Enhancing your Code: Combining R and C++ via Rcpp and RcppArmadillo

Short Course

Lukas Mödl and Erin Sprünken

DAGStat 2025

March 2025

WHAT IS C++?

- C++ is a general-purpose language
- It originates from the C-language
- It is object-oriented and compiled
- Yet has abilities for low-level manipulation

WHY USE C++?

- Consider the following situation:
- You aim to conduct a simulation study with 3 methods, 10^4 simulation runs and 2600 scenarios

WHY USE C++?

- Consider the following situation:
- You aim to conduct a simulation study with 3 methods, 10^4 simulation runs and 2600 scenarios
- resulting in at least $3 \cdot 10^4 \cdot 26 \cdot 10^2 = 78 \cdot 10^6$ computations

WHY USE C++?

- Consider the following situation:
- You aim to conduct a simulation study with 3 methods, 10^4 simulation runs and 2600 scenarios
- resulting in at least $3 \cdot 10^4 \cdot 26 \cdot 10^2 = 78 \cdot 10^6$ computations
- If one computation took one second, then the whole simulation took

WHY USE C++?

- Consider the following situation:
- You aim to conduct a simulation study with 3 methods, 10^4 simulation runs and 2600 scenarios
- resulting in at least $3 \cdot 10^4 \cdot 26 \cdot 10^2 = 78 \cdot 10^6$ computations
- If one computation took one second, then the whole simulation took
  - $78 \cdot 10^6$ seconds

WHY USE C++?

- Consider the following situation:
- You aim to conduct a simulation study with 3 methods, 10^4 simulation runs and 2600 scenarios
- resulting in at least $3 \cdot 10^4 \cdot 26 \cdot 10^2 = 78 \cdot 10^6$ computations
- If one computation took one second, then the whole simulation took
  - $78 \cdot 10^6$ seconds
  - ... or $1.3 \cdot 10^6$ minutes

WHY USE C++?

- Consider the following situation:
- You aim to conduct a simulation study with 3 methods, 10^4 simulation runs and 2600 scenarios
- resulting in at least $3 \cdot 10^4 \cdot 26 \cdot 10^2 = 78 \cdot 10^6$ computations
- If one computation took one second, then the whole simulation took
  - $78 \cdot 10^6$ seconds
  - ... or $1.3 \cdot 10^6$ minutes
  - ... or $2.17 \cdot 10^4$ hours

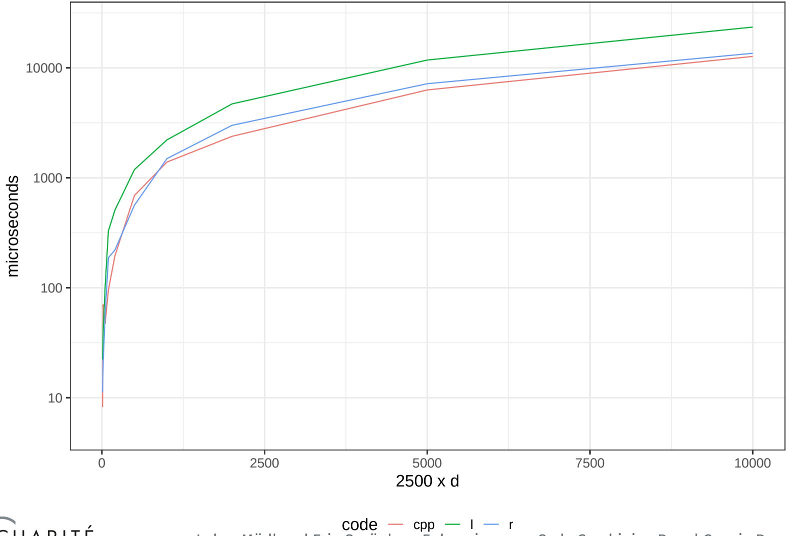
WHY USE C++?

- Consider the following situation:
- You aim to conduct a simulation study with 3 methods, 10^4 simulation runs and 2600 scenarios
- resulting in at least $3 \cdot 10^4 \cdot 26 \cdot 10^2 = 78 \cdot 10^6$ computations
- If one computation took one second, then the whole simulation took
  - $78 \cdot 10^6$ seconds
  - ... or $1.3 \cdot 10^6$ minutes
  - ... or $2.17 \cdot 10^4$ hours
  - ... or 902.7 days

WHY USE C++?

- Consider the following situation:
- You aim to conduct a simulation study with 3 methods, 10^4 simulation runs and 2600 scenarios
- resulting in at least $3 \cdot 10^4 \cdot 26 \cdot 10^2 = 78 \cdot 10^6$ computations
- If one computation took one second, then the whole simulation took
    - $78 \cdot 10^6$ seconds
    - ... or $1.3 \cdot 10^6$ minutes
    - ... or $2.17 \cdot 10^4$ hours
    - ... or 902.7 days
- We can do better!

