# Notes to Instructors for Math 377

Professor Bradley Warner

07 May, 2020

## Introduction

Math 377 - Advanced Probability and Statistics, is for most of our students their first introduction to a course that emphasizes analysis. In the past this course was more of a mathematical statistics course. We still have the mathematics but now have moved to add more computational lessons. Computational solutions offer easier assess to problems for students, can be more easily adapted to novel problems, and are the preferred method in many academic and industry settings.

### Software

We have selected `R` for our course software. It is open source, it is used extensively in industry and academia, and it is a statistical package.

The work of Hadley Wickham to emphasize the tidy data framework is an important contribution. It requires the researcher to understand the data in relationship to questions being asked.

There are several papers that help guide the thinking for this course and they are posted to the *Stats and Data Science Course Design* team under the Math 377 group in a folder called resources for mosaic. Let's recap some important ideas from the mosaic team:

> Think like a programmer. We don't really think of our classroom use of R as programming since we use R in a mostly declarative rather than algorithmic way. It doesn't take sophisticated programming skills to be good at using R. In fact, most uses of R for teaching statistics can be done working one step at a time, where each line of code does one complete and useful task. After inspecting the output (and perhaps saving it for further computation later), one can proceed to the next operation. Nevertheless, we can borrow from the collective wisdom of the programming community and adopt some practices that will make our experience more pleasurable, more efficient, and less error-prone.

1. Store your code in a file.
   It can be tempting to do everything in the console. But the console is ephemeral. It is better to get into the habit of storing code in files. Get in the habit (and get your students in the habit) of working with R scripts and especially RMarkdown files. More Info `R` can be used to create executable scripts. Option parsing and handling is supported with the `optparse` package. RStudio has additional options for executing some or all lines in a file. See the buttons in the tab for any `R` script, RMarkdown or Rnw file. (You can create a new file in the main File menu.) If you work at the console's interactive prompt and later wish you had been putting your commands into a file, you can save your past commands with savehistory("someRCommandsIalmostLost.R") In RStudio, you can selectively copy portions of your history to a script file (or the console) using the History tab.
2. Use meaningful names.
   Rarely should objects be named with a single letter. Adopt a personal convention regarding case of

letters. This will mean you have one less thing to remember when trying to recall the name of an object. For example, in the mosaicData package, all data frames begin with a capital letter. Most variables begin with a lower case letter (a few exceptions are made for some variables with names that are well-known in their capitalized form).

3. Adopt reusable idioms. Computer programmers refer to the little patterns that recur throughout their code as idioms. For example, here is a "compute, save, display" idiom.

```
# compute, save, display idiom
footModel <- lm(length ~ width, data=KidsFeet); footModel
Call:
lm(formula = length ~ width, data = KidsFeet)
Coefficients:
(Intercept) width
9.82 1.66
start teaching with r 113
# alternative that reflects the order of operations
lm(length ~ width, data=KidsFeet) -> footModel; footModel
Call:
lm(formula = length ~ width, data = KidsFeet)
Coefficients:
(Intercept) width
9.82 1.66
```

Often there are multiple ways to do the same thing in R, but if you adopt good programming idioms, it will be clearer to both you and your students what you are doing.

4. Write reusable functions.
   Learning to write your own functions will greatly increase your efficiency and also help you understand better how `R` works. This, in turn, will help you debug your students error messages. It also makes it possible for you to simplify tasks you want your students to be able to do in `R`. That is how the mosaic package originated – as a collection of tools we had assembled over time to make teaching and learning easier.

5. Comment your code.
   It's amazing what you can forget. The comment character in `R` is `#`. If you are working in RMarkdown or Rnw files, you can also include nicely formatted text to describe what you are doing and why.

The document *Minimal R for Stats* lists common commands and ideas for the course and is a must read.

