

Chapter 1

Basic Arithmetic Operations on Vectors in C++ and R

The first thing to notice is that with functions in C++, the return type and argument type have to be declared in the function. Also note functions can have the same name with different arguments. In R, if x and y are vectors we can add the vectors with $x + y$. We can't do this in C++, at least not without overloading the $+$ operator. Rcpp sugar makes this possible with a little bit of what seems like "magic". Rcpp sugar will be covered later.

1.1 Addition

Simple program to add two vectors and add a scalar to a vector

code/add_vec.cpp

```
1  #include <vector>
2  #include <functional>
3  #include <iostream>
4
5  std::vector<double> add_vec(const std::vector<double>& v1,
6                             const std::vector<double>& v2){
7      std::vector<double> v(v1.size());
8      std::transform(v1.begin(), v1.end(), v2.begin(), v.begin(),
9                     std::plus<double>());
10     return v;
11 }
12
13 std::vector<double> add_vec(const std::vector<double>& v1,
14                             const double& n){
15     std::vector<double> v(v1.size());
16     std::transform(v1.begin(), v1.end(), v.begin(),
17                   std::bind2nd(std::plus<double>(), n));
18     return v;
19 }
20
21 int main(){
22     std::vector<double> vec1(10);
23     std::vector<double> vec2(10);
24
25     // fill vec1 with 1 to 10
```

```

26 // fill vec2 with 10 to 19
27 for(int i = 0; i < vec1.size(); i++){
28     vec1[i] = i + 1;
29     vec2[i] = i + 10;
30 }
31
32 std::vector<double> vec3(10);
33 vec3 = add_vec(vec1, vec2);
34
35 std::vector<double> vec4(10);
36 vec4 = add_vec(vec1, 5.5);
37
38
39 std::cout << "Result of add_vec(vec1, vec2)" << std::endl;
40 for(int i = 0; i < vec3.size(); i++){
41     std::cout << vec3[i] << " ";
42 }
43 std::cout << std::endl;
44
45 std::cout << "Result of add_vec(vec1, 5.5)" << std::endl;
46 for(int i = 0; i < vec4.size(); i++){
47     std::cout << vec4[i] << " ";
48 }
49 std::cout << std::endl;
50
51 return 0;
52 }

```

1.2 Subtraction

Simple program to subtract two vectors and subtract a vector by a scalar

code/sub-vec.cpp

```

1 #include <vector>
2 #include <functional>
3 #include <iostream>
4
5 std::vector<double> sub_vec(const std::vector<double>& v1,
6     const std::vector<double>& v2){
7     std::vector<double> v(v1.size());
8     std::transform(v1.begin(), v1.end(), v2.begin(), v.begin(),
9         std::minus<double>());
10     return v;
11 }
12
13 std::vector<double> sub_vec(const std::vector<double>& v1,
14     const double& n){
15     std::vector<double> v(v1.size());
16     std::transform(v1.begin(), v1.end(), v.begin(),
17         std::bind2nd(std::minus<double>(), n));
18     return v;
19 }
20

```

```

21 int main(){
22     std::vector<double> vec1(10);
23     std::vector<double> vec2(10);
24
25     // fill vec1 with 1 to 10
26     // fill vec2 with 10 to 19
27     for(int i = 0; i < vec1.size(); i++){
28         vec1[i] = i + 1;
29         vec2[i] = i + 10;
30     }
31
32     std::vector<double> vec3(10);
33     vec3 = sub_vec(vec1, vec2);
34
35     std::vector<double> vec4(10);
36     vec4 = sub_vec(vec1, 5.5);
37
38     std::cout << "Result of sub_vec(vec1, vec2)" << std::endl;
39     for(int i = 0; i < vec3.size(); i++){
40         std::cout << vec3[i] << " ";
41     }
42     std::cout << std::endl;
43
44     std::cout << "Result of sub_vec(vec1, 5.5)" << std::endl;
45     for(int i = 0; i < vec4.size(); i++){
46         std::cout << vec4[i] << " ";
47     }
48     std::cout << std::endl;
49
50     return 0;
51 }

```

1.3 Multiplication

Simple program to multiply two vectors and multiply a vector by a scalar

code/mult_vec.cpp

```

1  #include <vector>
2  #include <functional>
3  #include <iostream>
4
5  std::vector<double> mult_vec(const std::vector<double>& v1,
6      const std::vector<double>& v2){
7      std::vector<double> v(v1.size());
8      std::transform(v1.begin(), v1.end(), v2.begin(), v.begin(),
9          std::multiplies<double>());
10     return v;
11 }
12
13 std::vector<double> mult_vec(const std::vector<double>& v1,
14     const double& n){
15     std::vector<double> v(v1.size());
16     std::transform(v1.begin(), v1.end(), v.begin(),