EXAMPLE: COLUMN-WISE VARIANCE COMPUTATION



DIFFERENCE TO C AND FORTRAN

- R provides native interfaces to the languages C and FORTRAN
- Both are highly efficient programming languages that operate close to the machine-level

DIFFERENCE TO C AND FORTRAN

- R provides native interfaces to the languages C and FORTRAN
- Both are highly efficient programming languages that operate close to the machine-level
  - ... e.g., C has been used to write operating systems such as UNIX
  - ... e.g., FORTRAN has been (and is still) used for scientific computing in a variety of fields – For example at NASA or the BLAS and LAPACK libraries

DIFFERENCE TO C AND FORTRAN

- R provides native interfaces to the languages C and FORTRAN
- Both are highly efficient programming languages that operate close to the machine-level
  - ... e.g., C has been used to write operating systems such as UNIX
  - ... e.g., FORTRAN has been (and is still) used for scientific computing in a variety of fields – For example at NASA or the BLAS and LAPACK libraries
- But: They lack certain capabilities that C++ does offer

DIFFERENCE TO C AND FORTRAN

- R provides native interfaces to the languages C and FORTRAN
- Both are highly efficient programming languages that operate close to the machine-level
  - ... e.g., C has been used to write operating systems such as UNIX
  - ... e.g., FORTRAN has been (and is still) used for scientific computing in a variety of fields – For example at NASA or the BLAS and LAPACK libraries
- But: They lack certain capabilities that C++ does offer
- While C++ is indeed a successor of C (originally an extension), it is much easier to understand and write

DIFFERENCE TO C AND FORTRAN

- R provides native interfaces to the languages C and FORTRAN
- Both are highly efficient programming languages that operate close to the machine-level
  - ... e.g., C has been used to write operating systems such as UNIX
  - ... e.g., FORTRAN has been (and is still) used for scientific computing in a variety of fields – For example at NASA or the BLAS and LAPACK libraries
- But: They lack certain capabilities that C++ does offer
- While C++ is indeed a successor of C (originally an extension), it is much easier to understand and write
- This is mainly due to the object-oriented nature of C++ (More on this later!)

WHEN TO NOT USE C++?

- While C++ offers large gains w.r.t to computation time and efficiency, we have to be aware ...

WHEN TO NOT USE C++?

- While C++ offers large gains w.r.t to computation time and efficiency, we have to be aware ...
  - ... about portability

WHEN TO NOT USE C++?

- While C++ offers large gains w.r.t to computation time and efficiency, we have to be aware ...
  - ... about portability
  - ... about simplicity

WHEN TO NOT USE C++?

- While C++ offers large gains w.r.t to computation time and efficiency, we have to be aware ...
    - ... about portability
    - ... about simplicity
    - ... about general efficiency

WHEN TO NOT USE C++?

- While C++ offers large gains w.r.t to computation time and efficiency, we have to be aware ...
  - ... about portability
  - ... about simplicity
  - ... about general efficiency
- What does that mean?

WHEN TO NOT USE C++?

- While C++ offers large gains w.r.t to computation time and efficiency, we have to be aware ...
  - ... about portability
  - ... about simplicity
  - ... about general efficiency
- What does that mean?
  - Are colleagues (or other people) able to run the code with little to no effort?

WHEN TO NOT USE C++?

- While C++ offers large gains w.r.t to computation time and efficiency, we have to be aware ...
  - ... about portability
  - ... about simplicity
  - ... about general efficiency
- What does that mean?
  - Are colleagues (or other people) able to run the code with little to no effort?
  - Are you, your colleagues (or other people involved) able to understand the code?

WHEN TO NOT USE C++?

- While C++ offers large gains w.r.t to computation time and efficiency, we have to be aware ...
    - ... about portability
    - ... about simplicity
    - ... about general efficiency
- What does that mean?
    - Are colleagues (or other people) able to run the code with little to no effort?
    - Are you, your colleagues (or other people involved) able to understand the code?
    - Most importantly: Is the task to be solved worth the effort to write more complicated code?

WHEN TO NOT USE C++?

