Fall 2020

Biostat 213
Programming *for* Data Science *in* R

Efstathios (Stathis) D Gennatas, MBBS AICSM PhD

Assistant Professor
Div. Bioinformatics
Dept. of Epidemiology and Biostatistics
University of California, San Francisco

efstathios.gennatas@ucsf.edu
github.com/egenn

UCSF

Outline

- Welcome to Biostat 213 PDSR!
  - 1-minute **introductions**
- Course **goals** and **structure**
- **Data Science in biomedicine**: goals and challenges
- The **R language** and why it's great for biomedical data
- The **R package ecosystem**: CRAN, Bioconductor, GitHub
- The **RStudio** Integrated Development Environment (IDE)
- **Rmarkdown**, RStudio notebooks, Jupyter notebooks
- Introduction to class **projects**

Introductions!

Research interests

Experience with programming and R

Ideas, concerns, expectations

Biostat 213 Goals & Structure

Course goals

- Understand basic programming concepts and how they are implemented in R
  - These are essential for anything you want to do in R in the future: epi, bio, ML
- Learn methods for data importing / exporting / cleaning / transformation / visualization
  - These are essential for any data project
- Reach a level of confidence with R where you can quickly learn to use a new package / perform a new task
  - Using builtin & package documentation and, of course, the internet / web search

Class structure

Each class:

- Discussion to address issues from previous week's material and assignments
- Interactive lecture - demonstration
- 10 minute break
- Interactive lecture - demonstration / lab
- Q&A

Class structure

- Do ***ask questions / interrupt*** if anything big or small is unclear - or too obvious
- Goal is to learn as much as possible in class

Coursework & grading

60% Weekly assignments

- Attempt every question!

40% final project:

- **Proposal** (PDF) - *due 10/26/2020*
- **Midway report** (PDF). - *due 11/16/2020*
- 2-min poster-style **presentation** & **completed report** (PDF & HTML) - *due final Class 12/07/2020*

Data Science in Biomedicine

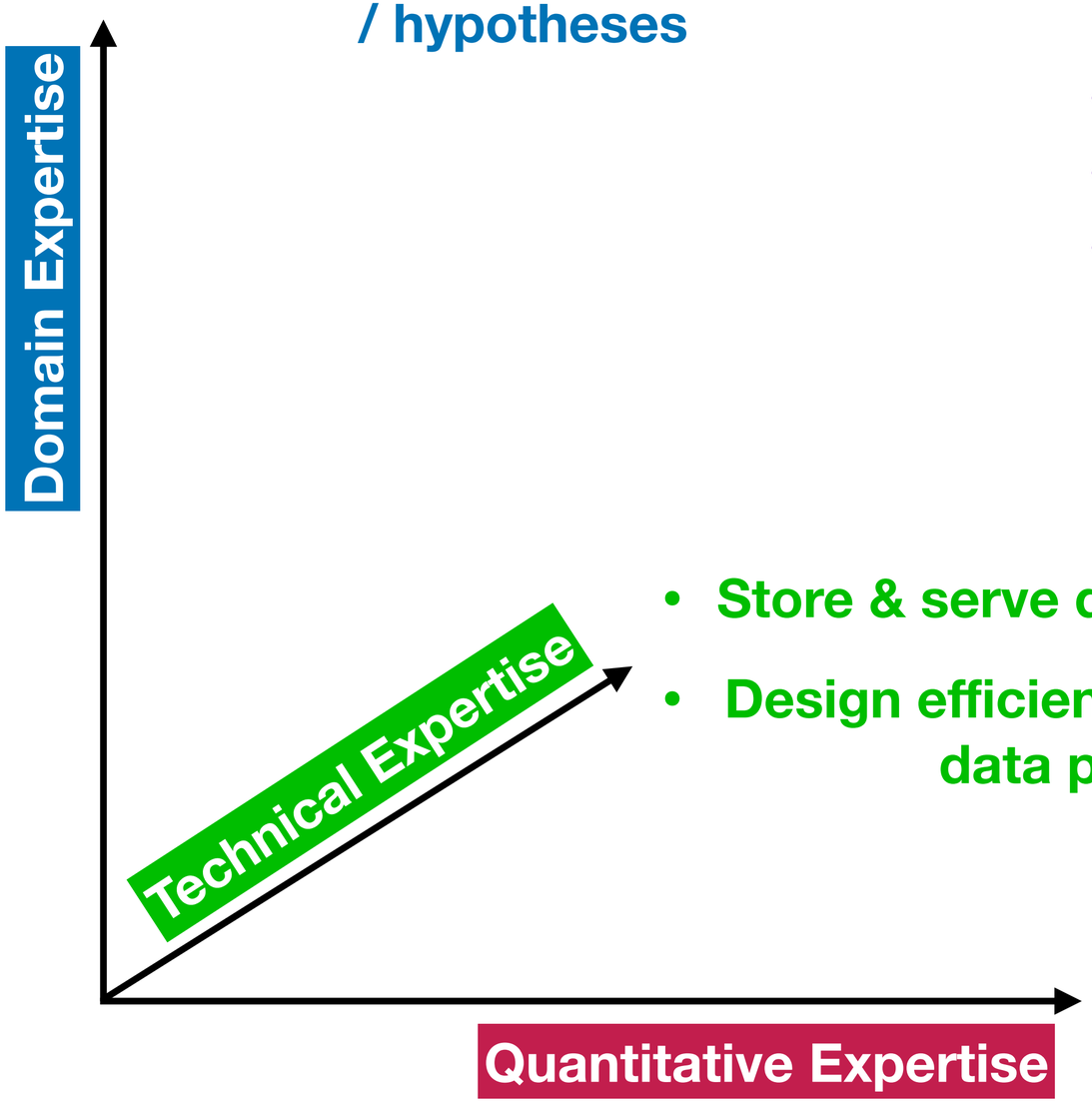
Data Science extracts insights from data

