# Instructor Information

Instructors: Jarrett Byrnes, PhD; Jillian Dunic, BSc

[Jarrett.Byrnes@umb.edu](mailto:Jarrett.Byrnes@umb.edu); [Jillian.Dunic001@umb.edu](mailto:Jillian.Dunic001@umb.edu)

Phone (M): 857-313-8296

Office Location: ISC 3201 – Research Staff

Office Hours: Wednesday 12:00 – 14:00

# Course Information

**Course Title:** Introduction to practical computing in R

**Credits:** 1.0

**Online?** no

**Course**

**Description:** The goal of this course is to make research and data analysis easier by introducing students to computing tools commonly used in biological data analysis. The focus of this course is on practical computing so that students will leave the course and implement these tools to effectively work on collaborative research projects and prepare reproducible research. This course is targeted at novice R users, however, students at all levels may benefit from lectures on reproducible research and data management. Students will learn through live-coding, in-class exercises, and short weekly assignments. This course will be taught using the statistical programming language R, however, the concepts covered are not specific to the choice of programming language. Assignments are designed to make students comfortable using additional tools such as git, Github, Rmarkdown, Make, and Shiny in conjunction with R.

**Context:** This course lays the foundation of computational skills that students will use in their thesis research. It will prepare students for advanced courses in data analysis and computation such as BIOL 607 – Introduction to Computational Data Analysis for Biology, BIOL 664 – DNA and Protein Sequence Analysis, or comparable courses in their own department.

**Prerequisites:** none

**Prerequisite**

**Skills:** Previous experience using a programming language will be helpful but is not necessary.

**Course**

**Objectives:** By fully participating in this course, you should be able to:

1. Use the version control system git to keep track of your coding history.
2. Create and use custom functions in R.
3. Prepare messy data for analysis by organizing it into the ‘tidy’ format described by Hadley Wickham.
4. Use R to visualise data.
5. Create an automated data analysis pipeline using Make.
6. Create a report using R Markdown.
7. Create an interactive data visualization app using Shiny.

**Core Goals:** The objectives for this course focus on the following core competencies:

1. Graduates should understand the value of reproducible research and be able to implement reproducible research strategies in their own work.
2. Graduates should have the confidence and skills to learn new and/or more advanced computing techniques using R or other software.
3. Graduates should have an appreciation for the ways that computational tools can improve the efficiency of their research.
4. Graduates should emerge as better data managers.

**Required**

**Assignments:**

**Weekly Assignments:** Students are expected to complete weekly assignments written using Rmarkdown and submitted via Github. Assignments will be evaluated using a check-plus, check-0, check-minus system. Students will receive additional feedback through peer review.

*Check-plus:* The student has gone beyond expectations and requirements (e.g., high effort, tools used not covered, sophisticated application of tools covered). They have correctly completed the task required. Their code will be efficient and human readable with complicated tasks broken down into smaller, simpler pieces. The student adheres to a commonly accepted coding style (e.g., Google’s R Style Guide, Hadley Wickham’s style guide) and comments their code clearly. Where appropriate, code may include ‘defensive programming’ with built-in error checking and fail-safes.

*Check-0:* The student meets expectations and requirements. They have correctly completed the task required and applied tools covered in the course appropriately. Their code is efficient and human readable, but may have some style errors, or may not use the most efficient methods. The student generally adheres to a commonly accepted coding style with some errors and includes some comments.

*Check-minus:* The student does not meet expectations and requirements. Their work is either incomplete or contains multiple errors. No consistent coding style is followed, complicated problems are handled in one ‘chunk’, and the code is difficult to read. Their code is inefficient with repetition that could have been functionalised. The student does not demonstrate competency using the tools covered in the course.

**Peer Review of Assignments:** Each week, students will be assigned to peer review two homework assignments. Peer reviews will also be assessed on the check-plus, check-0. Comments must be kind and constructive.

*Check-plus:* Comments must be kind, constructive and specific. There is an attempt to offer suggestions or help. If there are no problems with the assignment, the peer review should comment on what they have learned from the assignment or highlight a useful technique that they noticed.

**Course Rubric:** Checks will be converted to a score of 0 (not submitted) to 3 (check-plus) for conversion to grade-point average.

|  |  |  |
| --- | --- | --- |
| **Assignment/Deliverable** | **Number** | **Grade %** |
| **1. Weekly Assignments** | **13** | **80** |
| **2. Peer Review of Assignments** | **12** | **20** |
| **No final project/exam** |  |  |

**Course**

**Policies: Participation and attendance**

Students are expected to attend all classes, complete all required reading and writing assignments prior to class, thoughtfully participate in discussions, and take responsibility for creating a positive learning environment. This course is designed to be a collaborative learning experience and as such, discussion with your neighbor is encouraged. You are all resources for one another. Students are expected to follow along in code examples on their own computers, typing along as they go, rather than copying and pasting from any posted examples.

**Group work**

This course is designed to be collaborative. However, all work you complete should be done on your own.

# Grading

**Grading:** Grade type for the course is a whole or partial letter grade. (Please see table below)

Note: the lowest passing grade for a graduate student is a “C”. Grades lower than a “C” that are submitted by faculty will automatically be recorded as an “F”.