- While C++ offers large gains w.r.t to computation time and efficiency, we have to be aware ...
  - ... about portability
  - ... about simplicity
  - ... about general efficiency
- What does that mean?
  - Are colleagues (or other people) able to run the code with little to no effort?
  - Are you, your colleagues (or other people involved) able to understand the code?
  - Most importantly: Is the task to be solved worth the effort to write more complicated code?
- Usually, simple tasks are usually not worth to be tackled with C, FORTRAN or C++

DIFFERENCES BETWEEN C++ AND R

- There are some differences between C++ and R

DIFFERENCES BETWEEN C++ AND R

- There are some differences between C++ and R
- While R is interpreted, C++ is compiled

DIFFERENCES BETWEEN C++ AND R

- There are some differences between C++ and R
- While R is interpreted, C++ is compiled
  $\Rightarrow$ R does immediately understand, translate and run lines of code; C++ needs a compiler to translate code into machine-language and execute it

DIFFERENCES BETWEEN C++ AND R

- There are some differences between C++ and R
- While R is interpreted, C++ is compiled
  $\Rightarrow$ R does immediately understand, translate and run lines of code; C++ needs a compiler to translate code into machine-language and execute it
- While R is mostly procedural, C++ is mostly object-oriented

DIFFERENCES BETWEEN C++ AND R

- There are some differences between C++ and R
- While R is interpreted, C++ is compiled
  $\Rightarrow$ R does immediately understand, translate and run lines of code; C++ needs a compiler to translate code into machine-language and execute it
- While R is mostly procedural, C++ is mostly object-oriented
  $\Rightarrow$ R is focused on variables, data structures and functions of these; C++ is focused on objects with classes and class-specific attributes and functions

DIFFERENCES BETWEEN C++ AND R

- There are some differences between C++ and R
- While R is interpreted, C++ is compiled
  $\Rightarrow$ R does immediately understand, translate and run lines of code; C++ needs a compiler to translate code into machine-language and execute it
- While R is mostly procedural, C++ is mostly object-oriented
  $\Rightarrow$ R is focused on variables, data structures and functions of these; C++ is focused on objects with classes and class-specific attributes and functions
- While R can understand ”on-the-fly” declarations, C++ can not

DIFFERENCES BETWEEN C++ AND R

- There are some differences between C++ and R
- While R is interpreted, C++ is compiled
  $\Rightarrow$ R does immediately understand, translate and run lines of code; C++ needs a compiler to translate code into machine-language and execute it
- While R is mostly procedural, C++ is mostly object-oriented
  $\Rightarrow$ R is focused on variables, data structures and functions of these; C++ is focused on objects with classes and class-specific attributes and functions
- While R can understand "on-the-fly" declarations, C++ can not
  $\Rightarrow$ In R, we can easily introduce or re-declare variables; C++ needs to know about variables and their type before compilation

WHAT DOES OBJECT-ORIENTED MEAN?

- ”Object-oriented programming (OOP) is a programming paradigm based on the concept of objects, which can contain data and code: data in the form of fields (often known as attributes or properties), and code in the form of procedures (often known as methods). In OOP, computer programs are designed by making them out of objects that interact with one another.” (Wikipedia, 10.10.2024)

WHAT DOES OBJECT-ORIENTED MEAN?

- ”Object-oriented programming (OOP) is a programming paradigm based on the concept of objects, which can contain data and code: data in the form of fields (often known as attributes or properties), and code in the form of procedures (often known as methods). In OOP, computer programs are designed by making them out of objects that interact with one another.” (Wikipedia, 10.10.2024)
- For examples, variables are stored in an object that has certain properties and methods attached to it

WHAT DOES OBJECT-ORIENTED MEAN?

- ”Object-oriented programming (OOP) is a programming paradigm based on the concept of objects, which can contain data and code: data in the form of fields (often known as attributes or properties), and code in the form of procedures (often known as methods). In OOP, computer programs are designed by making them out of objects that interact with one another.” (Wikipedia, 10.10.2024)
- For examples, variables are stored in an object that has certain properties and methods attached to it
- Usually, objects are defined via classes

NOW, WHAT ARE CLASSES?

- While R also has classes, we don't really *see* them

NOW, WHAT ARE CLASSES?

- While R also has classes, we don't really *see* them
- In C++, we may use the classes more actively

NOW, WHAT ARE CLASSES?

- While R also has classes, we don’t really *see* them
- In C++, we may use the classes more actively
- Classes are abstract concepts that pre-define certain methods or properties for objects

NOW, WHAT ARE CLASSES?

- While R also has classes, we don't really *see* them
- In C++, we may use the classes more actively
- Classes are abstract concepts that pre-define certain methods or properties for objects

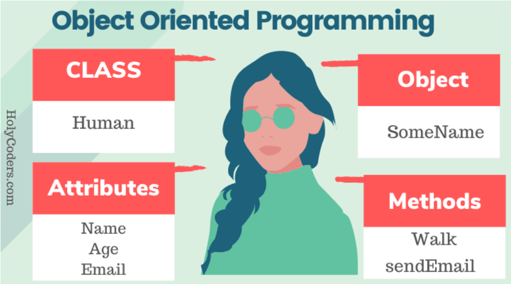


Figure 1: Source: https://2531.medium.com/class-and-instance-attributes-4b4e8eee36b8, called 21.02.2025