**Course Structure**

The important ideas of statistics and probability will be taught from a computational few and supplemented with mathematics. Each lesson will have notes to read and then in class work and homework. The notes are based, loosely in some points, on the following two books:

1. Introductory Statistics with Randomization and Simulation, First Edition by Diez, Barr, and Cetinkaya-Rundel. This is an open source book and it is under the creative commons license, see http://www.openintro.org/perm/stat2nd_v1.txt.
2. Introduction to Probability and Statistics Using R by G. Jay Kerns

You can use these books as references for the course.

In addition, Lt Col Ken Horton developed a set of notes for the Fall 2019 offering. These notes are used and/or modified in this course.

**Course Philosophy**

A great deal of work by Danny Kaplan, Randy Prium, Ben Baumer, Nicholas Horton, and Mine Cetinkaya-Rundel, just to name a few, has influenced this course. In particular Danny and Randy have outlined the following guidance, which we have adopted for this course:

**Strategies**

1. Start right away.

Do something with `R` on day 1. Do something else on day 2. Have students do something by the end of week 1 at the latest.

2. Illustrate frequently.

Have `R` running every class period and use it as needed throughout the course so students can see what `R` does. Preview topics by showing before asking students to do things.

3. Teach `R` as a language. (But don't overdo it.)

There is a bit of syntax to learn – so teach it explicitly.

a. Emphasize that capitalization (and spelling) matter.

b. Explain carefully (and repeatedly) the syntax of functions.

Fortunately, the syntax is very straightforward. It consists of a function name followed by an opening parenthesis, followed by a comma-separated list of arguments (which may be named), followed by a closing parenthesis.

```
functionname ( name1=arg1, name2=arg2, ... )
```

Get students to think about what a function does and what it needs to know to do its job.

Generally, the function name indicates what the function does. The arguments provide the function with the necessary information to do the task at hand.

c. Every object in `R` has a type (class).

Ask frequently:

*What type of thing is this?*

Students need to understand the difference between a variable and a data frame and also that there are different kinds of variables (factor for categorical data and numeric for numerical data, for example).

Instructors and more advanced students will want to know about vector and list objects.

Give more details in higher level courses.

Upper level students should learn more about user-defined functions and language control structures such as loops and conditionals. Students in introductory courses don't need to know as much about the language.

4. "Less volume, more creativity." [Mike McCarthy, head coach, Green Bay Packers]

Use a few methods frequently and students will learn how to use them well, flexibly, even creatively. Focus on a small number of data types: numerical vectors, character strings, factors, and data frames. Choose functions that employ a similar framework and style to increase the ability of students to transfer knowledge from one situation to another.

5. Find a way to have computers available for tests.

It makes the test match the rest of the course and is a great motivator for students to learn `R`. It also changes what you can ask for and about on tests.

6. Rethink your course.

With ubiquitous computing, some things disappear from your course:

a. Reading statistical tables.

Does anyone still consult a table for values of `sin`, or `log`? Since we don't use them in our professional work, why would we teach this to students?

b. "Computational formulas".

Replace them with computation. Teach only the most intuitive formulas. Focus on how they lead to intuition and understanding, not computation.

c. (Almost all) hand calculations.

At the same time, other things become possible that were not before:

a. Large data sets

b. Beautiful plots

c. Simulations and methods based on randomization and resampling

d. Quick computations

e. Increased focus on concepts rather than calculations

Get your students to think that using the computer is just part of how statistics is done, rather than an addon.

7. Keep the message as simple as possible and keep the commands accordingly simple.

Particularly when doing graphics, beware of distracting students with the sometimes intricate details of beautifying for publication. If the default behavior is good enough, go with it.

8. Anticipate computationally challenged students, but be confident that you are leading them down the right path.

Some students pick up R very easily. In every course there will be a few students who struggle. To help them, focus on diagnosing what they don't know and how to help them "get it".

In our experience, the computer is often a fall guy for other things the student does not understand. Because the computer gives immediate feedback, it reveals these misunderstandings. For example, if students are confused about the distinctions among variables, statistics, and observational units, they will have a difficult time providing the correct information to a plotting function. The student may blame R, but that is not the primary source of the difficulty. If you can diagnose the true problem, you will improve their understanding of statistics and fix R difficulties simultaneously

Even students with a solid understanding of the statistical concepts will encounter R errors that they cannot eliminate. Tell students to copy and paste R code and error messages into email when they have trouble. When you reply, explain how the error message helped you diagnose their problem and help them generalize your solution to other situations.