Please see the Graduate Catalog for more detailed information on the University’s grading policy.

|  |  |  |  |
| --- | --- | --- | --- |
| **Grading Policy** | | | |
|  | **Letter Grade** | **Percentage** | **Quality Points** |
|  | A | 93-100% | 4.00 |
|  | A- | 90-92% | 3.75 |
|  | B+ | 87-89% | 3.25 |
|  | B | 83-86% | 3.00 |
|  | B- | 80-82% | 2.75 |
|  | C+ | 77-79% | 2.25 |
|  | C | 73-76% | 2.00 |
|  | F | 0-72% | 0.0 |
|  | **INC** | A grade of Incomplete (INC) is not automatically awarded when a student fails to complete a course. Incompletes are given at the discretion of the instructor. They are awarded when satisfactory work has been accomplished in the majority of the course work, but the student is unable to complete course requirements as a result of circumstances beyond his/her control. The student must negotiate with and receive the approval of the course instructor in order to receive a grade of incomplete | N/A |
|  | IF | Received for failure to comply with contracted completion terms. | N/A |
|  | W | Received if withdrawal occurs before the withdrawal deadline. | N/A |
|  | AU | Audit (only permitted on space-available basis) | N/A |
|  | NA | Not Attending (student appeared on roster, but never attended class. Student is still responsible for tuition and fee charges unless withdrawal form is submitted before deadline. NA has no effect on cumulative GPA.) | N/A |

**Required Text(s):**

Google!

**-----------------------------------------End of BHE section no more than 5 pages------------------------------------**

**Software:** R – http://www.r-project.org/

RStudio – <http://ww.rstudio.org/>

Git – git-scm.com/downloads

**Recommended Texts:**

Beckerman AP. & Petchey, OL. (2012) *Getting started with R: An introduction for biologists. Oxford, UK: Oxford University Press.*

Haddock, S., & Dunn, C. (2011). *Practical computing for biologists.* Sunderland, Mass.: Sinauer Associates.

Wickham, H. (2009). *ggplot2 elegant graphics for data analysis*. Dordrecht: Springer.

**Other**

**Reading:** Additional readings will come from online sources.

# Course Schedule

Class 1: September 8, 2015

|  |  |
| --- | --- |
| **Topic** | Introduction to reproducible research, git, and class overview |
| **Learning**  **Objectives:** | * Define reproducible research * Explain why reproducible research is important * Install/setup git on a computer |
| **Reading**  **Assignment** | Dynamic ecology: <https://dynamicecology.wordpress.com/2015/02/18/the-biggest-benefit-of-my-shift-to-r-reproducibility/>  Ropensci: <https://ropensci.org/blog/2014/06/09/reproducibility/>  Christopher Gandrud: <http://christophergandrud.blogspot.com/2013/07/getting-started-with-reproducible.html> |
| **Synopsis and course goals** | Today will be a day for troubleshooting and getting everyone setup with their computer. Ideally everyone should come to class having attempted the pre-class homework and have contacted me if they struggled with it. However, today is the day that we get everyone on track for some smooth sailing. This week will be a bit of a bumpy ride but that is completely okay! We’re going to take our time setting up so that we all start on the same page once we get started doing the fun things in R! |
| Assignment:  Due Date: | See class 2. |

Class 2: September 9, 2015

|  |  |
| --- | --- |
| **Topic** | Version control using git + Github |
| **Learning**  **Objectives:** | * Initiate a git repository * Commit a change to a remote git repository * Describe what git is and how it can be used in your research |
| **Reading**  **Assignment** | https://try.github.io/levels/1/challenges/1 |
| **Synopsis and course goals** | We will introduce a tool called git, which is used for version control. Think ‘infinite undo’ and ‘better track changes’. Git is a powerful tool, but it isn’t just for computer scientists and software engineers. The initial set up can be a challenge which is why we’re going to spend an entire class on set up. After that we will use git everyday so that by the end of class it will be trivial. Today we will learn the very basics of git so that you can keep track of changes that you make to your code, your data, or even a text document. Like I said, ‘infinite undo’, we can all think of cases where that would come in handy!  Today will be a bit daunting because along with git we’ll start to work with the command line/terminal. Why bother when there are graphical user interfaces (point and click tools) that you can use? A large part of this course is teaching you how to *learn* to use computational tools that are becoming increasingly important in biological data analysis and I want you to be prepared to face using the command line as it does sneak into work where you may not expect it. Don’t worry, today will be confusing for many of you, and that’s okay! That’s why we’re going to practice this everyday! |
| Assignment:  Due Date: | 1 - Complete the git introductory tutorial and push your first commit  Due September 10th. 2015 |