- Inherently interdisciplinary field
- Related to Statistics, Machine Learning, Data Mining, Data Analysis
- Focus is often on deriving **actionable insights**
- Estimates suggest 60-80+% of data science is
  data preparation, cleaning, organizing

Statistics / Machine Learning / Data Science

- Define research questions / hypotheses

- Design experiments & data collection
- Preprocess data
- Select algorithms
- Evaluate results



- Store & serve data securely & efficiently
- Design efficient & reproducible data pipelines

Coding for science

- You may or may not enjoy coding
- The better you get at it, the more you'll enjoy it

  but

- The focus is on doing your science correctly and efficiently
- Technical hurdles should not get in the way

Data challenges in Biomedicine

- Small sample sizes are very common, big data is less common but this is changing
- Rare medical cases regardless of sample size
- High noise
  - In both features and outcomes
  - Measurement error vs inherent, biological stochasticity
- Missing data
- Hidden confounders
- Lack of controls
- Bias in data

The R language

The R language

- **S** statistical programming language was developed in 1976 at Bell Labs by John Chambers and others "to turn ideas into software, quickly and faithfully”
- **R** is an implementation of S; developed by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand; initial version released in 1995
- Supported by the **R Foundation for Statistical Computing**, developed by the **R Development Core Team**
- Official part of the Free Software Foundation’s GNU project available under GNU GPL v2
- Current version 4.0.2 released 2020-06-22

The programming language known as R

- Written in **C**, **Fortran**, and **R**
- **High-level**
- **Interpreted**
- Mostly **functional**
- **Object-oriented programming** also available
- **Domain-specific language (DSL)** for statistical computing

Function to add two numbers

Assembly

C — vi add.asm — 45×23

```
.model small
.data
 opr1 dw 1234h
 opr2 dw 0002h
 result dw 01 dup(?),\'$\'
.code
        mov ax,@data
        mov ds,ax
        mov ax,opr1
        mov bx,opr2
        clc
        add ax,bx
        mov di,offset result
        mov [di], ax

        mov ah,09h
        mov dx,offset result
        int 21h

        mov ah,4ch
        int 21h
        end
                        1,1           All
```

low level

C

C — vi add.c — 45×23

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    double a,b,sum;

    a = atof(argv[1]);
    b = atof(argv[2]);
    sum = a+b;

    printf("%lf\n", sum);

