Biostatistics 615 - Statistical Computing

Lecture 1
Introduction to Statistical Computing

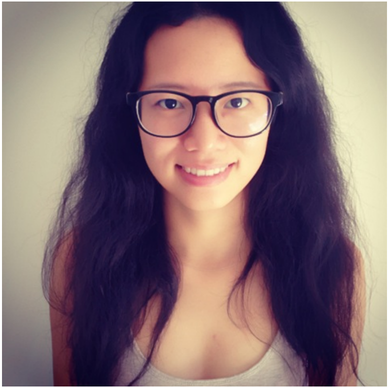
Jian Kang

September 8, 2015

Instructor

- Jian Kang, PhD
- Assistant Professor of Biostatistics
- Email: jiankang@umich.edu
- Office: 3651 SPH I
- Office hour: Thursday 10:15AM – Noon, or By Appointment
- Webpage: http://www-personal.umich.edu/~jiankang/

Graduate Student Instructor



- Pin Li, B.S. Master Student of Biostatistics
- Email: pinli@umich.edu
- Office hour: twice weekly, two hours (based on the survey)
- Room: TBD

Objective

- Understand computational issues related to the implementation of statistical methods, from basics of programming languages to inner workings of sophisticated statistical methods.
- Learn basic C++ programming and advanced R techniques

A Brief History

- It was taught by



Goncalo Abecasis	Hyun Min Kang	Hui Jiang
2004 – 2010	2011 – 2012	2013 – 2014

- The current course materials are adapted from previous instructors

Target Audience

- Students may have little experience in C++ programming.
  - But students must be strongly motivated to learn C++ programming
  - Students with limited experience in C++ might spend many additional hours than other students to accomplish homework and course project
- Students are expected to know basics of R language
  - Students will learn advanced R techniques and use them to accomplish homework and course project
- Students should be familiar with linear algebra (matrix theory), multivariate calculus, basic concept of probability distribution, hypothesis testing, and simple regression.
  - Biostatistics 601 or equivalent is required prior to or in parallel to taking Biostatistics 615.

Clarifications

- This is NOT a course about C++ programming
  - But some homework assignments require C++ programming.
  - You must learn C++ programming on your own
- During this course, you will learn the followings:
  - Algorithm design and complexity analysis, numerical methods and [statistical] computing (from lectures).
  - Programming, problem solving, algorithm design and implementation, debugging (from homework assignments and course project).
- This is NOT a course just about the theory of statistical computing. It is about how to do statistical computing in practice.

Core Competencies

After taking this class, students are expected to be able to

- Understand core numerical and statistical algorithms for data analysis
- Analyze the computational time complexity of statistical algorithms
- Efficiently implement sophisticated statistical methods using C++ and R
- Develop C++ command-line tools and R packages
- Perform simulation studies to compare different methods
- Write scientific reports to introduce developed software

Topics – Part I

Basics of C++, R, Data Structure and Algorithms

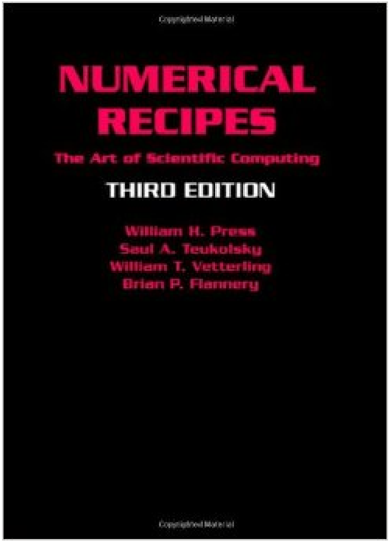
- Introduction to C++ and R
- Key Data Structure
- Computational time complexity
- Basic algorithms (Sorting, Divide and Conquer, Dynamic Programming)
- Interfacing C++ and R languages
- Introduction to parallel computing

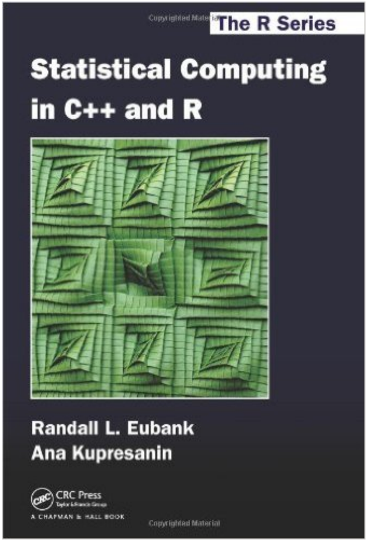
Topics – Part II

Numerical and Statistical Methods

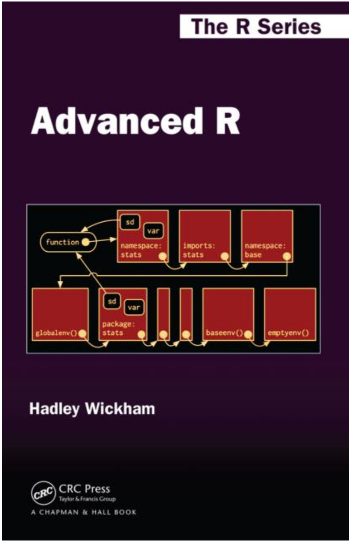
- Numerical methods (Matrix computing, Optimization, Root finding, Numerical integration)
- Monte Carlo methods (Random number generation, Importance sampling, Bootstrap)
- Expectation-Maximization (EM) algorithm
- Markov chain Monte Carlo (MCMC) methods
- Variable selection algorithms