DECLARATIONS, INDEXING AND CONTROL-FLOW

- Most importantly: Contrary to R, C++ needs to know where a line of code ends

DECLARATIONS, INDEXING AND CONTROL-FLOW

- Most importantly: Contrary to R, C++ needs to know where a line of code ends
- After every line of code or statement you must put a semicolon (;)

DECLARATIONS, INDEXING AND CONTROL-FLOW

- Most importantly: Contrary to R, C++ needs to know where a line of code ends
- After every line of code or statement you must put a semicolon (;)
- As mentioned earlier, variables must be declared before compilation (and also before first appearance)

DECLARATIONS, INDEXING AND CONTROL-FLOW

- Most importantly: Contrary to R, C++ needs to know where a line of code ends
- After every line of code or statement you must put a semicolon (;)
- As mentioned earlier, variables must be declared before compilation (and also before first appearance)
- That means, if – at some point in your code – you want to use a variable `x_dagstat`, you have to declare it together with its datatype beforehand

DECLARATIONS, INDEXING AND CONTROL-FLOW

- Most importantly: Contrary to R, C++ needs to know where a line of code ends
- After every line of code or statement you must put a semicolon (;)
- As mentioned earlier, variables must be declared before compilation (and also before first appearance)
- That means, if – at some point in your code – you want to use a variable `x_dagstat`, you have to declare it together with its datatype beforehand

Code 5 – Declaring a Variable

```
int x_dagstat;
// Here goes some code
x_dagstat = 5;
```

LOOPS

- In R, we declare a `for`-loop like this:

Code 6 – R `for` Loop

```r
for(i in 1:20){
    print(i)
}
```

- In C++, this works differently:

Code 7 – C++ `for` Loop

```cpp
for(int i=0; i<20; i++){
    Rcpp::Rcout << i;
}
```

- Note the differences in creating the loop!

LOOPS

- The C++ `for`-Loop has three statements:
  1. Execute before start (once)
  2. Condition to execute block
  3. Execute after block (every time)
- E.g.:

Code 8 – Another C++ `for` Loop

```
for(int i=5; i<16; i=i+3){
    Rcpp::Rcout << i;
}
```

CONDITIONS

- In R, we declare a `for`-loop like this:

Code 9 – R `if` Condition

```r
i = 0
if(i <= 10) {
      print(i)
}
```

- Conditions in C++ conditions are checked in the same way:

Code 10 – C++ `if` Condition

```cpp
int i = 0;
if(i <= 10){
    Rcpp::Rcout << "This is obviously true";
}
```

SOME REMARKS

- Usually, there is no vectorization in C++

SOME REMARKS

- Usually, there is no vectorization in C++
  $\Rightarrow$ Need to loop through most objects manually

SOME REMARKS

- Usually, there is no vectorization in C++
  $\Rightarrow$ Need to loop through most objects manually
- But: Maybe we can, at least, use linear algebra to make things faster?!

SOME REMARKS

- Usually, there is no vectorization in C++
  $\Rightarrow$ Need to loop through most objects manually
- But: Maybe we can, at least, use linear algebra to make things faster?!
  $\Rightarrow$ C++ library Armadillo

WHAT IS ARMADILLO?

Figure 2: Source: https://en.wikipedia.org/wiki/Armadillo#/media/File:Nine-banded_Armadillo.jpg, on Feb 21, 2025

- Armadillo is a famous C++ library for linear algebra
- It makes use of BLAS low-level routines (these are also implicitly used by R, e.g., `axpy`, `gemv`, `gemm`)
- Armadillo provides a huge library of objects and functions
- Since Armadillo is perfectly suited for linear algebra, we ignore the vector- and matrix types of standard Rcpp and focus on Armadillo
- We recommend (as well as in R): Using namespaces, as for Armadillo the namespace `arma::`

VECTORS AND MATRICES

- (Column-)Vector: `vec`
- Rowvector: `rowvec`
- Matrix `mat`
- Similar to R, Armadillo constructs matrices column by column

VECTOR- AND MATRIX-INITIALIZATION

- Initialize an empty vector named `x` of length 10: `arma::vec x(10);`
- Initialize an empty matrix named `y` of size 10×10: `arma::mat y(10, 10);`
- Both the `arma::mat` and `arma::vec` class have useful member functions: `.ones()`, `.zeros()`, `.fill()`, `.randu()`, `.randn()`, `.t()`, ...

