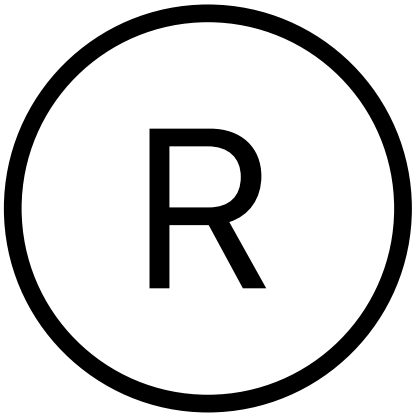
Graduate program Digital Public Health

université de BORDEAUX

HeavyR: Fast and efficient R code made easy

HeavyR is an **R package designed to streamline the development of fast and efficient R code**. It equips researchers and scientists with tools for code organization, optimization, parallelization, and collaborative development using GitHub.



This course delves into the world of efficient R code development, equipping scientists and researchers with the latest tools. We'll explore **writing R packages** for better code organization, delve into **profiling techniques** to pinpoint performance bottlenecks, and **utilize Rcpp for targeted code optimization**. Additionally, the course covers **parallelization for faster computations** and introduces **GitHub for collaborative development and continuous code checking** via GitHub Actions, all aimed at empowering you to write high-performance R code.

Prerequisites

A good working knowledge of R will be necessary.
Participants are expected to be proficient with functional programing and be familiar with the concept of R package.

WARNING: this course is not suitable for R beginners!

HeavyR: Fast and efficient R code made easy

PROGRAM

1. Brief recap on writing R packages as a useful tool for code development
2. How to measure computation time and profile code to identify bottlenecks and compare different implementations
3. Use Rcpp to optimize the code portion that should be
4. Easily parallelize one's code
5. Use GitHub to collaboratively develop open-source R code
6. Use GitHub actions to both perform continuous check of your code and power a companion website

Methodology :

1. This course is a mix of lessons and computer practical sessions
2. A laptop with an up-to-date installations of Rstudio and R, as well as working C++ and fortran compilers (with Rtools on Windows, macrtools on macOS, gcc on Linux) is necessary
3. Attendance to all sessions is mandatory

Training Objectives

At the end of this course, you will be able to:

- identify computational bottlenecks in one's code
- optimize a function using C++ integration through Rcpp
- harvest multicore's speed by easy parallelization of your code
- evaluate and compare speed-up gains of competing implementations

Registration deadline :
Date :
Schedule :
Place :