```

```

17         std::bind1st(std::multiplies<double>(), n));
18     return v;
19 }
20
21 int main(){
22     std::vector<double> vec1(10);
23     std::vector<double> vec2(10);
24
25     // fill vec1 with 1 to 10
26     // fill vec2 with 10 to 19
27     for(int i = 0; i < vec1.size(); i++){
28         vec1[i] = i + 1;
29         vec2[i] = i + 10;
30     }
31
32     std::vector<double> vec3(10);
33     vec3 = mult_vec(vec1, vec2);
34
35     std::vector<double> vec4(10);
36     vec4 = mult_vec(vec1, 5.5);
37
38     std::cout << "Result of mult_vec(vec1, vec2)" << std::endl;
39     for(int i = 0; i < vec3.size(); i++){
40         std::cout << vec3[i] << " ";
41     }
42     std::cout << std::endl;
43
44     std::cout << "Result of mult_vec(vec1, 5.5)" << std::endl;
45     for(int i = 0; i < vec4.size(); i++){
46         std::cout << vec4[i] << " ";
47     }
48     std::cout << std::endl;
49
50     return 0;
51 }

```

1.4 Division

Simple program to divide two vectors and divide a vector by a scalar

code/div_vec.cpp

```

1  #include <vector>
2  #include <functional>
3  #include <iostream>
4
5  std::vector<double> div_vec(const std::vector<double>& v1,
6      const std::vector<double>& v2){
7      std::vector<double> v(v1.size());
8      std::transform(v1.begin(), v1.end(), v2.begin(), v.begin(),
9          std::divides<double>());
10     return v;
11 }
12

```

```
13 std::vector<double> div_vec(const std::vector<double>& v1,
14     const double& n){
15     std::vector<double> v(v1.size());
16     std::transform(v1.begin(), v1.end(), v.begin(),
17         std::bind2nd(std::divides<double>(), n));
18     return v;
19 }
20
21 int main(){
22     std::vector<double> vec1(10);
23     std::vector<double> vec2(10);
24
25     // fill vec1 with 1 to 10
26     // fill vec2 with 10 to 19
27     for(int i = 0; i < vec1.size(); i++){
28         vec1[i] = i + 1;
29         vec2[i] = i + 10;
30     }
31
32     std::vector<double> vec3(10);
33     vec3 = div_vec(vec1, vec2);
34
35     std::vector<double> vec4(10);
36     vec4 = div_vec(vec1, 5.5);
37
38     std::cout << "Result of div_vec(vec1, vec2)" << std::endl;
39     for(int i = 0; i < vec3.size(); i++){
40         std::cout << vec3[i] << " ";
41     }
42     std::cout << std::endl;
43
44     std::cout << "Result of div_vec(vec1, 5.5)" << std::endl;
45     for(int i = 0; i < vec4.size(); i++){
46         std::cout << vec4[i] << " ";
47     }
48     std::cout << std::endl;
49
50     return 0;
51 }
```


Chapter 2

Computing the Sum and Mean of Vectors in C++ and R

2.1 Sum

Compute the sum of a vector. The `sum1` function uses a for loop and the `sum2` function uses `std::accumulate` from the STL.

code/vec_sum.cpp

```
1 #include <vector>
2 #include <numeric>
3 #include <iostream>
4
5 double sum1(const std::vector<double>& v1){
6     double acc = 0;
7     for(int i = 0; i < v1.size(); i++){
8         acc += v1[i];
9     }
10    return acc;
11 }
12
13 double sum2(const std::vector<double>& v1){
14     return std::accumulate(v1.begin(), v1.end(), 0.0);
15 }
16
17 int main(){
18     std::vector<double> vec1(10);
19
20     // fill vec1 with 1 to 10
21     for(int i = 0; i < vec1.size(); i++){
22         vec1[i] = i + 1;
23     }
24
25     double num1, num2;
26
27     num1 = sum1(vec1);
28     num2 = sum2(vec1);
29
30     std::cout << "sum1(vec1) = " << num1 << std::endl;
31     std::cout << "sum2(vec2) = " << num2 << std::endl;
```

```

32
33     return 0;
34 }

```

2.2 Cumulative Sum

Compute the cumulative sum of a vector. The `cumsum1` function uses a for loop whereas the `cumsum2` function uses `std::partial_sum` from the STL.