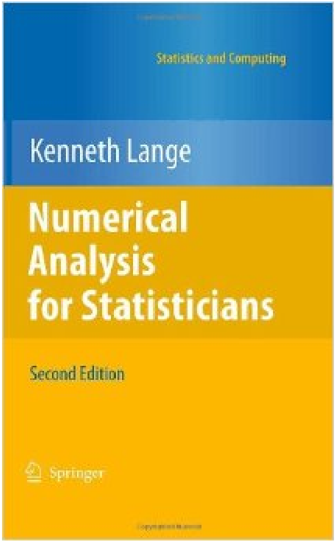
Reference Books (Optional)





Reference Books (Optional)





Grading – Homework: 50%

- Five homework assignments with one for every other week
- **Best four scores will be counted towards the final grade**
- Students can discuss homework questions with others
- Must implement and write up the assignment individually
- Plagiarism will NOT be tolerated
- Late homework will NOT be graded
- Sample homework questions:
  - Write a C++ program to calculate quantiles
  - Write a C++ program for logistic regression
  - Write a C++ program to implement an MCMC algorithm for classifying spatial point patterns
  - Optimize a given R script to compare different methods for variable selection
  - Develop an R package for regression analysis of big data

Grading – Course project: 40%

- Two or three students form a team
- Develop a C++ command-line tool or an R package for cutting-edge statistical methods with a detailed help document for users
- Give an oral presentation to demonstrate the developed software package
- Write a scientific report for introducing the developed software package
- Excellent software and reports are encouraged to be submitted to journals for publication (E.g., Journal of Statistical Software, Bioinformatics Application Notes, The R Journal)

Grading – Quiz: 10%

- Small quizzes will be given during classes
- A total of about 15 pop-up quizzes
- **Best 10 scores will be counted towards to the final grade**
- Multiple choice questions
- Sample quiz question:

  What is the output of the following R code?

```
n = function(x) x/2
o = function(x){
      n  = 10
      n(n)
}
o(10)
```

A 5
B 10
C NA
D Error: could not find function "n"

Grading scale

A+	$[95, 100]$	B−	[40, 49]
A	[85, 94]	C+	[30, 39]
A−	[75, 84]	C	$[20, 29]$
B+	[60, 74]	C−	$[10, 19]$
B	[50, 59]	D	$(0, 9]$
		F	0

Important Dates

- September 15th: Homework 1 distributed
- September 28th: **Drop / Add deadline for full term classes**
- September 29th: Homework 1 due & Homework 2 distributed
- October 13th: Homework 2 due & Homework 3 distributed
- October 20th: **No class for fall break**
- October 27th: Homework 3 due & Homework 4 distributed
- November 3rd: Team members and topics for course project due
- November 10th: Homework 4 due & Homework 5 distributed
- November 24th: Homework 5 due
- November 26th: **No class for Thanksgiving**
- December 1st: Course project oral presentation begins
- December 18th: Course project due

Honor Code

- Honor code is STRONGLY enforced throughout the course.
  - The key principle is that all the code you produce must be on your own. http://www.sph.umich.edu/academics/policies/conduct.html for details.
- You are NOT allowed to share any piece of your homework with your colleagues electronically (e.g. via E-mail or IM), or by a hard copy.
- Discussion between students are encouraged
  - You may discuss homework problems with your colleagues for better understanding and/or brainstorming.
  - You may help your colleague setting up the programming environment necessary for the homework.
  - You may help debugging your colleagues' homework by sharing your trial and errors only up to non-significant fraction of your homework
  - Significant fraction of help can be granted if notified to the instructor, so that the contribution can be reflected in the assessment.

Class Website: Canvas

- Announcements
- Resources
- Gradebook
- Forum
- Course Evaluation
- What else are needed?

Questions?

Why is statistical computing important?

Now is "Big Data" era

- ✓ Next-generation sequencing studies often become >100TB in size
- ✓ The amount of fMRI data for 1,500 subjects can be up to 5.4TB
- ✓ Google Maps has over 20 PB (= 20,000 TB) of data
- ✓ Computation of simple statistics (e.g. mean) will take a long time.