Class 3: September 10, 2015

|  |  |
| --- | --- |
| **Topic** | An introduction to Rstudio and Rmarkdown |
| **Learning**  **Objectives:** | * Create an Rstudio project * Create an Rmarkdown html document * Commit and push changes to remote git repository |
| **Reading**  **Assignment** | https://try.github.io/levels/1/challenges/1 |
| **Synopsis and course goals** | Today we are going to start our first Rstudio project! Rstudio is an awesome tool that makes working in R much friendlier! At its simplest, it organises all the windows you’ll need to have open to work in R. But beyond that, it has many extra tools that make R friendlier to novice and advanced users.  ‘Projects’ will allow us to contain all the work that we deem as part of a project in one collection. Ultimately it is just a folder on your computer with a bit of extra information stored by Rstudio.  In our project we will create our first Rmarkdown document. These documents can be very useful as they can contain code and coded output like graphs, or model outputs. Many people even use them to write their manuscripts. They are a key aspect of reproducible research because they are automatically generated when changes to the code or text are made. This means you could start a document with pilot data and then rerun it as you get more data! A great way to start writing on the go.  We will also make our Rstudio project a git project as well! Then we will tell git to keep track of all the changes we make to our Rmarkdown document. If by the end of the day, you’re still unsure of git, don’t worry! Today was our first run of the workflow that we will employ throughout the course. If not today, then in the next week or two you’ll be ready to start this workflow with your own research! Start naming those manuscript folders! |
| Assignment(s):  Due Date: | See class 4 |

Class 4: September 11, 2015

|  |  |
| --- | --- |
| **Topic** | Setting up R and meeting our dataset |
| **Learning**  **Objectives:** | * Interpret the output of str() * Read in a data file using base::read.csv() and readr::read\_csv() * Install packages using Rstudio interface * Install packages using base::install.packages() |
| **Reading**  **Assignment** | Hadley Wickham style guide: <http://adv-r.had.co.nz/Style.html>  Google style guide: <https://google-styleguide.googlecode.com/svn/trunk/Rguide.xml>  Tim Smith's "aRrgh: a newcomer's (angry) guide to R": <http://tim-smith.us/arrgh/> |
| **Synopsis and course goals** | Now that everything is set up, we are really going to get started in R. We will learn some of the basics of reading data into R, installing packages, and learning common pitfalls to watch out for. A large part of today, and particularly the assignment for this week, will be learning how to decode error messages that you get because these often make programming more daunting than it needs to be. The assignment for this week will cover the most simple and common mistakes that are made when coding. Google will be your friend here.  Today we will also meet our dataset. A key aspect of our dataset is that it is going to be clean and appropriately organised. This dataset may be one of your first experiences with a ‘tidy’ dataset. We will briefly touch on what ‘tidy’ data is, in preparation for using it later in the course. |
| Assignment:  Due Date: | 2 - Let me google that for you: common errors in coding in R!  Due September 17th. 2015 |

Class 5: September 15, 2015

|  |  |
| --- | --- |
| **Topic** | Data.frame basics and an introduction to ggplot2 |
| **Learning**  **Objectives:** | * Index data.frames using $ and [] * View properties of data.frames using summary(), str(), head(), and tail() * Create simple plots in base R using plot() and hist() * Make a scatterplot, histogram, and boxplot using ggplot2 |
| **Reading**  **Assignment** | Wickham, H. (2011). The split-apply-combine strategy for data analysis. The Journal of Statistical Software. 40(1), pp. 1–29  Data wrangling cheatsheet: http://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf |
| **Synopsis and course goals** | Today we’re going to get into some basics that you’ll need to know as you start to work with data. We’re going to talk about objects, but most importantly we’re going to get you into the habit of being aware of what you’re working with! It’s too easy to just type words and functions into R but not actually know what’s going on. Today we meet str(). This will become your best friend. When you get an error, you will check str(). It will solve at least 50% of your problems, which when you’re starting, is a lot!  We’ll start practicing the simple tricks in R that you’re going to use everyday. We’ll learn how to index objects (the equivalent of selecting a cell in Excel), make quick and dirty plots (always good to visualise your data), learn the different data types (e.g., numbers, factors, characters).  Then we’ll follow this up with an assignment on common mistakes that you will make! Nothing is more frustrating when you’re just learning and you’ve written a great code snippet… that doesn’t work because you forgot a comma! |
| Assignment:  Due Date: | 2 - Let me google that for you: common errors in coding in R!  Due September 17th. 2015 |

Class 6: September 17, 2015

|  |  |
| --- | --- |
| **Topic** | ggplot2 continued and open Q&A |
| **Learning**  **Objectives:** | * Use colour to visualise groupings of different factors in a dataset * Use facets to visualise groupings of different factors in a dataset |
| **Reading**  **Assignment** | Wickham, H. (2011). The split-apply-combine strategy for data analysis. The Journal of Statistical Software. 40(1), pp. 1–29  Data wrangling cheatsheet: http://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf |
| **Synopsis and course goals** | We have learned so much up to now!  We’re going to wrap up some more techniques in ggplot2 and leave this time for some open Q&A. Bring all your questions on anything that we’ve covered up to date, or even some things that you are curious about that may be to come.  We’ll also use this time to start to assess how we feel about what is to come. Do we need to slow down? Speed up? This is the day to share. Please be honest and open! No matter how you’re feeling there are probably others in the class that feel the same! |
| Assignment:  Due Date: | 3 - Create these graphs part I  Due September 22th. 2015 |

