The Georgia R School: Course Description

Introduction to R Programming

Level: Introductory; No experience or knowledge of R is assumed.

Course Duration: *Introduction to R Programming* is a 10-week course that meets 2 ¾ hours one day per week for 10 non-consecutive weeks. 'After-class' self-paced exercises are provided weekly (with solutions the following week). All live classes are recorded in real-time and the recordings are provided to all participants immediately after each class is concluded.

Prerequisites: None.

Description: *Introduction to R Programming* is a gentle, yet comprehensive introduction to the practice of general-purpose application development in the R environment. Participants may already be skilled programmers (in other languages) or they may be complete novices to R programming or to programming in general, but their common objective is to write R applications for diverse domains and purposes. No statistical knowledge is necessary. This course is a thorough introduction to using the R environment and language for general-purpose application development. The course covers programming-related skills that are not specific to any one domain, for example, statistical techniques or data mining, but rather R programming skills that apply to any application domain. It is a "hands-on" course that makes use of many extended examples of the development of R programs for different purposes. RStudio, a popular, open source Integrated Development Environment (IDE) for developing R applications, is utilized in the course, supplemented with R-based direct scripts (e.g. 'command-line prompts') when necessary.

The materials for *Introduction to R Programming* derive from two main sources:
(1) Norman Matloff's acclaimed book *The Art of R Programming: A Tour of Statistical Software Design* (2011, No Starch Press; $24 on Amazon.com); and
(2) Comprehensive R Archive Network (CRAN) R programming documentation: (a) *An Introduction to R*; (b) *R Language Definition*; and (c) *Writing R Extensions*, each developed by the R Core Development Team. CRAN documentation is freely available on the Internet.

Course participants are not required to purchase Matloff's book, although it is recommended as a landmark publication covering the practice of general-purpose programming in R. The book is unique in that it is not influenced by any one domain (for example, forestry data) or technique (for example, multivariate analysis; Bayesian statistics; or monte carlo simulation). Read what Revolution Analytics has to say about Matloff's book: "if a person really wants to be able to speak the R language and become a competent R programmer then, at the present time, one can find no better guide than Norman Matloff's The Art of R Programming. Professor Matloff is a statistician and a computer scientist with a considerable amount of teaching experience. His book is no mere programming reference guide. It is a carefully crafted sequence of lessons that start at the beginning and work up to some fairly advanced topics including a lucid account of object-oriented programming in R, a discussion of R programming for the internet, examples of parallel programming with R, and a discussion spanning several chapters of how to write production-level R code that includes methods and advice on debugging R code, writing efficient R code, and interfacing R with other languages. Other distinguishing features of the book are brief examples showcasing a large number of functions (including rare gems such as D() for symbolic differentiation) that indicate the power and scope of R, and over thirty "Extended Examples" each of which is a credible study in writing careful, professional code."

Course Summary: The course demonstrates how to structure and write programs using R statistical software. The R programming is taught using dozens of extended examples of R programs for different purposes (please see detailed schedule below). The course presents a comprehensive training platform for learning to program in R. RStudio, an open source Integrated Development Environment (IDE) for developing R applications, is utilized, supplemented with R-based direct scripts (e.g. 'command-line prompts') when necessary.

Course Outline: *Introduction to R Programming I* topics are listed below. Topics are illustrated using appropriate R code ('scripts') and accompanying data sets. All data sets, scripts and sample programs are provided with the class materials. All topics and programming techniques are demonstrated 'live' in class. There are weekly 'between-class' exercises (and later, the solutions) provided. All software, all scripts demonstrated in class, appropriate explanatory documentation, and all data sets are provided along with all course materials, and in advance of each class session. The course materials include all class slides, any additional data or material referenced in class by the instructor, as well as complete, permanent, digital, high fidelity, audio and video recordings of all of the live class sessions. The permanent recordings, and all class materials, are retained by the participants, as part of their registration fee, when the course is completed. Finally, a personalized 'course participation certificate' from the non-profit Georgia R School is provided to each named participant who successfully completes the course. All major topics discussed in the course are itemized on the course completion certificate.

The course outline follows:

Introduction to R Programming

DAY 1: The first "live" class is comprised of two parts: (1) an introductory session which serves to introduce the participants to: (a) the electronic classroom; (b) R software; (c) the instructor and each other; and (d) the course content, approach, and agenda: and (2) a regular 'lesson', entitled "getting started" which focuses on the material in the first chapter of *The Art of R Programming*:

DAY 1: A First R Session

- **Example:** Running an R program in batch mode

Introduction to Functions

- **Example:** Counting the odd numbers in a vector of integers

Important R Data Structures

- **Extended Example:** Regression analysis of exam grades.