Code 11 – Example Vectors and Matrices

```
// Generate vectors and matrices:
arma::vec x(10), x_one, x_42;
arma::mat y_zero(3,3), y_unif(5,5), y_norm(4,6);
// Fill vectors with 1.0 and 42.0:
x_one.ones(5);
x_42.fill(42.0);
// Fill matrices with 0.0, uniform, and standard normal random numbers:
y_zero.zeros();
y_unif.randu();
y_norm.randn();
```

MATRIX COMPUTATIONS

- Generally, computations are identical to R, **except** multiplication
- in Armadillo, the `*` operator is the matrix multiplication (as `%*%` is in R) and `%` is the elementwise multiplication (as `*` in R)
- Transposing is done via the member-function `.t()`
- Similar to R, equality comparisons (`==`, `!=`, `>=`, `<=`) are not recommended for floating point numbers due to issues with floating point precision

SOURCING C++-CODE – INLINE

- The function `cppFunction()` provides a way of creating a single C++ function in R itself
- Consider this as some sort of inline-C++-creation

Code 12 – Inline C++ (R-Level)

```
example_code = "arma::vec square(const arma::vec x){
    arma::vec y = arma::pow(x,2);
    return y;
}"
Rcpp::cppFunction(example_code, depends = "RcppArmadillo")
```

- Function writing is similar to R, but we have to remember the declaration of variables
- Since we are using (Rcpp)Armadillo, the import into R needs to know about that dependency

TASK I – MATRIX COMPUTATIONS

Let X, Y be vectors of length n and A, B matrices with dimension $m \times n$.

Task (15 Min): Write code to compute:

(a) $X + Y$

(b) $X^\top Y$

(c) $XY^\top$

(d) $||X - Y||_2$ (2-norm)

(e) $||Y - X||_5$ (5-norm)

(f) $AB^\top$

(g) $A \odot B$ (element-wise product)

(h) $a_{ij} > b_{ij} \; \forall \; i, j$

TASK I – MATRIX COMPUTATIONS – SOLUTIONS

All other solutions available on Github

Code 13 – Solution for (d)

```
norm_2 = "double sqNorm(const arma::vec x, const arma::vec y){
  arma::vec z = x - y;
  arma::vec z_sq = arma::pow(z,2);
  double result = std::sqrt(arma::sum(z_sq));
  return result;
  }"
Rcpp::cppFunction(norm_2, depends = "RcppArmadillo")
```

IMPORTANT FUNCTIONS

- `arma::solve(arma::mat, arma::mat)` solves the equation $\mathbf{AX} = \mathbf{B}$ for $\mathbf{X}$
- `arma::inv(arma::mat)` computes the inverse of a matrix $\mathbf{X}$
- `arma::mean(arma::mat, int)` column/row-wise mean
- `arma::var(arma::mat, int)` column/row-wise variance

From the standard C/C++ library:

- `std::tgamma(double)` Gamma function $\Gamma(x)$
- `std::erf(double)` Error function $\mathrm{erf}(x)$
- `std::sin(double)`, `std::cos(double)`, `std::tan(double)` standard trig functions $\sin(x)$, $\cos(x)$, $\tan(x)$

SOURCING C++ CODE

- We have already learned about inline sourcing
- Now, we will understand the full scale sourcing
- Especially for larger projects, this is very meaningful
- On the R-level, it's fairly simple; Consider a C++-file named `simu1.cpp`
- We import this file in R using `Rcpp::sourceCpp(file = "simu1.cpp")`
- On the C++-Level, we need to do a few things...

WRITING A C++-FILE FOR R

- The first line(s) are the `#include` statements
- These serve to include header-files
- To use RcppArmadillo, we need at least `#include <RcppArmadillo.h>`
- Right after the `#include` statements comes the dependency statement for RcppArmadillo: `// [[Rcpp::depends(RcppArmadillo)]]`
- Finally, functions that should be exported to the R-Level need an export statement: `// [[Rcpp::export()]]`

WRITING A C++-FILE FOR R – EXAMPLE

Create a file `ex1_de.cpp`:

Code 14 – (d) and (e) – C++-Level

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

// [[Rcpp::export()]]
double pNorm(const arma::vec x, const arma::vec y, const double p = 2.0){
    arma::vec z = arma::abs(x - y);
    arma::vec z_p = arma::pow(z,p);
    double result = std::pow(arma::sum(z_p), (1.0/p));
    return result;
}
```

Code 15 – (d) and (e) – R-Level

```r
sourceCpp("ex1_de.cpp")
```

WRITING A C++-FILE FOR R – EXAMPLE

- After using `Rcpp::sourceCpp`, your environment in R should show a function called `pNorm`
- We can now call that function in R (and double check with using base R functions)

Code 16 – Exercise 1 (d) and (e) in R

```
# Create some example data:
a = rnorm(10)
b = rnorm(10, 3, 2.5)