Class 7: September 22, 2015

|  |  |
| --- | --- |
| **Topic** | Data aggregation: getting the summary statistics that you want |
| **Learning**  **Objectives:** | * Use dplyr to get summary statistics (mean, median, totals, min, max, range) for groupings in a dataframe * Use colours in ggplot2 to visualise summary statistics |
| **Reading**  **Assignment** | Wickham, H. (2011). The split-apply-combine strategy for data analysis. The Journal of Statistical Software. 40(1), pp. 1–29  Data wrangling cheatsheet: http://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf |
| **Synopsis and course goals** | Today we will introduce a new package that has a friendlier syntax than the base R code that some of you have have experienced. More experienced users will probably be jealous that they didn’t get to learn this package from the start – (no more using the which() function! ). Meanwhile, novice users will have a gentle introduction to the power of coding. No more manually grouping things in excel, colour coding, or spreading your data across 15 different worksheets. With a few lines of code you can breakup your data into all sorts of groups and calculate summary statistics like means, medians, sums, etc., The application of dplyr functions with ‘tidy’ data will show you the importance of organising data in the appropriate way. Human readable is not always the easiest to work with.  By now we’re ggplot2 masters! We can make even cooler graphics that include these summary statistics. |
| Assignment:  Due Date: | 3 - Create these graphs part I  Due September 29th. 2015 |

Class 8: September 24, 2015

|  |  |
| --- | --- |
| **Topic** | R data structures: we’ve seen them, now let’s understand what they’re all about |
| **Learning**  **Objectives:** | * Describe the difference between vectors, matrices, arrays, dataframes, lists * Recognize an attribute of a data structure * Describe the difference between atomic vector types: character, double (numeric), factor, and logical |
| **Reading**  **Assignment** | Advanced R – Chapter: Data structures - <http://adv-r.had.co.nz/Data-structures.html> |
| **Synopsis and course goals** | We introduced R data structures when we first looked at data in R. We didn’t go in depth then because, frankly they’re a boring introduction to R. But now that we’ve started to use them more intimately I’m sure you all have some questions. What is a data.frame really? I know lists and data.frames are different but how and why? There are so many ways of indexing, what is the best way? What is this type ‘double’ business?  Now that you’ve had exposure to different data types and structures, we’ll talk about the differences and when and why they matter. It is also really important to at least have an idea about the differences because some of you may come across more complex objects (e.g., spatial objects) and learning some of the characteristics of data types will help you navigate these objects in future! |
| Assignment:  Due Date: | See class 5 |

Class 9: September 29, 2015

|  |  |
| --- | --- |
| **Topic** | Doing the same thing over and over? Writing for loops |
| **Learning**  **Objectives:** | * Describe what a for loop does * Use a for loop to iterate over a set of values |
| **Reading**  **Assignment** | (Chapter 9) Haddock, S., & Dunn, C. (2011). Decisions and loops. In practical computing for biologists. Sunderland, Mass.: Sinauer Associates. |
| **Synopsis and course goals** | DRY! Do not Repeat Yourself!  You are going to run a lot of analyses. You probably have multiple replicates. You’re probably going to want to make the same graph over and over again, but with a slight change each time as you dig deeper into your data.  Today we’re going to start learning how to run the same thing over and over again. But without copying and pasting. Without going through and manually highlighting subsets of your Excel spreadsheet.  We will write our first for loop. It will be daunting but it will also be empowering. We’re going to look at the for loop again on Thursday so that by the end of the week you will be rocking the for loop.  We even have a cool hands-on (something actually physical!) exercise to drive the point home, for the kinaesthetic learners out there. |
| Assignment:  Due Date: | 4 - For loop assignment using conditionals  Due October 6th. 2015 |

Class 10: October 1, 2015

|  |  |
| --- | --- |
| **Topic** | Conditional statements (if/else), more R operators, and an intro to browser() |
| **Learning**  **Objectives:** | * Use if/else statements to control flow of code * Use the R operators ‘|’ ‘&’, ‘==’, ‘!’ to test logical conditions * Use browser() to identify an error in a for loop |
| **Reading**  **Assignment** | (Chapter 9) Haddock, S., & Dunn, C. (2011). Decisions and loops. In practical computing for biologists. Sunderland, Mass.: Sinauer Associates.  Control flow help page: <http://stat.ethz.ch/R-manual/R-patched/library/base/html/Control.html>  Control flow: <https://howtoprogramwithjava.com/the-5-basic-concepts-of-any-programming-language-concept-2/> |
| **Synopsis and course goals** | R operators! That is what you need to google if you need to look up the symbols that allow you to tell R ‘and’, ‘or’, ‘not’, etc.,  Today we get a tour of the operators that we use to test conditional statements (If X do Y). Then we’ll integrate them into our for loops. Sometimes you don’t want to save a result if for example, the value is less than zero.  We’ll also meet browser(). I use this function almost every time I write a for loop or a function. It let’s you see ‘inside’ the for loop. You’ll get what I mean when we see it in class. It will be our first real exposure to thinking about ‘environments’ when programming. browser() is really important in helping you debug your code because as we keep hearing, it lets you see what is actually happening behind those words you type on the screen. It’s often not what you think, which is why your code doesn’t work when you think it should! |
| Assignment:  Due Date: | See class 9 |