Startup and Shutdown

Getting Help

DAY 2: Vectors, Matrices and Arrays

Scalars and Vectors

Declarations and Recycling

Common Vector Operations

Using all() and any()

- **Extended Example:** Finding runs of consecutive 1’s
- **Extended Example:** Predicting discrete-valued time series

Vectorized Operations

Vectorized if-then-else

- **Extended Example:** A measure of association
- **Extended Example:** Recoding an abalone data set

Matrices and Arrays

Creating Matrices

General Matrix Operations

- **Extended Example:** Generating a covariance matrix

Applying Functions to Matrix Rows and Columns

- **Extended Example:** Finding outliers

Adding and Deleting Matrix Rows and Columns

- **Extended Example:** Finding the closest pair of vertices in a graph

DAY 3: Lists and Data Frames

Creating Lists; General List Operations

- **Extended Example:** Text concordance

Accessing List Components and Values

Applying Functions to Lists

- **Extended Example:** More text concordance
- **Extended Example:** Back to the abalone data

Recursive Lists

Creating Data Frames

- **Extended Example:** Regression analysis of exam grades continued

Typical Data Frame Operations

- **Extended Example:** A salary study

Merging Data Frames

- **Extended Example:** An employee database

Applying Functions to Data Frames

- **Extended Example:** Applying logistic regression models
- **Extended Example:** Aids for learning Chinese dialects

DAY 4: Factors and Tables; Programming Structures

- Factors and Levels
- Common Functions used with Factors
- Working with Tables
  - **Extended Example:** Extracting a subtable
  - **Extended Example:** Finding the largest cells in a table
- More Factor and Table-Related Functions
- Programming Structures
- Control Statements
- Arithmetic, Boolean Operators and Values; Default Argument and Return Values
- Environment and Scope Issues
  - **Extended Example:** A function to display contents of call frame

DAY 5: More Programming Structures

- Writing Upstairs
  - Writing to nonlocals with Superassignment operator and assign() function
  - **Extended Example:** Discrete-event simulation in R
  - When to use global variables
- Recursion
  - Quicksort implementation
  - **Extended Example:** A binary search tree
- Replacement Functions
  - **Extended Example:** A self-bookkeeping vector class
- Text Editors and Integrated Development Environments (IDEs)
- Writing Binary Operations and Anonymous Functions

DAY 6: Doing Math and Performing Simulations in R

- Math Functions
  - **Extended Example:** Calculating a probability
  - Cumulative sums and products
  - Minima and Maxima
  - Calculus
- Functions for Statistical Distributions
- Sorting
- Linear Algebra Operations
  - **Extended Example**: Vector cross product
  - **Extended Example:** Finding stationary distributions of Markov chains
- Set Operations
- Simulation Programming in R
  - Random variates generators
  - **Extended Example:** A combinatorial simulation

DAY 7: Object-Oriented Programming; S3 and S4 Classes

- S3 Classes
  - S3 generic functions

Example: OOP in lm()
Writing S3 classes
Inheritance
Extended Example: A class for storing upper-triangular matrices
Extended Example: A procedure for polynomial regression
S4 Classes
Writing S4 classes
Implementing a generic function of an S4 class
S3 versus S4
Managing Objects

DAY 8: Input and Output; String Manipulation
Accessing Keyboard and Monitor
Reading and Writing Files
Extended Example: Reading PUMS census files
Extended Example: Summing the contents of many files
Accessing the Internet
Extended Example: Implementing parallel R
Overview of String Manipulation Functions
Regular Expressions
Extended Example: Testing a filename for a given suffix
Extended Example: Forming filenames
Use of String Utilities in edtdbg Debugging Tool

DAY 9: Graphics and Debugging
Creating and Customizing Graphics
Extended Example: Two density estimates on same graph
Extended Example: More of polynomial regression example
Extended Example: Magnifying a portion of the curve
Three-Dimensional Plots
Saving Graphs to Files
Fundamental Principles of Debugging
Ensuring Consistency in Debugging Simulation Code
Debugging Tools; R Debugging Facilities
Extended Examples: Two full debugging sessions

DAY 10: Speed and Accuracy Performance Enhancements
Writing Fast R Code
The Dreaded For Loop
Extended Example: Achieving more speed conducting monte carlo simulations
Functional Programming and Memory Issues
Extended Example: Avoiding memory copy
Using rprof() function to find ‘slow spots’ in code
Chunking
Fitting data into memory; R packages for memory management
Course Wrap-Up