# Calling the cpp-Function first for (d) then for (e):
pNorm(x = a, y = b) # (d)
pNorm(a, b, 5) # (e)

# Base R versions
(sum(abs(a-b)^2))^(1/2) # (d)
(sum(abs(a-b)^5))^(1/5) # (e)
```

TASK II – MULTIPLE LINEAR REGRESSION

- Consider the following data:
    - Dependent variable: Body height in cm
    - Independent variables:
        - Age in years
        - Weight in kg
        - Average height of parents in cm
        - Average weekly time of physical activity in hours
- Remember the Linear Regression Equation: $\mathbf{y} = \mathbf{X}\beta + \epsilon$ ...
- with the least squares estimator: $\widehat{\beta} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{Y}$ ...
- which could be considered a function $f(\mathbf{X}, \mathbf{Y})$
- **Task (20 Min)**: Write C++-Code that takes a matrix **X** and vector **y** as arguments and returns the least squares estimator $\widehat{\beta}$ from above; Import it into R.

TASK II – MULTIPLE LINEAR REGRESSION – SOLUTIONS

- We begin by creating a file `ex2.cpp`:

Code 17 – Exercise 2 – C++-Level

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

// [[Rcpp::export()]]
arma::vec beta_est(const arma::mat X, const arma::vec y){
    arma::mat xx = X.t() * X;
    arma::mat xy = X.t() * y;
    arma::vec beta_hat = arma::inv_sympd(xx) * xy; // inv(xx) should work, too
    return beta_hat;
}
```

TASK II – MULTIPLE LINEAR REGRESSION – SOLUTIONS

Code 18 – Exercise 2 – R-Level

```r
# Create some example data:
set.seed(1234)
x = matrix(c(rep(1, 30), sample(10:60, size = 30, replace = T), rnorm(30,70,10), rnorm(30,170,15),
rexp(30, 1/3)), ncol = 5) # this is the X-matrix data
y =  x %*% c(0.15, 0.05, 0.05, 0.9, 0.1) + rnorm(30)# y = X beta + epsilon

# Import Cpp
Rcpp::sourceCpp("ex2.cpp")

# R vs. Cpp
beta_r = solve(crossprod(x)) %*% crossprod(x, y)
beta_cpp = beta_est(x, y)
```

TASK III – MULTIPLE LINEAR REGRESSION SIMULATION

- Consider the linear regression model $\mathbf{y} = \mathbf{X}\beta + \epsilon$ from the previous example
- Goal: implement a simulation for β (assume $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and deterministic $\mathbf{X}$)
- **Task (20 Min)**: For this task, follow these steps:
  1. Create a C++-File `ex3.cpp`
  2. Write a function `data_gen` that takes $\mathbf{X}$ and the true β as an argument and computes $\mathbf{y} = \mathbf{X}\beta + \varepsilon$ with ϵ being sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
  3. Write a function `arma::mat sim_reg` that, at least, takes the number of samples as an argument and returns a matrix of coefficients (i.e., each row or column corresponds to one estimated $\widehat{\beta}$)
- Hints:
  - Order of functions is important! Any function to be used has to be declared first!
  - Use your `beta_est` function from the previous exercise

TASK III – MULTIPLE LINEAR REGRESSION SIMULATION – SOLUTIONS

Code 19 – Exercise 3 – C++-Level (file: `ex3.cpp`)

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

arma::mat data_gen(const arma::mat X, const arma::vec beta){
    arma::vec epsilon = arma::randn(X.n_rows);
    arma::mat y = epsilon + X * beta + epsilon;
    return y;
}

// insert the beta_est function from before at this place

// [[Rcpp::export()]]
arma::mat sim_reg(const arma::mat X, const arma::vec beta, const int n_sim = 1e4){
    arma::mat betas(X.n_cols, n_sim);
    for(int i = 0; i < n_sim; i++){
        betas.col(i) = beta_est(X, data_gen(X, beta));
    }
    return betas;
}
```

TASK III - MULTIPLE LINEAR REGRESSION SIMULATION – SOLUTIONS

Code 20 – Exercise 3 – R-Level

```r
# Create some example data:
set.seed(7)
x = matrix(c(rep(1, 30), sample(10:60, size = 30, replace = T), rnorm(30,70,10), rnorm(30,170,15),
rexp(30, 1/3)), ncol = 5) # this is the X-matrix data
beta = c(0.15, 0.05, 0.05, 0.9, 0.1)