Class 11: October 6, 2015

|  |  |
| --- | --- |
| **Topic** | Want more than means and totals? Let’s write some functions! |
| **Learning**  **Objectives:** | * Describe the three components of a function * Create a simple custom function |
| **Reading**  **Assignment** | Writing functions:  <http://nicercode.github.io/guides/functions/>  Advanced R: <http://adv-r.had.co.nz/Functions.html> |
| **Synopsis and course goals** | Writing functions in R gives you more flexibility in what you can do in R. dplyr was awesome. But sometimes you need more than just a mean or a total.  Sometimes the model output in R gives you a huge list of values but you only need one, sometimes you want to make the same graph over and over again but with a different dataset each time, sometimes you need to run the same model over and over again but with a slightly different parameter.  Or you want to run a power analysis – how many plots should I sample to detect a significant (biologically and statistically) effect size? Or maybe for management purposes you want to visualise the survival probabilities for a population?  You get the point. There are a lot of tasks you could run again and again.  We’re going to write our first custom function today. Even if you never go on to write a function (though I think you will surprise yourself), this will give you a better understanding of *how* functions in R work. |
| Assignment:  Due Date: | 5 - Write your own functions and debug my broken ones  Due October 13th. 2015 |

Class 12: October 8, 2015

|  |  |
| --- | --- |
| **Topic** | Writing functions part II: debugging your code |
| **Learning**  **Objectives:** | * Use tryCatch() to handle errors/warnings * Combine debugging tools and flow control tools in a custom function |
| **Reading**  **Assignment** | Writing functions:  <http://nicercode.github.io/guides/functions/>  Advanced R - Functions: <http://adv-r.had.co.nz/Functions.html> |
| **Synopsis and course goals** | “I’m not a programmer, why are we talking about debugging???”  You’re a programmer now! You have been writing code for the last month! You’ve also hit stumbling blocks and learned that there are lots of possible errors that can creep up. Some of these errors are errors in your code. Sometimes though, there are just numerical problems. A model won’t converge for a given subset of your data, some cool new package you just learned fails to find a matching species name, one of your genbank queries out of a 1000 fails. You *could* go through line by line and find the problem… But we’re supposed to making your life easier. Today we’re going to learn a way to automate catching and/or skipping over these errors. |
| Assignment:  Due Date: | See class 11 |

Class 13: October 13, 2015

|  |  |
| --- | --- |
| **Topic** | Simulating data |
| **Learning**  **Objectives:** | * Generate data from a distribution (e.g., rnorm()) * Use replicate() to simulate data from a given function (equation) * Use ggplot2 to plot simulated data |
| **Reading**  **Assignment** | TBA |
| **Synopsis and course goals** | Simulating data can be incredibly useful. Depending on where you go with your data analysis, or what classes you take in the future, this is a skill you will need to use.  You don’t even need to include simulations in your work. Sometimes simulations can be helpful just for *you,* personally, to understand patterns in your data. And as usual, there is nothing more rewarding than getting a pretty picture from your code block. We’re also going to get some practice turning and equation into a function. The homework assignment will focus on the equation for discrete population growth, something likely familiar to all of you. |
| Assignment:  Due Date: | 6 - Discrete population growth  Due October 20th. 2015 |

Class 14: October 15, 2015

|  |  |
| --- | --- |
| **Topic** | Git + Github extension: collaboration |
| **Learning**  **Objectives:** | * Create a new branch from an existing repository, make a change, and submit a pull request * Use partial staging to break changes you have made into smaller chunks |
| **Reading**  **Assignment** | TBA |
| **Synopsis and course goals** | We’ve hit the half-way point in the class and now we’re going to shift gears. We have the basic tools to navigate R and our computers!  Today is an interlude. We’ve been work with git daily. At this point it is probably feeling quite trivial. And we’ve already touched on cloning, making changes, and then submitting pull requests in class when one of you has an error in their code and just can’t figure it out. It’s the workflow that I use to help you make a correction.  We’re going to take a day (or two if necessary) to learn some git tools that are very helpful for collaboration. This may be collaborating on a manuscript (little or no code involved), or working on an analysis with a colleague, or contributing to a common data file (read: a better, much less painful google spreadsheet with better track changes ability!).  Git use starts to get more complicated now that we’re working as a team. But we’ll touch on it again next class to check in with how everyone is feeling. We can assess how much more we want to practice the skills from today, but I want you to at least get some exposure, as this is becoming an increasingly common tool used in research collaborations. |
| Assignment:  Due Date: | See class 13 |

