Syllabus

ESM 296-3W - Environmental Informatics - Winter 2016

Ben Best, Naomi Tague

2015-11-23

Contents

Overview ................................................................. 1
Instructors .............................................................. 1
Workload ............................................................... 2
Grading ................................................................. 2
Schedule ............................................................... 2
  Wk 1. Reproducible Science Tools ................................. 2
  Wk 2. Programming Concepts ..................................... 2
  Wk 3. Reading and Wrangling Data ............................... 2
  Wk 4. Tidying up Data ............................................. 4
  Wk 5. Visualizing Data ............................................ 4
  Wk 6. Interactive Plots and Maps ................................. 4
  Wk 7. Interactive Applications .................................... 4
  Wk 8. Advanced Programming .................................... 5
  Wk 9. Documentation and Testing ............................... 5
  Wk 10. Final Project Presentations ............................... 5

Overview

ESM 296-2S is an introduction to the management and analysis of environmental information, providing students with the necessary computational background for more advanced Bren courses. Topics include: the basic computing environment (hardware and operating systems); programming language concepts; program design; data organization; software tools; generic analytical techniques (relational algebra, graphics & visualization, etc.); and specific characteristics of environmental information. We’ll focus on using the R environment for data reading, manipulation, analysis and visualization. An emphasis will be placed on reproducability, including versioning such as using git and github.

Topics will be presented in weekly 3-hour modules mixing lectures and hands-on examples, using students' own computers. There are no prerequisites.

Instructors

- Naomi Tague (office hours TBD)
- Ben Best (office hours TBD)
Workload

Each week you will be given an assignment in class, and we will spend some time working on it there. The completed assignment will be due (submitted on Gauchospace) at the beginning of class the following week. For most assignments, you will work in pairs.

There will also be a short paper accompanied by an in class presentation to be submitted the final week of class. This project will provide a review of several examples of innovative applications of data analysis or computing that illustrate how the strategic use of informatics can change how we think about or approach solving environmental problems. You will also work in pairs for the final project.

Grading

- 70% assignments (7 assignments @ 10% each)
- 20% final project (paper + presentation)
- 10% participation

Schedule

Wk 1. Reproducible Science Tools

Environmental science and management is increasingly a group enterprise involving many stakeholders from various disciplines. Environmental science also increasingly requires collection, processing, analysis and interpretation of large data sets. There are a variety of tools that help make collaborative data analysis easier. We’ll focus this first week in getting you up to speed with the basics of operating two technologies that are currently the most popular and intuitive:

1. Rmarkdown enables you to weave rendered chunks of code to directly create tables, figures, formulas and references and output to documents, PDFs, websites or interactive online applications. You are expected to turn in all your homework assignments with this authoring format. See Rmarkdown cheat sheet, reference guide and Figure 1.

2. Git is the most popular file versioning software which allows you to play nicely with others when it comes to code and data. Github is the most popular online site for hosting git repositories, and has many bonus features for rendering formats (md, csv, geojson, . . .) and handling project management (issues, wiki,. . .). You’ll create your own personal private repository to turn in homework (a la Github Classroom) and manage your code and files like a project (ie turning in homework = resolving an issue).

Wk 2. Programming Concepts

Programming is a general term used for developing sets of instructions for data generation, analysis, interpretation and visualization. We will introduce some basic programming concepts: data types, flow control and functions We will also cover programming “best practices”. While the specific syntax here applies to R, the concepts are universal to all programming languages.

Wk 3. Reading and Wrangling Data

Getting your data into the format you require is often one of the most frustrating and time consuming task involved in data analysis. Fortunately there are tools that make this easier. You will become inculcated into the “Hadley”-verse of R packages which represent a new wonderful paradigm of data science which embraces readability of code. We’ll focus on these R packages in particular:
Figure 1: Rmarkdown can be used throughout the life cycle of your data to weave rendered code and documentation into a variety of formats, from portable static documents and to dynamic online applications. Figure source: Reproducible Research - Johns Hopkins University | Coursera
• **readr**: read and write tabular data with sensible defaults (i.e., no factors). We’ll also cover related packages such as rgdal to read and write spatial data.

• **dplyr** is your main data wrangling tool with a piping idiom (%>%%) that encourages very readable SQL-like sequential statements: `select`, `filter`, `arrange`, `group_by`, `summarize`. The other beauty about dplyr is that you can initially write for a simple CSV, and scale up the back end to work with databases (such as sqlite, MySQL, PostgreSQL or even Google BigQuery) and dplyr translates the backend functions automatically, so no need to rewrite the rest of your code (concept of “middleware”).

See also the data wrangling cheat sheet with dplyr, tidyr.

**Wk 4. Tidying up Data**

Data comes in a wide variety of formats. Literally. You’ll learn about “wide” vs “narrow” formats with the **tidyr** package, as well as how to handle dates/times with **lubridate**, and strings with **stringr**. We’ll throw in a bit about **regular expressions** for good measure.

**Wk 5. Visualizing Data**

Visualization allows you to find patterns in your data. Good visualization allows you to communicate what you learn from data to others. New tools provide users with efficient and flexible ways to generate elegant informative visualizations of their data. We will introduce you ‘best practices’ and R’s powerful visualization “grammar” **ggplot2** which allows you too quickly generate some pretty fancy plots and tailor them to your audience. See the **ggplot2** cheat sheet.

**Wk 6. Interactive Plots and Maps**

The majority of exciting interactive application development is happening these days on the web, and specifically with powerful JavaScript libraries (especially with node framework). R and particularly the RStudio environment have taken advantage of this with the new htmlwidgets architecture, which enables exciting interactive visualizations right from the RStudio IDE (as a Viewer pane), rendered as a standalone HTML document (so easy to share with colleagues or on website), and/or integrated within a Shiny application (for full featured slice and dice capabilities but dependant on an R backend engine; see next week). Check out the **htmlwidgets showcase** for a sample of the types of interactive visualizations made easy to render:

- **leaflet**: geospatial mapping
- **dygraphs**: time series charting
- **metrics graphics**: scatterplots and line charts with D3
- **networkD3**: graph data visualization with D3
- **d3heatmap**: interactive heatmaps with D3
- **dataTables**: tabular data display
- **threejs**: 3D scatterplots and globes
- **DiagrammeR**: Diagrams and flowcharts

**Wk 7. Interactive Applications**

Continuing with the online interactive theme, we’ll explore the world of making **Shiny** apps for truly interactive applications that allow for backend R functions reactive to user inputs to a clean web interface easily rendered with the most minimal amount of code. See the **shiny** cheat sheet.
Wk 8. Advanced Programming

Developing more complex programs involves breaking data analysis down into key components - and organizing these components so that they can be easily re-used, modified and linked with other programs. We will introduce you to techniques for structured programming. You’ll learn how to create your own R package.

Wk 9. Documentation and Testing

Two essential components of programming best practices are documentation and testing. Particularly when programming and data analysis involve multiple steps or collaborative programming, good documentation and testing are essential. We will introduce you to ways to help you to write documentation inline using roxygen2 and ways to automate testing of your programs.

Wk 10. Final Project Presentations

You’ll share your final project presentations in class, describing the scientific question asked, methodological steps taken to gather and clean data, analytical steps and visualizations. This will be done with an Rmarkdown presentation having a Shiny app embedded with all code made available on a Github repository (ie at your group’s org.github.io site).