OPTIMIZING R CODE WITH RCPP

Rcpp classes and vectors

Romain François
Consulting Datactive, ThinkR

Previously on this course

- Create C++ functions ✅
- Write loops ✅

Vector classes

Rcpp vector classes:

- `NumericVector` to manipulate `numeric` vectors, e.g. `c(1,2,3)`
- `IntegerVector` for `integer` e.g. `1:3`
- `LogicalVector` for `logical` e.g. `c(TRUE, FALSE)`
- `CharacterVector` for strings e.g. `c("a", "b", "c")`

Also:

- `List` for lists, aka vectors of arbitrary R objects

Vector classes api

Most important methods:

- `x.size()` gives the number of elements of the vector `x`
- `x[i]` gives the element on the `i`th position in the vector `x`

C++ indexing

Indexing in C++ starts at 0. The index is an offset to the first position.

```
// first element of the vector
x[0]

// last element
x[x.size()-1]
```

Indexing in R starts at 1.

```
# first
x[1]

# last
x[length(x)]
```

Indexing in C++ starts at 0

The first element of a vector

```
// x comes from somewhere else (patience ...)
NumericVector x = ... ;

double value = x[0] ;

x[0] = 12.0 ;
```

The last element of a vector

```
// x comes from somewhere else (patience ...)
NumericVector x = ... ;
int n = x.size() ;

double value = x[n-1] ;

x[n-1] = 12.0 ;
```

Looping around a vector

```
// x comes from somewhere
NumericVector x = ... ;
int n = x.size() ;

for( int i=0; i<n; i++){
    // manipulate x[i]
}
```

OPTIMIZING R CODE WITH RCPP

Let's practice!

OPTIMIZING R CODE WITH RCPP

Creating vectors

Romain François
Consulting Datactive, ThinkR

Get a vector from the R side

C++ code

```
// [[Rcpp::export]]
double extract( NumericVector x, int i){
    return x[i] ;
}
```

Called from R

```
x <- c(13.2, 34.1)

extract(x, 0)
13.2

x[1]
13.2
```

NumericVectors

```
// [[Rcpp::export]]
double extract( NumericVector x, int i){
    return x[i] ;
}
```

Several cases

```
# x is already a numeric vector
extract( c(13.3, 54.2), 0 )
13.3
```

```
# x is an integer vector, it is first coerced to a numeric vector
extract( 1:10, 0 )
1
```

```
# conversion not possible: error
extract( letters, 0 )
Error in extract(letters, 0) :
  Not compatible with requested type: [type=character; target=double].
```

Create a vector of a given size

```
// [[Rcpp::export]]
NumericVector ones(int n){

    // create a new numeric vector of size n
    NumericVector x(n) ;

    // manipulate it
    for( int i=0; i<n; i++){
        x[i] = 1 ;
    }
    return x ;
}
```

Calling `ones` from R:

```
ones(10)

1 1 1 1 1 1 1 1 1 1
```

Constructor Variants

```
double value = 42.0 ;
int n = 20 ;

// create a numeric vector of size 20
// with all values set to 42
NumericVector x( n, value ) ;
```

Given set of values

```
NumericVector x = NumericVector::create( 1, 2, 3 ) ;

CharacterVector s = CharacterVector::create( "pink", "blue" ) ;
```

Given set of values with names

Naming all values

```
NumericVector x = NumericVector::create(
    _["a"] = 1, _["b"] = 2, _["c"] = 3
) ;
```

Only naming some values

```
IntegerVector y = IntegerVector::create(
    _["d"] = 4, 5, 6, _["f"] = 7
) ;
```

Vector cloning

```
// [[Rcpp::export]]
NumericVector positives( NumericVector x ){

    // clone x into y
    NumericVector y = clone(x) ;

    for( int i=0; i< y.size(); i++){
        if( y[i] < 0 ) y[i] = 0 ;
    }
    return y ;
}
```

OPTIMIZING R CODE WITH RCPP

Let's practice!