Class 15: October 20, 2015

|  |  |
| --- | --- |
| **Topic** | Introduction to relational databases and considerations for better data management |
| **Learning**  **Objectives:** | * Define a relational database * Draw a schema for a simple, 4 table relational database (one-to-one, and one-to-many relationships) * Explain to another person, the relationships between tables/variables given a database schema |
| **Reading**  **Assignment** | (Chapter 15) Haddock, S., & Dunn, C. (2011). Relational Databases. In practical computing for biologists. Sunderland, Mass.: Sinauer Associates. |
| **Synopsis and course goals** | Data. Data. Data.  If you’re in this class, you’re likely doing research, or interested in doing research. This means that you will be collecting some data. Yet, we (particularly biologists) NEVER get exposure to how to organise data. We’re used to recording data in lab notebooks. We focus on organising data so that we can read it. But for anyone that has tried to start analysing data, or supply data to an online repository (something that is an increasingly common requirement for publication), or extract data from online databases, data management is not trivial.  You aren’t going to walk out of this class and go build a database tomorrow. You will walk out of today’s class with a new respect for data management and a set of tools that you can use to start organising your data better. Whether you have a small dataset (yes ecologists, this is probably you with your n = 3) or a large dataset with thousands of lines of data and hundreds of variables, the basic principles of organisation are the same.  Today is about giving you the vocabulary to ask for help from people who are data management experts. Today is about showing you some tricks that help you organise data that you otherwise don’t know how to store in an Excel spreadsheet. We will walk through real life examples that I have faced (not even with big data!) and I’ll show you that there are solutions. No, you are not going to be data engineers after this lecture, but you will be able to talk to one. |
| Assignment:  Due Date: | 7 - Tidy your ~~room~~ data! Part I  Due October 27th. 2015 |

Class 16: October 22, 2015

|  |  |
| --- | --- |
| **Topic** | Data wrangling part I: Tidy data |
| **Learning**  **Objectives:** | * Draw a schematic for a tidy dataset given a messy dataset * Use the package tidyr to convert a messy dataset into a tidy datset |
| **Reading**  **Assignment** | Reread Tidy data: Wickham, H. (2014). Tidy data. The Journal of Statistical Software, 59(10), 1-24. Retrieved from <http://www.jstatsoft.org/v59/i10/> |
| **Synopsis and course goals** | We’ve had an introduction to relational databases. This is a perfect segue into tidy data. You’ve seen tidy data. You’ve probably started to grow an appreciation for how tidy data makes your analysis much simpler.  But those beautiful, tidy datasets that you have seen, typically come from ugly, ugly origins. Data clean-up/carpentry is probably the most difficult skill that you will learn. However, it is the skill that you will probably use 90% of the time in your research (well once you start working with your data). We now have the foundation to start tidying our data. We will work together on tidying a single dataset, however, I encourage you to start tidying your own data and bring the problems that you encounter to class.  The first step in turning messy data into tidy data, is to be able to sketch what the final product needs to look like. We will start with pencil and pen today, then move onto learn about tidyr, a new tool to convert messy data to tidy data! |
| Assignment:  Due Date: | See class 15 |

Class 17: October 27, 2015

|  |  |
| --- | --- |
| **Topic** | Data wrangling part II: Manipulating data frames |
| **Learning**  **Objectives:** | * Use merge() to combine dataframes * Describe a ‘join’ between data tables/data frames * Advanced objective: use a one-to-many join |
| **Reading**  **Assignment** | TBA |
| **Synopsis and course goals** | Today, you’re going to recognise some of the concepts of the relational database that we covered last week! Without a doubt, almost all of you are going to have to combine more than one data file or Excel worksheet. Today we learn how to do this, and the different ways that this can be done.  As usual, we’ll use some simplified, real-world examples of when to do this. We’ll also cover some of the hurdles that you can hit, particularly when reading in different datasets. |
| Assignment:  Due Date: | 8 - Tidy your ~~room~~ data! Part II  Due November 3rd. 2015 |

Class 18: October 29, 2015

|  |  |
| --- | --- |
| **Topic** | Data wrangling part III: Using regular expressions, other string manipulations, dates |
| **Learning**  **Objectives:** | * Use regular expressions to find and replace strings/characters in a dataset * Use lubridate to manipulate date strings * Use tidyr::split() to split strings into multiple columns |
| **Reading**  **Assignment** | Regular expression interactive tutorial: <http://regexone.com/lesson/0> |
| **Synopsis and course goals** | Dates. Dates are the WORST.  2015/08/02; 02/08/2015; 08/02/2015; 02/Aug/2015; 02-Aug-2015; 02-August-2015; 02-Aug/15…  You get the idea. Today we learn how to deal with this. Today we learn how to tell R what day we really mean in one line of code. We also reinforce how important it is to think about the data that you are going to collect, *before* you collect it.  In addition to dates, we’ll introduce regular expressions. They can be confusing, but they are also incredibly powerful. We won’t walk out as regex masters. But I’ll show you an awesome online resource, and give you some practice so that when you do need it, you know where to look and how to start.  And last, we’ll explore how we can take our compound strings and break them into unique columns so that we can make sure that we have tidy data. |
| Assignment:  Due Date: | See class 17 |

Class 19: November 3, 2015

|  |  |
| --- | --- |
| **Topic** | Workflow and data analysis pipelines – current you protecting future you |
| **Learning**  **Objectives:** | * Describe a data analysis pipeline * Create a schematic for a data analysis pipeline * Setup a flat file data analysis pipeline |
| **Reading**  **Assignment** | Blog post on workflow using R: http://robjhyndman.com/hyndsight/workflow-in-r/  More TBA |
| **Synopsis and course goals** | A huge part of this course is protecting future you, from current you. This is a pleasant by-product of making your research reproducible. Today we will formally talk about how to setup a reproducible workflow and data analysis pipeline.  Opening an old project a week, a month, or a year later – this is a gut wrenching moment. Your code, scattered across 25 different files – test1.R, test2.R, test\_trial1?.R. Thirty-two different versions of your data – data\_clean1.R, data\_clean2.csv, data\_clean\_for\_real\_this\_time.csv, data\_ignore\_the\_last\_file.csv.  Today we learn a few different ways that you can organise your project so that this doesn’t happen. So that if you update your data, you can run your scripts in order and a fully updated version of your analysis, or even better, your manuscript gets spit out on the other end. For today, this is mostly organisational with minimal coding. But on Thursday we will talk about how you can automate this using a package in R called remake. |
| Assignment:  Due Date: | 9 - Saving future you from current you: implementing a workflow  Due November 10th. 2015 |