    return 0;
}
"add.c" 15L, 186C          1,1           All
```

high level

R

C — vi add.R — 45×23

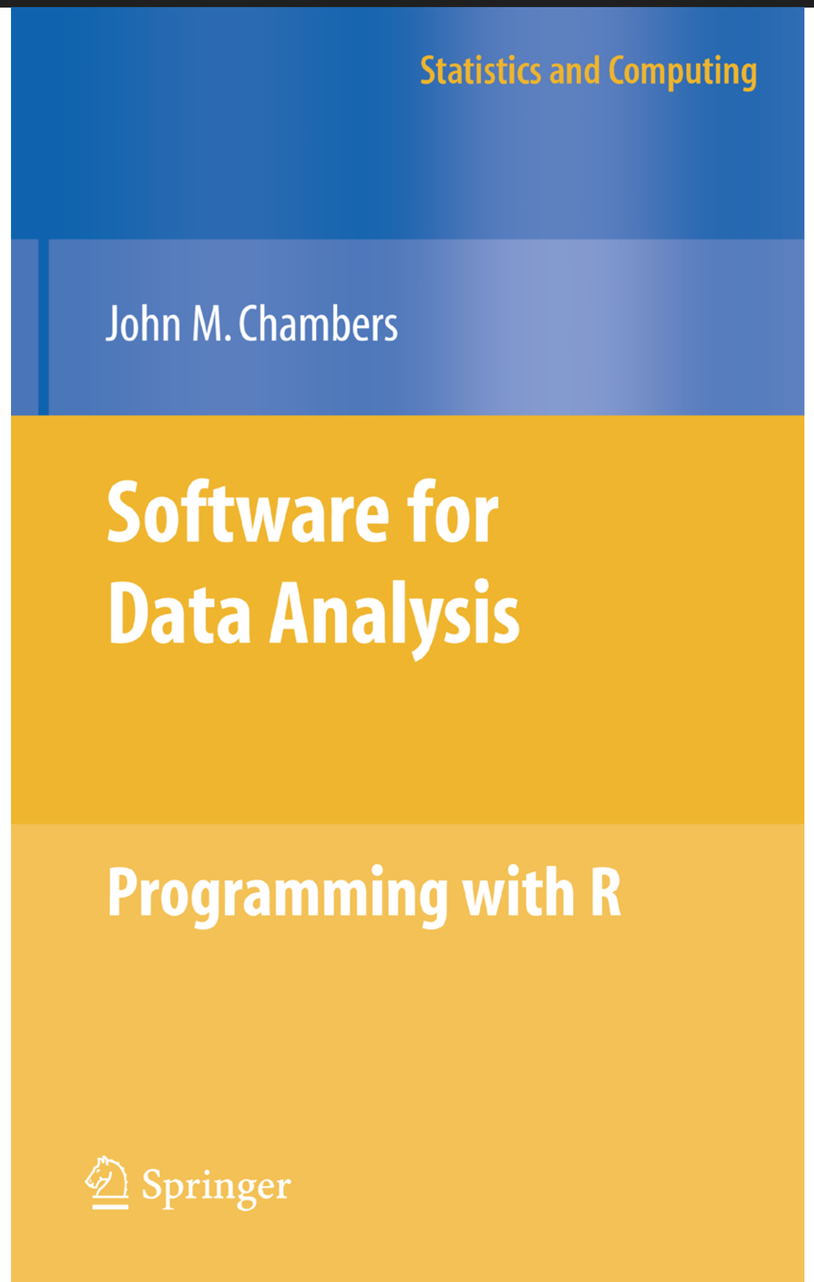
```
add ← function(x, y) {
    x + y
}

