PSYC798W: R Programming for Behavioral Sciences (Winter 2018)

Basic info

Instructor: Dr. Scott Jackson

Email: scottrj@umd.edu

Office phone: 301-226-8881

Class location: BPS 1236

Time: M-F, 9:30-12:30

Office hours: 12:30-1:30, or by appointment

Office hours location: TBD

Course material repository: `https://github.com/scottrjackson/r_programming_bsos_winter2018`

Description

R is a programming language and environment designed for statistical analysis. It is free and open-source, and it includes integration with thousands of cutting-edge packages contributed by users from around the world. It has become a *de facto* standard and *lingua franca* for statistical analysis. It is an incredibly powerful tool for data analysis and visualization, and thus an indispensable tool for any kind of quantitative work in the behavioral and social sciences. However, because many (if not most) students and researchers in these fields are not otherwise trained in programming techniques, learning R can be difficult, and poor understanding of programming concepts and techniques can create problems in analysis and reporting of results.

`www.r-project.org`

This course aims to give you a foundation in programming, in order to facilitate future work with R. It is not a stats course, though we will address a few topics critical to data analysis such as cleaning, formatting, and visualizing your data. The focus of the course is building concepts, skills, and habits to make you a better programmer, in order to get the most out of R.

Goals

This course is organized around two sets of themes. First, there is the underlying argument for why a programming language like R is a superior alternative to more limited graphical-interface programs

like SPSS or spreadsheet programs like Excel. You can think of the advantages of R as superpowers for doing research. There are (at least) three of them:

Reproducibility You can go back to see **exactly** what you did, and you can do it again, with virtually no effort. This also helps make your code work better, and easier to improve upon.

Repetition You can take something you did, and do it hundreds, thousands, or *millions* of times (literally), with very little effort (though maybe some patience). This opens up new avenues like simulations or other kinds of industrial-strength analysis.

Re-usable code You can take code you write and apply it to new situations. This makes what you do more useful, and more share-able. Share-ability leads to fame and fortune.

In this course, you will learn the basics of each of these superpowers. You will be able to:

- work in a more reproducible way using scripts, notebooks, and version control.
- repeat analysis in loops and other structures to be able to perform large-scale repetition and simulation.
- write functions to make your code more re-usable and generally applicable, for yourself and others.

In order to acquire these superpowers, there are three types of topics we will cover:

Core R language: You will learn the basics of the R language and be able to work with a variety of data types, functions, and computations.

Tools: You will learn to use one or more interfaces to R (i.e., ways to run R), you will have an opportunity to learn and practice workflows for data analysis, and you will learn the basics of version control software (`git`).

Techniques: There are lots of ways to do things in R, and you will learn useful tips and best practices for structuring code, documenting code, adapting code, and finding help when you need it.

In summary, this course will help you approach your work with R more like a programmer, which will enable you to make the most of the superpowers that R provides access to.

Structure of class time

Each class is 3 hours. This time will be divided roughly into four parts:

1. **Review:** We will go over the previous day's homework and answer any outstanding questions.
2. **Lecture:** I'll present some concepts, with slides and/or demos.
3. **Practice:** You will have some in-class exercises to try, in order to immediately apply the concepts from the lecture.
4. **Follow-up and troubleshooting:** Things will inevitably go wrong. You are very unlikely to perfectly apply the concepts the first time during practice, and I will probably forget to mention some things along the way. So we will reserve the last hour to talk about the problems you encounter in practice and solutions to those problems. If everything goes unexpectedly smoothly, we will use this time to cover more advanced extensions of the concepts.

Schedule

The following schedule is a work in progress. Check the online version of this syllabus for the most up-to-date schedule, in case you need to miss a class but are interested in specific topics. In particular, the "special topic" and "review" days may be used as "spillover" days if we need to move more slowly or repeat any of the other topics, and if we have time for special topics, those may change based on student interest.