Efficiency affects the feasibility of statistical methods

- ✓ In the analysis of big data, it is not uncommon that more accurate methods takes much longer time than less accurate approximation (e.g. 40 years vs 1 day).
- ✓ Implementation with low-level languages such as C++ has more room for speed improvements than higher level languages such as R
- ✓ Even with the same language, different algorithms can result in substantial gain in speed without losing accuracy.

A Simple Example in R

To generate a sequence of $x = [1^2, 2^2, \ldots, 10000^2]$

```
> system.time({x=c(); for (i in seq(100000)) x=c(x,i^2)})

   user  system elapsed
 13.585  17.570  31.199

> system.time({x=seq(100000); for (i in 1:length(x)) x[i]=x[i]^2})

   user  system elapsed
  0.108   0.003   0.112

> system.time({x=seq(100000)^2})

   user  system elapsed
  0.001   0.000   0.000
```

Computer representation of numbers

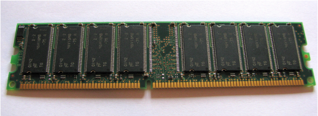
For C++ *primitive data types* include

- *boolean*: for variables that take on the two values true and false
- *integer*: which translates into specific types such as short, unsigned short, int, long int and unsigned long int that provide different amounts of storage for integer values
- *character*: indicated by the modifier char in source code, that is used for variables that have character values such as "A" and "x"
- *floating point*: which encompasses basically all non-integer numbers with the three storage types float, double and long double.

Transistors

- Computer operations must work with representations of numbers that are accomplished by transistors

Central Processing Units (CPU)



Random Access Memory (RAM)

- Two possible states off and on
- Number of transistors is finite, although can be up to billions
  - How and How much information can actually stored
- To manage overall memory effectively,
  - Restrict the amount of memory that can be allocated to different kinds of numbers
  - There are limits on the range of different kinds of numbers

Decimal system

- In the decimal system, numerical values are represented in units or powers of 10
- For a nonnegative integer k,

$$k = \sum_{j=0}^{m} a_j (10)^j, \qquad k = (a_m a_{m-1} \dots a_0)_{10}$$

where $a_j \in \{0, 1, \dots, 9\}$ for $j \geq 0$ and $a_m \neq 0$ if $m \geq 1$.

- For example,

$$2015 = 2 * (10)^3 + 0 * (10)^2 + 1 * (10)^1 + 5 * (10)^0$$

- Then

$$2015 = (2015)_{10}$$

Binary representation of integer

- Since transistors have only two states it is not surprising that the base of choice for computer arithmetic is binary or base 2. The *binary expansion* of k is given by

$$k = \sum_{j=0}^{m} b_j (2)^j, \qquad k = (b_m b_{m-1} \ldots b_0)_2$$

  where $b_j \in \{0, 1\}$ for $j \geq 0$ and $b_m \neq 0$ if $m \geq 1$.

- For example,

$$2015 = 1*(2)^{10} + 1*(2)^9 + 1*(2)^8 + 1*(2)^7 + 1*(2)^6 + 0*(2)^5 \\ +1*(2)^4 + 1*(2)^3 + 1*(2)^2 + 1*(2)^1 + 1*(2)^0$$

- Thus,

$$2015 = (11111011111)_2$$

Bit and Byte

- Bit: the basic unit of information in computing and digital communications
- Byte: a unit of digital information that most commonly consists of eight bits.
- The connection between machine memory and computer arithmetic:

$$\begin{array}{ccc} \text{Bits} & \leftrightarrow & \text{Transistors} \\ \text{Bytes} & \leftrightarrow & \text{Block of 8 Transistors} \end{array}$$

Represent a number in memory

Its binary representation is physically created by

- allocating it a block of memory, e.g., a group of contiguous transistors
- identifying the individual transistors in the block with a power of 2 from its binary representation
- turning on those transistors that correspond to powers of 2 that have unit (1) coefficients

In the example,

- We need 11 transistors to hold integer $2015 = (11111011111)_2$.
- 11 bits are needed, so 1 byte is not enough, at least 2 bytes

More on bytes

- A document, an image, a movie .. how many bytes?
- 1 byte is enough to hold 1 typed letter, e.g. "b" or "X"
- All measured in bytes, despite being very different hardware
  - Kilobyte, KB, $=$ 1 thousand bytes $= 10^3$ bytes
  - Megabyte, MB, $=$ 1 million bytes $= 10^6$ bytes
  - Gigabyte, GB, $=$ 1 billion bytes $= 10^9$ bytes
  - Terabyte, TB, $=$ 1 trillion bytes $= 10^{12}$ bytes
  - Petabyte, PB, $=$ 1 quadrillion bytes $= 10^{15}$ bytes

Summary

- ✓ Syllabus
- ✓ Introduction to statistical computing
- ✓ Introduction to computer representation of data
  - ✓ Binary representation
  - ✓ Storage unit (Bit, Byte, KB, MB, GB, TB, PB)
- ✓ Class survey