# Import Cpp
Rcpp::sourceCpp("ex3.cpp")

# R vs. Cpp
sim_beta_r = sapply(1:1e4, function(sim) solve(crossprod(x)) %*% crossprod(x, x %*% beta +
rnorm(nrow(x))))
sim_beta_cpp = sim_reg(x, beta)
```

TASK IV – GROUPWISE MEANS

- Consider a standard task in data analysis: Computing the means (or other statistics) for each subgroup
- Your data might look like this (here: `iris` dataset in R):

Sepal.Length	Species
5.1	setosa
⋮	⋮
7.0	versicolor
⋮	⋮
6.3	virginica

- **Task (15 Min)**: Write a C++-Function `tapply_cpp` that takes two arguments, a vector `x` and a vector of groups, and returns a vector of group-wise means
- Hints:
  - You want to consider the Armadillo function `arma::find`
  - Armadillo also offers a function `arma::unique`, similar to R

TASK IV – GROUPWISE MEANS – SOLUTIONS

Code 21 – Exercise 4 – C++-Level (file: ex4.cpp)

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

// [[Rcpp::export()]]
arma::vec tapply_cpp(const arma::vec x, const arma::vec grp){
    arma::vec grps = arma::unique(grp);
    int d = grps.n_elem;
    arma::vec results(d);

    for(int i = 0; i < d; i++){
        double current_grp = grps(i);
        arma::uvec ind = arma::find(grp == current_grp);
        results(i) = arma::mean(x.elem(ind));
    }
    return results;
}
```

TASK IV – GROUPWISE MEANS – SOLUTIONS

Code 22 – Exercise 4 – R-Level

```r
data(iris)
Rcpp::sourceCpp("ex4.cpp")

# C++
tapply_cpp(iris$Sepal.Length, iris$Species)
# Check with R
tapply(iris$Sepal.Length, iris$Species, mean)
```

TASK V – T-TEST SIMULATION

- Simulate 10^4 datasets and compute the t-test statistic for mean comparison on each data set
- For this **Task (20 Min)**, follow these steps:
  1. Write a function in C++ that:
     1.1 Takes data as an argument (and everything necessary to distinguish the two samples)
     1.2 Computes a t-test statistic for each sample
     1.3 Stores each test statistic in a vector
     1.4 Returns the vector of test statistics
  2. Write a similar function in R
  3. Generate data in R
  4. Invoke both functions within R to compare the results

TASK V – T-TEST SIMULATION – SOLUTIONS

Create a file `ex5.cpp`:

Code 23 – Exercise 5 – C++-Level

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

// [[Rcpp::export()]]
arma::mat t_test_cpp(const arma::mat x, const int n1, const int n2) {
    arma::mat x1 = x.rows(0, n1 - 1), x2 = x.rows(n1, n1 + n2 - 1);
    arma::rowvec delta = arma::mean(x1, 0) - arma::mean(x2, 0);
    arma::rowvec ss1 = ((double)n1 - 1.0) * arma::var(x1, 0), ss2 = ((double)n2 - 1.0) *
    arma::var(x2, 0);
    arma::rowvec varp = (ss1 + ss2) / ((double)n1 + (double)n2 - 2.0);
    return arma::trans(delta / (arma::sqrt(varp * (1.0 / (double)n1 + 1.0 / (double)n2))));
}
```

TASK V – T-TEST SIMULATION – SOLUTIONS

In R:

Code 24 – Exercise 5 – R-Level

```
t_test_r = function(x, n1, n2) {
    x1 = x[1:n1,,drop = FALSE]; x2 = x[(n1+1):(n1+n2),,drop =FALSE]
    delta = colMeans(x1) - colMeans(x2)
    ss1 = n1 * (colMeans(x1**2) - colMeans(x1)**2); ss2 = n2 * (colMeans(x2**2) - colMeans(x2)**2)
    varp = (ss1 + ss2) / (n1 + n2 - 2)
    return(delta / sqrt(varp * (1 / n1 + 1 / n2)))
}
```

TASK V – T-TEST SIMULATION – SOLUTIONS

Generate data and invoke functions:

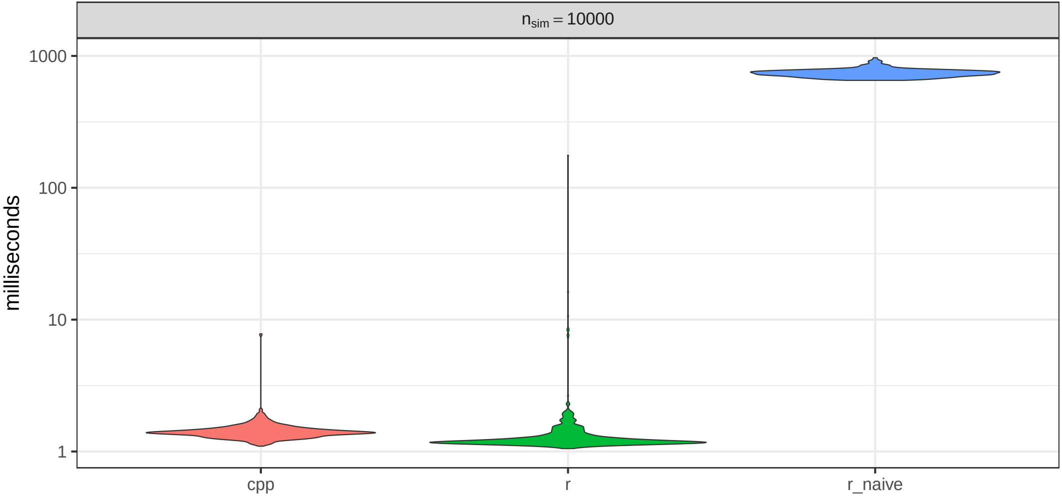
Code 25 – Exercise 5 – R-Level