Date	Topic
Jan 2	Getting started; installation; workflow overview
Jan 3	Basics #1: objects, functions, packages, and the environment
Jan 4	Basics #2: working with different kinds of data
Jan 5	Basics #3: working with complex objects and messy data
Jan 8	Basics #4: graphics
Jan 9	Review: more graphics or other review
Jan 10	Iteration #1: loops and control
Jan 11	Iteration #2: vectorization
Jan 12	Writing functions
Jan 15	MLK holiday
Jan 16	Review, special topics (TBD)

Grading

Overall

There are three main components that determine your grade:

1. In-class Practices
2. Out-of-class Homework
3. The Final Project

The overall course grading is based on how much of these components you complete on time:

A = Completed all three components

B = Completed two components

C = Completed one component

D = Did some work but did not complete any components

F = Left out one or more components entirely

What it means to "complete" each of these is described below.

Practices/attendance

Required to "complete": Seven in-class Practices

As described above, there will be one or more Practice exercises in each class. In order to get "credit" for a Practice, you need to make a full attempt. That is, you must complete some code or other activity that represents each step in the Practice, as described in the assignment. This code does not necessarily have to work! Practices represent an opportunity to try things and break things, and learn through challenges and even failures. But an honest attempt is required. I may require Practices to be submitted electronically during class, as needed.

Requiring Practices is the method I will use to require attendance, since these can only be completed during class. You will not receive credit for Practice assignments completed outside of class. There is no separate grade or tracking of attendance. There is no make-up for a Practice. The fact that you are only required to complete 7 Practices builds in flexibility if you need to miss one or more classes for any reason.

Homework

Required to "complete": Seven out-of-class Homework

Homework will be assigned with each class. These will be submitted to me electronically before the next class. The method of submission

will be detailed in the assignment. The assignments will represent some extension or variation on the lesson and Practice of that day. In order to complete a Homework, you must actually complete the objective for the assignment.

While the Homework objectives will be the same for all students, the data, and therefore the exact solutions, will be different for each student. The first assignment is to find a good data set to use throughout the class. The subsequent assignments will have you explore this data set and practice the things we learn in class. The purpose of this is to allow and encourage collaboration. Since everyone's "answers" will be slightly different, each student will need to adapt things for their particular data set. This closely mimics a common way to learn outside of class, namely cribbing off of other people's code, and thus is designed to set you up to continue to learn on your own.

This data set is also intended to be the data used in the Final Project (described below), and thus the Homework assignments may help you biuld towards that project.

Exercises are due before the beginning of the next class period, and this will be enforced strictly. As with Practices, the requirement to correctly complete only 7 Homework assignments builds in flexibility. But it is highly recommended to try to complete every Homework, since these will help you in your final project.

Final Project

The final part of the course that determines your grade is the Final Project. Here are the requirements and dates:

Required to pass the course: Submit proposal by Jan 7

Submit a proposal: You need to send me a brief written proposal (via email) for your project. This needs to outline how it will address the requirements (data, analysis, etc.). The deadline for this is **11:59 PM EST, Sunday, January 7**. If you do not send a proposal by this deadline, you will not count as "completing" the Final Project.

Required to "complete": Submit revised proposal (due date determined when I send you feedback)

Revise proposal: I will look at your proposal and either approve it or send it back to you with suggestions. If I send it back, I may require a revised proposal. I will set the deadline for this revision when I send it back to you.

Complete the project: The project is a set of code completing some kind of analysis. The requirements are:

1. Pick some data to work with
2. Perform some kind of analysis, which may result in one or more of:
   (a) Numerical results
   (b) A complex object (like a regression analysis)
   (c) Graphical results (like a plot)
3. Report the analysis with appropriate documentation
4. Some aspect of the above (data/analysis/reporting) needs to be "non-trivial," i.e., something we have not explicitly covered in class. Examples include:
   - Data: especially messy/big/complex data
   - Programming-dependent analysis: doing something that would be difficult/infeasible/impossible/annoying with a simple "point-and-click" program interface
   - Results: tricky visualization, novel way of reporting results
   - Code: providing useful new function that would be of interest to other people