OPTIMIZING R CODE WITH RCPP

Weighted mean

Romain François
Consulting Datactive, ThinkR

Weighted mean of x with weights w

$$\mu_w(x) = \sum_{i=1}^{n} x_i w_i / \sum_{i=1}^{n} w_i$$

R version

```
# see also ?weighted.mean
weighted_mean_R <- function(x, w){
    sum(x*w) / sum(w)
}
```

x	1.3	3.2	4.2	4.5	6.8
w	1	2	2	1	1

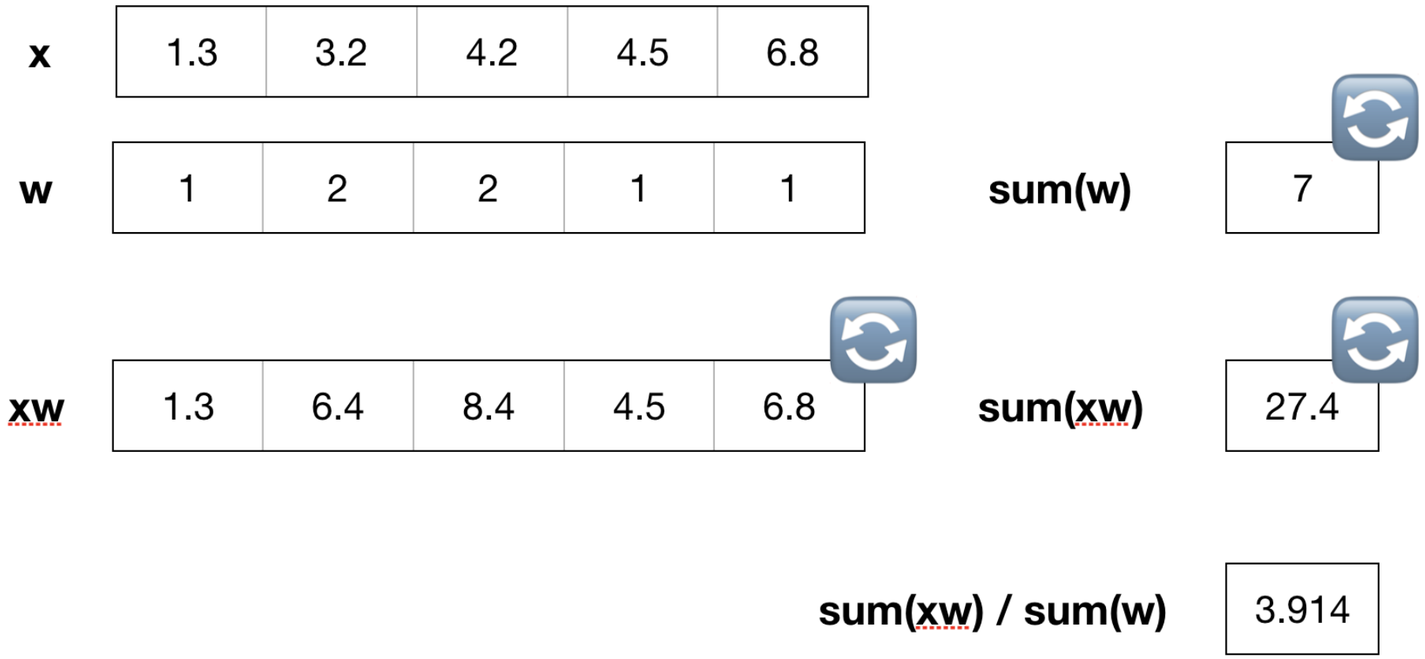
R version

```
# see also ?weighted.mean
weighted_mean_R <- function(x, w){
    sum(x*w) / sum(w)
}
```