**Tactics**

1. Introduce Graphics Early.

Introduce graphics very early, so that students see that they can get impressive output from simple commands Try to break away from their prior expectation that there is a "steep learning curve."

Accept the defaults – don't worry about the niceties (good labels, nice breaks on histograms, colors) to early. Let them become comfortable with the basic graphics commands and then play (make sure it feels like play!) with fancying things up.

Keep in mind that just because the graphs are easy to make on the computer doesn't mean your students understand how to read the graphs. Use examples that will help students develop good habits for visualizing data.

2. Introduce Sampling and Randomization Early.

Since sampling drives much of the logic of statistics, introduce the idea of a random sample very early, and have students construct their own random samples. The phenomenon of a sampling distribution can be introduced in an intuitive way, setting it up as a topic for later discussion and analysis

This course focuses on balancing mathematics with computation. In many cases we will use both approaches. Mathematical solutions offer insights and prepare you for further graduate studies. However, in practice the ability to use computational solutions is more important. Computational solution often have less assumptions and adjust to a wider variety of problems.

**Role in the Whole of Educational Experience**

Leo Breiman wrote an influencial paper about the two cultures of analysis. There is this contrast between models that allow interpretation and models that lead to predictive performance.[1] This has shaped our thinking on Math 377 and Math 378. Math 377 is the interpretative modeling with an emphasis on data collection and inference. We do include some exploratory analysis as well. Math 378 is our machine learning and predictive modeling class. Together these serve as our foundational analysis courses.

[1] Breiman, Leo. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). Statist. Sci. 16 (2001), no. 3, 199–231. doi:10.1214/ss/1009213726.

## Block 1 - Case Study

The case study is designed to introduce `R` and the general ideas in analysis.

Some students may have `R` on their machine, others will use the Rcloud version. They should have an account but be prepared to spend time getting them up and running.

### Objectives

1) Use R for basic analysis and visualization.

2) Compile a report using `knitr`.

### Tactics

1. We are doing both an exploratory and explanatory analysis. There is too much for students to do and they will get overwhelmed. Let's try to keep it simple and let them use. If they don't have `R` on their computer, then have them use Rcloud.

2. Emphasize the following when we use `R`:

a. What do I want `R` to do?

b. What information must I provide for `R` to do this?

Remember that we will use the `mosaic` package but the Data Camp course will emphasize the tidy verse.

3. Walk through code, live, and ask questions. In the end `knit` the document. Remember that all things are objects in `R` so a function is an object. But in the notes we have emphasized function by putting `()` at the end of the name.

4. Show them the RStudio interface. The file structure, the packages tab, etc. Show how to load packages. Open a new Markdown file.

5. Compile, `knit` your in class work to show them how. Just use HTML at first. They need the tinytex package if they want to make a pdf.

6. Have them start the application/homework if time allows.

## Data Basics

### Objectives

1) Define and use properly in context all new terminology to include but not limited to case, observational unit, variables, data frame, associated variables, independent, and discrete and continuous variables.

2) Identify and define the different types of variables.

3) From reading a study, explain the research question.

4) Create a scatterplot in `R` and determine the association of two numerical variables from the plot.

### Tactics

1) There are a great deal of new terms. The students should read the section, but pull up some of the data and ask them to explain the different data types.

2) The difference between numerical discrete and continuous is subtle. In analysis, I would probably treat population as numerical discrete even though it is counting type of data. Number of siblings is a little more obvious as numerical discrete. Spending versus spending per capita are different animals because we are creating more levels. For plotting and summarizing, we treat both the same way. However, in probability Poisson is different from the normal. In practice, all numeric data is finite with jumps. I would not get too pedantic about this.

3) The tidy data discussion is important because you have to think about how you want to analyze the data. What is an observation for your analysis purposes? It will be confusing to students at first but the more they work with data the better they will get.