Class 20: November 5, 2015

|  |  |
| --- | --- |
| **Topic** | Introduction to Make |
| **Learning**  **Objectives:** | * Use Make to automate a data analysis pipeline |
| **Reading**  **Assignment** | TBA |
| **Synopsis and course goals** | Depending on the size and complexity of your analysis. You won’t want to remember the exact order of how to run everything. Luckily there are tools that will allow you to tell the computer what your workflow is, and when ready, you can rerun that analysis, without relying on your memory to tell you the order in which every step needs to be done.  As your analysis progresses, certain steps depend on others. By automating this process, you ensure that dependencies are managed without having to manually check them every time you update your work. |
| Assignment:  Due Date: | See class 19 |

Class 21: November 10, 2015

|  |  |
| --- | --- |
| **Topic** | Interactive visualisations using Shiny – part I |
| **Learning**  **Objectives:** | * Describe the Shiny R package and identify a possible use case * Build a Shiny user interface * Add control widgets |
| **Reading**  **Assignment** | TBA |
| **Synopsis and course goals** | Back to more fun graphics!  The Rstudio team builds some really incredible tools, beyond Rstudio. Shiny is tool that lets you build interactive visualisations. This means you have produce a graph and then include a slider bar so that you can change a variable and watch how your graph changes!  The applications range from small, personal use. This tool can allow you to modify parameters in your research and test the response in a quick and dirty way. But you can think even bigger. Think about the public outreach section in your NSF proposal. Imagine being able to include the development of a Shiny app that will let managers visualise the change success probability of increasing lion habitat size, see how kelp abundance can change as wave frequency and intensity increase, visualise the outcome of herd immunity as number of vaccinated persons in a community changes.  Today we’re going to learn the foundations of creating your first Shiny app. |
| Assignment:  Due Date: | 10 - Interactive visualisations! Your first shiny app!  Due November 17th. 2015 |

Class 22: November 12, 2015

|  |  |
| --- | --- |
| **Topic** | Interactive visualisations using Shiny – part II |
| **Learning**  **Objectives:** | * TBA |
| **Reading**  **Assignment** | TBA |
| **Synopsis and course goals** | More Shiny! You should have the tools to finish your first Shiny app after this class. |
| Assignment:  Due Date: | See class 21 |

Class 23: November 17, 2015

|  |  |
| --- | --- |
| **Topic** | Accessing online data through R |
| **Learning**  **Objectives:** | * Use R to import data directly from the internet * Use the googlesheets package to read data from google spreadsheets |
| **Reading**  **Assignment** | TBA |
| **Synopsis and course goals** | Open data is on the rise! For many of you looking for final project ideas, or additional data sources to include in your thesis work, today we learn how to access data sources through R! There are a huge and rapidly increasing number of packages to help you access online data, so today we will only touch on a few. I will try and touch on ones that span the range of interests in our class. |
| Assignment(s):  Due Date: | No homework |

Class 24: November 19, 2015

|  |  |
| --- | --- |
| **Topic** | Optional special topics: Class to choose from list below |
| **Learning**  **Objectives:** | * TBA |
| **Reading**  **Assignment** | TBA |
| **Synopsis and course goals** | This time gives us wiggle room to slow the course down as necessary, or to address special topics selected by the class. |
| Assignment:  Due Date: | 11 - TBA |

Class 25: November 24, 2015

|  |  |
| --- | --- |
| **Topic** | Optional special topics: Class to choose from list below |
| **Learning**  **Objectives:** | * TBA |
| **Reading**  **Assignment** | TBA |
| Assignment:  Due Date: | 12 - TBA |

Class 26: November 26, 2015 – No Class - Thanksgiving

**Additional Special Topics -** include but are not limited to:’

* Introduction to mapping in R
* Analysing spatial data in R using raster
* Advanced graphics (e.g., heatmaps, networks)
* Guest lecture on using High Performance Computing resources (Chimera)
* Databases and SQL
* Building an R package

# Methods of Instruction

**Methods:**

This course will consist of lecture, demonstration of live-coding by the instructor, and discussion. Students will be expected to have a computer available during the course so that they can follow along in lecture and work on in-class exercises. Included during the lecture will be short exercises for students to test their knowledge and students are encouraged to work collaboratively to discuss the answers to these exercises.