   Example: analyze a complex public data set, with code that would be helpful to other people in wrangling this data set to work with on their own

   Example: a simulation-based analysis requiring loops/iteration

   Example: a cool new way to visually represent/explore data in your field

   Example: a convenience function that people in your lab could use in analyzing data that they commonly work with
5. Post results and replicable code via GitHub (preferred), or email a complete zipped repository to me. This will be described fully in class. The requirement means that your code should compile/run/complete fully. If it does, and it does what you said in your proposal, then the project will be complete. You will be able to verify that it works ahead of time, so there should be no uncertainty in whether the project "passes" or not. The due date for the project is **8:00 PM EST, Tuesday, January 23**. Projects posted after this time will not be evaluated.

   Required to pass the course: Email or post project via GitHub by Jan 23

A more complete set of instructions, suggestions, etc. will be made available during the first week of the course.

Optional reading

All of the course materials will be provided by me, free. However, because R is so popular and widely used, there are many other excellent resources. Below is a selection of "most recommended" things to look at, if you want additional information, or if you need a good reference book.

Official R docs and manuals

1. See here: `http://cran.r-project.org/manuals.html`

Recommended general-purpose books

1. *R in Action* (Kabacoff): "practical" intro with some more advanced topics, including examples of some common stats analysis
2. *R in a Nutshell* (Adler): a good overall reference book
3. *R Cookbook* (Teetor): another general reference, with a lot of "recipes" for doing different things
4. *The Art of R Programming* (Matloff): a very readable, insightful book from a more programming perspective, good for getting a better handle on the "guts" of R
5. *Advanced R* (Wickham): a great resource for digging deeper into understand programming in R, by one of the gods of R. There is a book, but also a website here: `http://adv-r.had.co.nz/`
6. *Software for Data Analysis* (Chambers): big treatise on How R Works, from both a conceptual and technical level. Good for really deepening your understanding of programming for data analysis.
7. *The R Inferno* (Burns): a tongue-in-cheek look at some of the traps and pitfalls of working with R (and how to avoid them)

More specialized stats books

1. *Discovering Statistics with R* (Field): nice general-purpose stats reference, written in a very light-hearted style, with tons of R examples
2. *An R Companion to Applied Regression* (Fox): excellent textbook on regression, with lots of useful R code, and an accompanying package (`car`) with lots of useful functions
3. *Data Analysis Using Regression and Multilevel/Hierarchical Models* (Gelman & Hill): should be required reading if you are interested in using mixed-effects models (aka multilevel/hierarchical models)
4. *Doing Bayesian Data Analysis* (Kruschke): very accessible intro to Bayesian analysis, with tons of R code
5. *Elements of Statistical Learning* (Hastie, Tibshirani, Friedman): a seminal text on the topic, with R code from the experts

6. *Regression Modeling Strategies* (Harris): a good book for digging deeper into regression models. Harris's `rms` and `Hmisc` packages are also very widely used

Handy websites

1. Quick-R: a lot of good tips and quick reference, by the author of *R in Action* (Kabacoff).
2. Cookbook for R: lots of handy stuff.
3. R Task Views: a great way to find useful packages.
4. StackOverflow: a great source for asking questions and getting answers. Google searches on error messages often lead here.
5. "The Google": when in doubt...

Policies and other info

Honor Code

You will be expected to abide by the student honor code. The exercises are be designed such that comparing notes with other students is allowed and even encouraged. However, you still need to do your own work. For any assignment you submit, you will be held to the honor code. If you have any questions at all, please ask me before it becomes a problem.

Accommodations

Please let me know about any requested accommodations due to disabilities as soon as possible, and we will come up with a plan.

Inclement weather

If the weather gets nasty, check the UMD website and/or phone line:

- `http://prepare.umd.edu/`
- 301-405-SNOW