x	1.3	3.2	4.2	4.5	6.8
w	1	2	2	1	1
xw	1.3	6.4	8.4	4.5	6.8

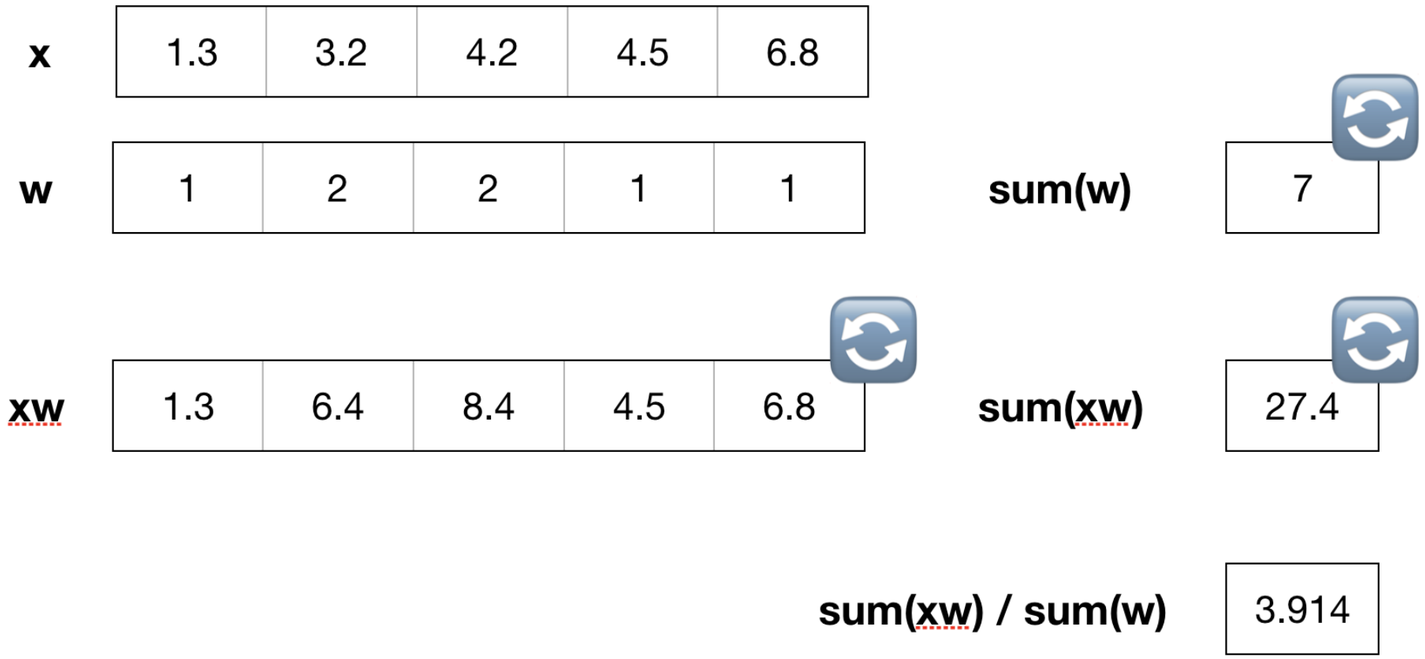
R version

```
# see also ?weighted.mean
weighted_mean_R <- function(x, w){
    sum(x*w) / sum(w)
}
```



R version

```
# see also ?weighted.mean
weighted_mean_R <- function(x, w){
    sum(x*w) / sum(w)
}
```



Inefficient R version

```
weighted_mean_loop <- function(x, w){
    total_xw <- 0
    total_w  <- 0

    for( i in seq_along(x)){
        total_xw <- total_xw + x[i]*w[i]
        total_w  <- total_w  + w[i]
    }

    total_xw / total_w

}
```

Skeleton of a C++ version

```
// [[Rcpp::export]]
double weighted_mean_cpp( NumericVector x, NumericVector w){
    double total_xw = 0.0 ;
    double total_w  = 0.0 ;

    int n = ___ ;

    for( ___ ; ___ ; ___ ){
        // accumulate into total_xw and total_w
    }

    return total_xw / total_w ;

}
```

Missing values

- Testing if a value is a missing value in a numeric vector

```
bool test = NumericVector::is_na(x) ;
```

- The representation of NA in double

```
double y = NumericVector::get_na() ;
```

OPTIMIZING R CODE WITH RCPP

Let's practice!

OPTIMIZING R CODE WITH RCPP

Vectors from the STL

Romain François
Consulting Datactive, ThinkR

Rcpp vectors vs STL vectors

Rcpp vectors

- Thin wrappers around R vectors
- Cannot (cost effectively) change size: data copy every time

STL vectors

- Independent of R vectors
- Cheap to grow and shrink: amortized copies

Extract positives values from a vector

Vectorised R code

```
extract_positives <- function(x){
    x[x>0]
}
```

Inefficient code that grows a vector in a loop

```
extract_positives_loop <- function(x){
    y <- numeric()
    for( value in x){
        if( value > 0 ){
            y <- c(x, y)
        }
    }
    y
}
```

Extract positive values: alternative algorithm

- First ☐ to count the final size

```
NumericVector x ;
int n = x.size() ;
int np = 0 ;
for( int i=0 ; i<n ; i++ ){
    if( ___ ){
        np++ ;
    }
}
```

- Create a vector of the right size

```
NumericVector result(np) ;
```

- Second ☐ to fill the vector

```
for( int i=0, j=0 ; i<n ; i++ ){
    if( ___ ){
        result[j++] = x[i] ;
    }
}
```

Simpler algorithm using the STL

```
// [[Rcpp::export]]
std::vector<double> positives_stl( NumericVector x ){
    std::vector<double> out ;
    out.reserve( x.size() / 2 ) ;

    for( ___ ; ___ ; ___ ){
        if( ___ ){
            out.push_back(___) ;
        }
    }

    return out ;

}
```

OPTIMIZING R CODE WITH RCPP

Let's practice!