# Accommodations

The University of Massachusetts Boston is committed to providing reasonable academic accommodations for all students with disabilities. This syllabus is available in alternate format upon request. If you have a disability and feel you will need accommodations in this course, please contact the Ross Center for Disability Services, Campus Center, Upper Level, Room 211 at 617.287.7430. <http://www.umb.edu/academics/vpass/disability/> After registration with the Ross Center, a student should present and discuss the accommodations with the professor. Although a student can request accommodations at any time, we recommend that students inform the professor of the need for accommodations by the end of the Drop/Add period to ensure that accommodations are available for the entirety of the course.

# Academic Integrity and the Code of Student Conduct

***Code of Conduct and Academic Integrity***

It is the expressed policy of the University that every aspect of academic life--not only formal coursework situations, but all relationships and interactions connected to the educational process--shall be conducted in an absolutely and uncompromisingly honest manner. The University presupposes that any submission of work for academic credit is the student’s own and is in compliance with University policies, including its policies on appropriate citation and plagiarism. These policies are spelled out in the Code of Student Conduct. Students are required to adhere to the Code of Student Conduct, including requirements for academic honesty, as delineated in the University of Massachusetts Boston Graduate Catalogue and relevant program student handbook(s).[UMB Code of Student Conduct](http://www.umb.edu/life_on_campus/policies/code/)

You are encouraged to visit and review the UMass website on *Correct Citation and Avoiding Plagiarism:* <http://umb.libguides.com/citations>

# Other Pertinent and Important Information

You are advised to retain a copy of this syllabus in your personal files for use when applying for future degrees, certification, licensure, or transfer of credit.

**Note to novice users**

This course is inspired and draws heavily from the successful and thoughtfully designed Software / Data Carpentry boot camps. These boot camps are typically 2-day long workshops that expose complete novices to tools commonly used in programming and data analysis. The keys to success include:

* Focus on active learning - every 10-15 minutes interactive exercises will be given
* This class is designed to be collaborative, not in the group-project sense, but with respect to learning. You are not on your own! There will be time to turn to your neighbour and ask, “can you help me?”
* Live coding has three benefits:

1. You will watch me make mistakes—it happens to all of us.
2. I can’t go much faster than you.
3. We all start with blank text files!

* Weekly check-ins – we have time in the schedule to slow down and respond to feedback. This is not a race. I want you to leave this course feeling confident and excited by possibility.
* Repetition – we will practice using the tools that we learn every week until they become habit/trivial. Our focus on the practical means that you will be able to immediately implement the tools we learn in your own research.

# Bibliography

CodeSchool. (n.d.). TryGit. Retrieved June 11, 2015, from https://try.github.io/levels/1/challenges/1

Duffy, M. (2015, February 18). The biggest benefit of my switch to R? Reproducibility. Retrieved June 11, 2015, from https://dynamicecology.wordpress.com/2015/02/18/the-biggest-benefit-of-my-shift-to-r-reproducibility/

FitzJohn, R., & Falster, D. (2013, April 30). Functions. Retrieved June 11, 2015, from http://nicercode.github.io/guides/functions/

FitzJohn, R., Pennell, M., Zanne, A., & Cornwell, W. (2014, June 9). Reproducible research is still a challenge. Retrieved June 11, 2015, from https://ropensci.org/blog/2014/06/09/reproducibility/

Gandrud, C. (2013, July 16). Getting started with reproducible research: A chapter from my new book. Retrieved June 11, 2015, from http://christophergandrud.blogspot.com/2013/07/getting-started-with-reproducible.html

Google's R Style Guide. (n.d.). Retrieved June 11, 2015, from https://google-styleguide.googlecode.com/svn/trunk/Rguide.xml

Haddock, S., & Dunn, C. (2011). Decisions and loops. In *practical computing for biologists*. Sunderland, Mass.: Sinauer Associates.

Haddock, S., & Dunn, C. (2011). Relational databases. In *practical computing for biologists*. Sunderland, Mass.: Sinauer Associates.

Hyndman, R. (2009, September 17). Workflow in R. Retrieved June 11, 2015, from http://robjhyndman.com/hyndsight/workflow-in-r/

Page, T. (2012, July 1). The 5 Basic Concepts of any Programming Language - Concept #2 - How to Program with Java. Retrieved June 11, 2015, from https://howtoprogramwithjava.com/the-5-basic-concepts-of-any-programming-language-concept-2/

RegexOne. (n.d.). Learn regular expressions with simple, interactive examples. Retrieved June 11, 2015, from http://regexone.com/lesson/0

Smith, T., & Ushey, K. (n.d.). ARrgh: A newcomer's (angry) guide to R. Retrieved June 11, 2015, from http://tim-smith.us/arrgh/

Wickham, Hadley. (2010). A layered grammar of graphics. Journal of Computational and Graphical Statistics, 19(1), pp. 3–28

Wickham, H. (2011). The split-apply-combine strategy for data analysis. The Journal of Statistical Software. 40(1), pp. 1–29

Wickham, H. (2014). Tidy data. The Journal of Statistical Software, 59(10), 1-24. Retrieved from http://www.jstatsoft.org/v59/i10

Wickham, H. (2015). Data structures. Retrieved June 11, 2015, from http://adv-r.had.co.nz/Data-structures.html

Wickham, H. (2015). Style guide. Retrieved June 11, 2015, from http://adv-r.had.co.nz/Style.html