code/vec_cumsum.cpp

```

1  #include <vector>
2  #include <numeric>
3  #include <iostream>
4
5  std::vector<double> cumsum1(const std::vector<double>& v1){
6      std::vector<double> v(v1.size());
7      v[0] = v1[0];
8      double acc = 0;
9      for(int i = 0; i < v.size(); i++){
10         acc += v1[i];
11         v[i] = acc;
12     }
13     return v;
14 }
15
16 std::vector<double> cumsum2(const std::vector<double>& v1){
17     std::vector<double> v(v1.size());
18     std::partial_sum(v1.begin(), v1.end(), v.begin());
19     return v;
20 }
21
22 int main(){
23     std::vector<double> vec1(10);
24
25     // fill vec1 with 1 to 10
26     for(int i = 0; i < vec1.size(); i++){
27         vec1[i] = i + 1;
28     }
29
30     std::vector<double> vec3(10);
31     vec3 = cumsum1(vec1);
32
33     std::vector<double> vec4(10);
34     vec4 = cumsum2(vec1);
35
36     std::cout << "Result of cumsum1(vec1)" << std::endl;
37     for(int i = 0; i < vec3.size(); i++){
38         std::cout << vec3[i] << " ";
39     }
40     std::cout << std::endl;
41
42     std::cout << "Result of cumsum2(vec1)" << std::endl;
43     for(int i = 0; i < vec4.size(); i++){

```



```

44         std::cout << vec4[i] << " ";
45     }
46     std::cout << std::endl;
47
48     return 0;
49 }

```

2.3 Mean

Compute the mean of a vector. Note how I use the same `sum2` function that was defined earlier by copy/-pasting it into the file. In general, this is a bad thing to do, but I am doing this for simplicity and so each file can be run as a stand-alone program. The correct way would be to define the `sum2` function in it's own .cpp file, then declare it in a header file, then include the header file.

code/vec_mean.cpp

```

1  #include <vector>
2  #include <numeric>
3  #include <iostream>
4
5
6  double sum2(const std::vector<double>& v1){
7      return std::accumulate(v1.begin(), v1.end(), 0.0);
8  }
9
10 double mean(const std::vector<double>& v1){
11     return sum2(v1) / (double)v1.size();
12 }
13
14 int main(){
15     std::vector<double> vec1(10);
16
17     // fill vec1 with 1 to 10
18     for(int i = 0; i < vec1.size(); i++){
19         vec1[i] = i + 1;
20     }
21
22     double num1;
23
24     num1 = mean(vec1);
25
26     std::cout << "mean(vec1) = " << num1 << std::endl;
27
28     return 0;
29 }

```


Chapter 3

Applying Functions to Vectors in C++ and R

3.1 Square Root

Compute the square root of each element in the vector

code/vec_sqrt.cpp

```
1  #include <vector>
2  #include <algorithm>
3  #include <math.h>
4  #include <iostream>
5
6  std::vector<double> vec_sqrt(const std::vector<double>& v1){
7      std::vector<double> v(v1.size());
8      std::transform(v1.begin(), v1.end(), v.begin(), sqrt);
9      return v;
10 }
11
12
13 int main(){
14     std::vector<double> vec1(10);
15
16     // fill vec1 with 1 to 10
17     for(int i = 0; i < vec1.size(); i++){
18         vec1[i] = i + 1;
19     }
20
21     std::vector<double> vec2(vec1.size());
22     vec2 = vec_sqrt(vec1);
23
24     std::cout << "Result of vec_sqrt(vec1)" << std::endl;
25     for(int i = 0; i < vec2.size(); i++){
26         std::cout << vec2[i] << " ";
27     }
28     std::cout << std::endl;
29
30     return 0;
31 }
```

3.2 Square

The `vec_squared` function computes the square each element in the vector using a for loop. The `vec_squared1` function computes the square each element in the vector using `std::multiplies` by multiplying the vector by itself.

