McGill initiative in Computational Medicine

Mission : aims to deliver inter-disciplinary research programs and empower the use of data in health research and health care delivery

McGill initiative in Computational Medicine

R - Beyond the Basics

Yi Lian

Audrey Baguette

McGill initiative in Computational Medicine

Efficiency

- Coding
- Computing

Why?

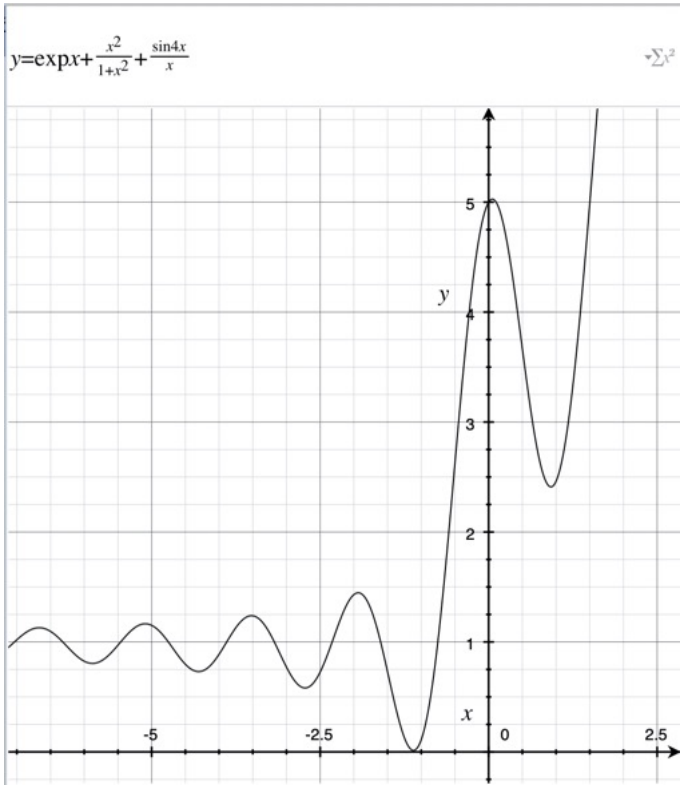
- Coding – short and clean codes make everything easier
  - Edit
  - Debug
  - Share
  - Collaborate
- Computing – short runtime, even make impossible possible
  - Big data
  - Machine learning/AI
  - Slow computer!
  - Deadlines!!!

General Rules

- It takes resources to run programs
  - Processor (CPU, GPU)
  - Memory (RAM)
  - Time
  - …
- Efficient coding ≠ Efficient computing
  - Short codes do not necessarily lead to short runtime
- Avoid duplicated operations, especially expensive ones
  - Matrix multiplication/inversion, etc.

Minimize $\exp x + \frac{x^2}{1+x^2} + \frac{\sin 4x}{x}$?

- Shortest code
  - How?
  - Will it be fast?
- Shortest runtime
  - How?
    - Newton's method?
    - Gradient descent?
  - How long is the code?



R Rules

- R emphasizes flexibility
  - Very good for research
- R is better with vectorized operations but not loops
- By default, R uses one CPU core
- Use established R functions and packages
  - These functions usually make coding and computing more efficient at the same time
    - Some of them have core computations written in other languages, e.g. C, C++, Fortran
    - Some of them implement parallel (multi-core) computing

Today's plan

- Efficient coding
  - R objects (if necessary)
  - Powerful R functions
    - `aggregate()`, `by()`, `apply()` family
    - `ifelse()`
    - `cut()`, `split()`
  - Writing our own functions
    - `function()`
- Efficient computing (will not be covered today)
  - Parallel computing
  - Mixed programming (`Rcpp`, etc.)

Hands-on -> https://github.com/ly129/MiCM2020