**First E-mail to students**

1. Include
   1. Website
   2. Syllabus – links to installing R and RStudio – or link to webpage describing that.
   3. Slack
   4. **Blog Article – part of 1st homework**
2. **Ask students to show up early for the first class, if they have trouble installing it.**

**Lecture 01**

**Goal for first class is to get oriented/express expectations, and to get through enough material that they can complete the first assignment.**

1. Start with Introductions
   1. Name, Year, Program/Position
   2. What is your interest in this class / experience level?
   3. Fun Fact
2. Introduce ourselves
   1. Emphasize amateur status
3. Talk about class structure / syllabus
   1. Schedule make-up class for July 4th. Suggest July 1st or Wed 6th.
   2. Make sure people have R and RStudio installed.
   3. Slack sign-ups
   4. Post-its

**Principles**

* Trying to keep it light, conversational, interesting. Less of a reference manual, and hopefully following a logical train of thought.

**Personal Introduction**

* I (Paul) first started to learn programming in grade school. I enjoyed learning VisualBasic and C++, but I was intimidated by the competition in the class to finish coding first. After taking a number of speed tests, I realized that I could not solve the problems as quickly as some of my peers, so I thought it would be difficult to pursue as a career. So, despite a strong interest in computing and technology, I did not pursue computer science or programming at all in undergrad, and I didn’t try to learn programming again until many years later.
* In graduate school, I took a modeling class that required programming, and I tried to learn R, but didn't get around to really using it until I joined the BCB program.
* You are all probably starting from slightly different backgrounds, which means that for most of the class, your peers will be your resource for learning.
* Learning can be painful and rewarding.

**Introduction: The parts of RStudio and R**

* Console
* Variable / Object
* Workspace
* Script vs. Interactive
  + A set of commands that are run in order
* Parts

**Concepts (in order of introduction)**

**Scientific calculation**

Operations

+, \*, ^, e, log(),

Function

Argument

Inf

Reserved words

Warning

Error/Sanity Checking – later in the course

log2(), sqrt(), exp(), abs()

Variable

Assignment

<-

Strings

ls()

rm()

Closing parentheses (plus sign)

**Statistical Analysis**

**High-Quality Plots**

**Reproducible Research**

Commenting code

Writing scripts

**Data Types: vectors, matrices, arrays, lists, and data frames**

class()

str()

summary()

Syntax

Vectors, subsetting - c() = concatenate, list.files

Tab completion

Keyboard shortcuts

Vector addition

**Packages**

functions; data; where to find/get them; bioconductor/ github / cran

install.packages(); source, repos, etc.

Install all the necessary ones for this course: dplyr, tidyr, etc.

Environment variables

ls(), environment(), search()

Quit

q()

CTRL+D

**Getting Help**

help() / ?

StackOverflow

**Overview**

* R (vs. Python)
  + Chart
* Case studies

**What is R?**

* Developed by Robert Gentleman and Ross Ihaka in 1990’s, in New Zealand, for use in statistical computing

**Advantages of R**

* Open source
* Academic and professional community
* High-quality visualization capabilities
* Multi-dimensional data analysis
* There are many ways to do the same thing

**Disadvantages of R**

* Variety of quality
* There are many ways to do the same thing

**Notes from Peng**

**The S/R philosophy**

* There are only two kinds of languages: the ones people complain about and the ones nobody uses —Bjarne Stroustrup
* From *user* to *coder*

**R structure**

* Base R
* Base Packages
* Optional Packages
  + CRAN
  + Bioconductor
  + GitHub

**Limitations of R**

* RAM memory

**Principles**

**Resources**

**Homework**