"add.R" 4L, 37C            1,1           All
```

higher level

The R language

- *“Everything that exists in R is an object”*
- *“Everything that happens in R is a function call”*
- *“Interfaces to other software are part of R"*

The R language



1.1 Exploration: The Mission

The first principle I propose is that our *Mission*, as users and creators of software for data analysis, is to enable the best and most thorough exploration of data possible. That means that users of the software must be ale to ask the meaningful questions about their applications, quickly and flexibly.

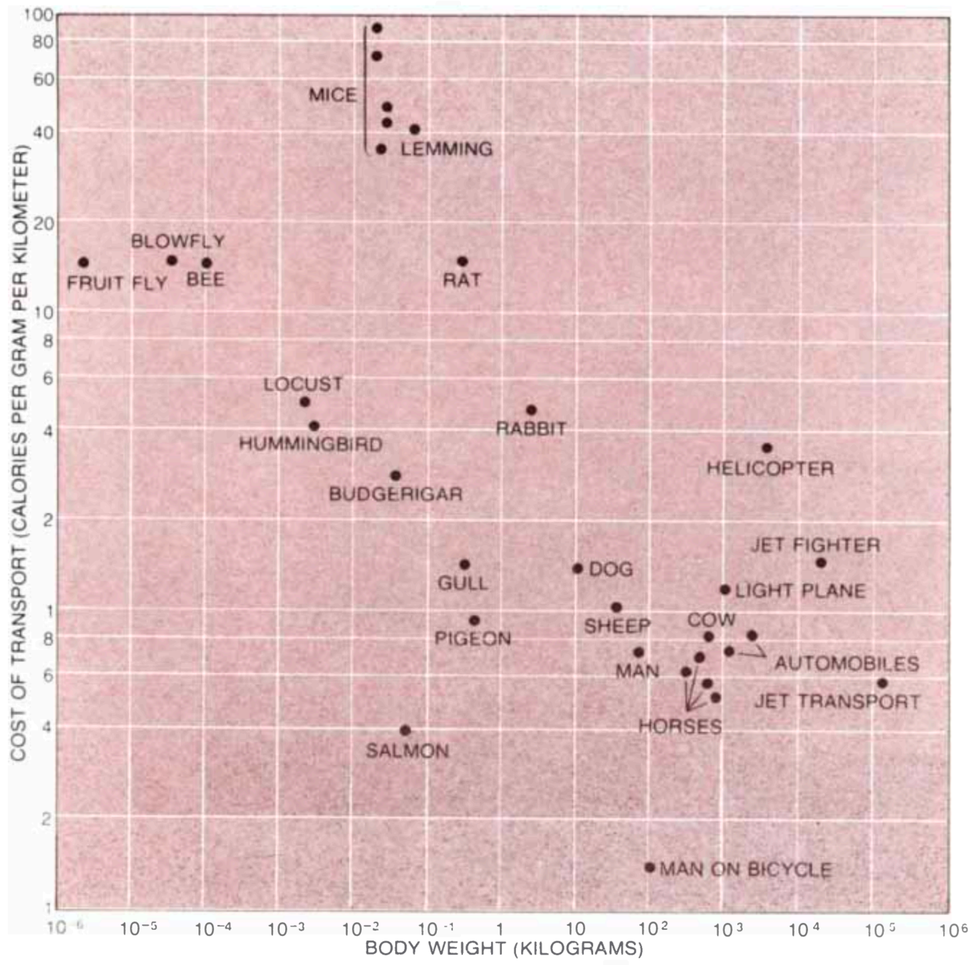
1.2 Trustworthy Software: The Prime Directive

Exploration is our mission; we and those who use our software want to find new paths to understand the data and the underlying processes. The mission is, indeed, to boldly go where no one has gone before. But, we need boldness to be balanced by our responsibility. We have a responsibility for the results of data analysis that provides a key compensating principle.

The R language

- What is a programming language for?
- Who is a programming language for?
- Programmers vs. users, scientists, etc.
- R does a great deal behind the scenes to help you code faster and enjoy an interactive session
- Speed - user friendliness tradeoff in some cases, but by understanding the language you can avoid most penalties

Programming and efficiency



MAN ON A BICYCLE ranks first in efficiency among traveling animals and machines in terms of energy consumed in moving a certain distance as a function of body weight. The rate of energy consumption for a bicyclist (about .15 calorie per gram per kilometer) is approximately a fifth of that for an unaided walking man (about .75 calorie per gram per kilometer). With the exception of the black point representing the bicyclist (***lower right***), this graph is based on data originally compiled by Vance A. Tucker of Duke University.

A computer is a “bicycle for our minds” - Steve Jobs

Programming is not just about using computers to do tasks that are hard to do otherwise,
it is a tool to increase efficiency by many orders of magnitude.

The R package ecosystem

The R package ecosystem

- R boasts extensive quantitative and statistical functionality in the base system
- Extended through a vast ecosystem of packages
  - **CRAN**: The Comprehensive R Archive Network
    https://cran.r-project.org/
  - **Bioconductor**: Bioinformatics-related packages and more
    https://www.bioconductor.org/
  - **GitHub**

The Comprehensive R Archive Network (CRAN)

- Long list of FTP and web servers (aka mirrors) around the world that store copies of the R base system, documentation and packages
- 16277 packages as of 2020-090-14
- CRAN Task Views provide curated lists of packages organized by field/use case: https://cran.r-project.org/web/views/
- From the R console use **`install.packages()`**

Bioconductor

- Provides tools for the analysis and comprehension of high-throughput genomic data
- Current version 3.11 includes 1903 packages
- From the R console install BiocManager first from CRAN:
  `install.packages("BiocManager")`
  then install packages using:
  `BiocManager::install()`

GitHub

- Most popular host of open source software projects
- Public and private repositories
- Host projects in any programming language
- Install R packages directly using remotes (or devtools):
  `remotes::install_github("account/repository")`
- Many R packages and most new ones available on GitHub
  - "developer" versions updated at any schedule
- All of CRAN available as readonly

R for Data Science

- Base R contains everything you need to do data science / statistics / machine learning
- There are vast amounts of resources online
  - Official documentation (quality variable; trustworthy)
  - Blogs (quality variable; trustworthiness highly variable)
  - Stack overflow - https://stackoverflow.com/questions/tagged/r (good quality; trustworthy)
- Trust no one - even how pleasant

Demo: R console

RStudio

RStudio

- The most advanced, full-featured Integrated Development Environment (IDE) for R
- Source code editor, R console, documentation viewer, graphics viewer, HTML viewer, R package management, build automation, git integration, custom add-ins support
- Desktop, Server, Cloud versions available
- Makes it easier, more efficient and more fun to work with data

Demo: RStudio