```r
# Compile the cpp function
Rcpp::sourceCpp("ex5.cpp")

# Generate data
nsim = 1e4
n_1 = 10; n_2 = 12
data_sim = matrix(rnorm((n_1 + n_2) * nsim), ncol = nsim)

# Run
r_results = t_test_r(data_sim, n_1, n_2)
cpp_results = t_test_cpp(data_sim, n_1, n_2)
# do not run:
#r_naive_results = apply(data_sim, 2, \(x) t.test(x[1:n_1], x[-(1:n1)], var.equal = TRUE)£stat]
```

COMPUTATION TIME – A DEMONSTRATION



PARALLELIZATION

- Often (especially for simulations), the same computations are performed, while only certain parameters vary
- Example: Type-I error analysis of t-test using bootstrap:
  1. Create n_{sim} datasets satisfying null hypothesis
  2. Run a t-test on all of these datasets
  3. Compute the rejection rate (i.e. empirical type-I error rate)
- These steps will always be the same (Embarrassingly Parallel Workload)
- However, we might vary the sample sizes, variances etc.
- We could choose to parallelize over:
  (a) Different Scenarios (i.e. n_{sim} computations per core/thread)
  (b) Different simulations (i.e. for each scenario we split the n_{sim} computations over different cores/threads)

PARALLELIZATION IN R – EXAMPLE

Code 26 – Linear Regression Simulation Example – R-level

```r
reg_r_par = function(X, N, cores = 4) {
    Y = matrix(ncol = N, nrow = nrow(X))
    cl = parallel::makeCluster(cores);
    parallel::clusterExport(cl, varlist = c("X", "Y"),  envir = environment())
    Y = parallel::parSapply(cl, 1:N, \(i) solve(crossprod(X)) %*% crossprod(X, rnorm(nrow(X))))
    parallel::stopCluster(cl)
    return(Y)
}
```

PARALLELIZATION IN C++ – EXAMPLE

Code 27 – Linear Regression Simulation Example – C++-level

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

// [[Rcpp::export]]
arma::mat reg_cpp_par(arma::mat X, int N, unsigned int threads = 4) {
    arma::mat Y(X.n_cols, N);
    #pragma omp parallel for schedule(dynamic) num_threads(threads)
    for(int i = 0; i < N; ++i) {
        Y.col(i) = arma::solve(X, arma::vec(X.n_rows, arma::fill::randn));
    }
    return Y;
}
```

PARALLELIZATION EXAMPLE RESULTS

Code 28 – Linear Regression Simulation Example – Comparison

```
set.seed(42) # seed
X = cbind(1, matrix(rnorm(2500), ncol = 25))

nsim = 1e4
cores = 4 # parallel::detectCores()

r_res = reg_r_par(X, nsim, cores)
cpp_res = reg_cpp_par(X, nsim, cores)
```

CAVEATS

- Overhead
- Thread safety
- Multithreading interference (e.g., by OpenBLAS)
- Only Embarrassingly Parallel Workloads
- Complexity

FURTHER ADVICES

- `Rcpp::List` and other Rcpp-Features outside of Armadillo (might be useful to mimic R-behavior)
- Include other libraries (e.g., boost)
- Classes
- Pointers and references (a little bit complicated, but could add efficiency)
- Other OpenMP directives: `#pragma omp parallel for schedule(static)`, `#pragma omp parallel for private()`, ...
- Seeding: `std::mt19937_64` with member function `.seed(unsigned int)`

SUMMARY

- Fast, efficient, portable
- Similar to R to some extent
- Use cases:
  - Simulation Studies
  - Repetitive Tasks (e.g. for loops)
  - Computationally intensive tasks
- Don't cases:
  - If you can do it with matrix computations in R, keep it in R
  - Generally: Keep it simple
  - Implementation time much larger than computation time gain

RESOURCES FOR PRACTITIONERS

- Stackoverflow (!!)
- C++ Armadillo
- Rcpp Gallery
- Rcpp (CRAN)
- Rcpp (Hadley Wickham)
- This Tutorial's GitHub Repo
- CppReference

SOURCES / BIBLIOGRAPHY

1. C Language (Wikipedia)
2. FORTRAN (Wikipedia)
3. C++ (Wikipedia)
4. R Language (Wikipedia)
5. C++ (ISO)
6. C++ Armadillo
7. Dirk Eddelbuettel