code/vec_squared.cpp

```

1  #include <vector>
2  #include <algorithm>
3  #include <iostream>
4
5  std::vector<double> vec_squared(const std::vector<double>& v1){
6      std::vector<double> v(v1.size());
7      for(int i = 0; i < v1.size(); i++){
8          v[i] = v1[i] * v1[i];
9      }
10     return v;
11 }
12
13 std::vector<double> vec_squared1(const std::vector<double>& v1){
14     std::vector<double> v(v1.size());
15     std::transform(v1.begin(), v1.end(), v1.begin(), v.begin(),
16                   std::multiplies<double>());
17     return v;
18 }
19
20 int main(){
21     std::vector<double> vec1(10);
22
23     // fill vec1 with 1 to 10
24     for(int i = 0; i < vec1.size(); i++){
25         vec1[i] = i + 1;
26     }
27
28     std::vector<double> vec3(10);
29     vec3 = vec_squared(vec1);
30
31     std::vector<double> vec4(10);
32     vec4 = vec_squared1(vec1);
33
34     std::cout << "Result of vec_squared(vec1)" << std::endl;
35     for(int i = 0; i < vec3.size(); i++){
36         std::cout << vec3[i] << " ";
37     }
38     std::cout << std::endl;
39
40     std::cout << "Result of vec_squared1(vec1)" << std::endl;
41     for(int i = 0; i < vec4.size(); i++){
42         std::cout << vec4[i] << " ";
43     }
44     std::cout << std::endl;
45
46     return 0;
47 }
```

3.3 Sum of Squares

Compute the sum of squares using `std::inner_product`

code/ssq.cpp

```
1 #include <vector>
2 #include <numeric>
3 #include <iostream>
4
5 double ssq(const std::vector<double>& v1){
6     return std::inner_product(v1.begin(), v1.end(), v1.begin(), 0.0);
7 }
8
9 int main(){
10     std::vector<double> vec1(10);
11
12     // fill vec1 with 1 to 10
13     for(int i = 0; i < vec1.size(); i++){
14         vec1[i] = i + 1;
15     }
16
17     double num1;
18     num1 = ssq(vec1);
19
20     std::cout << "Result of ssq(vec1): " << num1 << std::endl;
21
22     return 0;
23 }
```

3.4 Power

Compute the "n" power of each element in the vector.

code/vec_pow.cpp

```
1 #include <vector>
2 #include <functional>
3 #include <cmath>
4 #include <iostream>
5
6
7 std::vector<double> vec_pow(const std::vector<double>& v1, const double& n){
8     std::vector<double> v(v1.size());
9     for(int i = 0; i < v.size(); i++){
10         v[i] = pow(v1[i], n);
11     };
12     return v;
13 }
14
15 int main(){
16     std::vector<double> vec1(10);
17
18     // fill vec1 with 1 to 10
19     for(int i = 0; i < vec1.size(); i++){
```

```

20         vec1[i] = i + 1;
21     }
22
23     std::vector<double> vec3(10);
24     vec3 = vec_pow(vec1, 3.5);
25
26
27     std::cout << "Result of vec_pow(vec1, 3.5)" << std::endl;
28     for(int i = 0; i < vec3.size(); i++){
29         std::cout << vec3[i] << " ";
30     }
31     std::cout << std::endl;
32
33     return 0;
34 }

```

3.5 User Defined Function

Apply the user defined function to each element in the vector. Here I define a simple function to add 3 to a number.

code/vec_myfun.cpp

```

1  #include <vector>
2  #include <algorithm>
3  #include <math.h>
4  #include <iostream>
5
6  double add3(const double& x){
7      return x + 3;
8  }
9  std::vector<double> vec_myfun(const std::vector<double>& v1){
10     std::vector<double> v(v1.size());
11     std::transform(v1.begin(), v1.end(), v.begin(), add3);
12     return v;
13 }
14
15
16 int main(){
17     std::vector<double> vec1(10);
18
19     // fill vec1 with 1 to 10
20     for(int i = 0; i < vec1.size(); i++){
21         vec1[i] = i + 1;
22     }
23
24     std::vector<double> vec2(vec1.size());
25     vec2 = vec_myfun(vec1);
26
27     std::cout << "Result of vec_myfun(vec1)" << std::endl;
28     for(int i = 0; i < vec2.size(); i++){
29         std::cout << vec2[i] << " ";
30     }
31     std::cout << std::endl;

```

```
32 |  
33 |     return 0;  
34 | }
```


Chapter 4

Vector Slicing and Dicing in C++ and R

Maybe come up with a better, more technical, term for the title

4.1 Vector Sorting

Sort a vector in ascending order
sort a vector in descending order

4.2 Minimums and Maximums

Find the minimum value of the vector
Find the maximum value of the vector
TODO: runMin and runMax

4.3 Counting

Count the elements of the vector
Count the elements of a vector meeting a specified criteria

4.4 Slicing

Given a vector, return a new vector of elements meeting a specified criteria
Sum the elements of a vector meeting a specified criteria