

# compute the sum and mean of the vector
sum1 <- sum(vec)
mean1 <- mean(vec)

cat("The sum of the vector is: ", sum1, "\n")
cat("The mean of the vector is: ", mean1, "\n")
```

2.3 Rcpp code

Use the Rcpp and inline packages to write functions to compute the sum and mean of vectors.

```
library(Rcpp)
library(inline)

Sys.setenv(R_ARCH="/x86_64")

# sum1
# sum the vector using a loop
sum1 <- cxxfunction(signature(x = "numeric"),
                    'NumericVector xx(x);
                    double sum = 0;
                    for(int i = 0; i < xx.size(); i++){
                      sum += xx[i];
                    }
```

```
                    return wrap(sum);',
                    plugin = "Rcpp")

# sum2
# sum the vector using std::accumulate
sum2 <- cxxfunction(signature(x = "numeric"),
                    'NumericVector xx(x);
                    double res = std::accumulate(xx.begin(), xx.end(),
                    return wrap(res);',
                    plugin = "Rcpp")

# sum3
# sum the vector using Rcpp sugar
sum3 <- cxxfunction(signature(x = "numeric"),
                    'NumericVector xx(x);
                    double res = sum(xx);
                    return wrap(res);',
                    plugin = "Rcpp")

# mean1
# compute the mean of a vector using Rcpp
mean1 <- cxxfunction(signature(x = "numeric"),
                    'NumericVector xx(x);
                    double sum = std::accumulate(xx.begin(), xx.end(),
                    double res = sum / xx.size();
                    return wrap(res);',
                    plugin = "Rcpp")

# check to make sure mean1 is correct
mean1(1:100) == mean(1:100)

# test the sum functions with Rcpp and the base sum function
x <- 1:10000
# base R sum function
sum(x)

#check to make sure sum1, sum2, and sum3 are correct
sum1(x) == sum(x)
sum2(x) == sum(x)
sum3(x) == sum(x)
```

```
# test the speed of the two versions
library(rbenchmark)
res <- benchmark(sum(x), sum1(x), sum2(x), sum3(x),
                 columns = c("test", "replications", "elapsed",
                             "relative", "user.self", "sys.self"),
                 order = "relative", replications = 100)
print(res)
```

2.4 Output

Output of ex2.cpp

```
$ make ex2
$ ./ex2
1 2 3 4 5 6 7 8 9 10
sum1 function using loop
The sum of the vector is 55
sum2 function using accumulate
The sum of the vector is 55
the mean of the vector is 5.5
```

Output of ex2.R

```
$ Rscript ex2.R
1 2 3 4 5 6 7 8 9 10
The sum of the vector is:  55
The mean of the vector is:  5.5
```

Chapter 3

Add two vectors in C++ and R

3.1 C++ code

The C++ example in which I write a function to add two vectors together uses a traditional loop to do element by element addition

```
// example perform operations on vectors
// add two vectors

#include <iostream>
#include <vector>

// function to add two vectors of ints and
// return a vector that stores the result
std::vector<int> addVec(const std::vector<int>& v1,
                                                const std::vector<int>& v2){
        std::vector<int> result(v1.size());
        for(int i = 0; i < v1.size(); i++){
                result[i] = v1[i] + v2[i];
        }
        return result;
}

int main(){
        // initialize vectors
        std::vector<int> vec1(10);
```

```
        std::vector<int> vec2(10);

        // loop to assign elements 1:10 to vec1
        // and assign elements 20:30 to vec2
        for(int i = 0; i < vec1.size(); i++){
                vec1.at(i) = i + 1;
                vec2.at(i) = i + 20;
        }

        // initialize a vector to store the results
        // of adding vec1 and vec2
        std::vector<int> res;

        res = addVec(vec1, vec2);

        // print the vector elements to cout
        for(int i = 0; i < res.size(); i++){
                std::cout << res.at(i) <<" ";
        }
        std::cout << std::endl;

        return 0;
}
```

3.2 R code

Equivalent code in R. Notice how there is no need to define a function to add two vectors. Simply use the + operator.

```
# add 2 vectors

# initialize a vector with values 1:10
vec1 <- c(1:10)

# initialize a vector with values 20:29
vec2 <- c(20:29)

# add the two vectors together
res <- vec1 + vec2
```

```
# print the vector elements
cat(res)
```

3.3 Rcpp code

TODO write ex3_rcpp.R to add two vectors together

3.4 Output

Output of ex3.cpp

```
$ make ex3
$ ./ex3
21 23 25 27 29 31 33 35 37 39
```

Output of ex3.R

```
$ Rscript ex3.R
21 23 25 27 29 31 33 35 37 39
```