STAT 540 – Computing in Statistics

- Introduces programming skills in two important statistical computer languages/packages.
- 30-40% R and 60-70% SAS
- Examples of Programming Skills:
  1. Importing Data from External Files
  2. Data Manipulation and Data Organization
  3. Summarizing Data
  4. Graphics (in R)
- Very little emphasis on data analysis.

Origins of R

- S language – developed by John Chambers of Bell Labs
- Commercial Implementation: S-plus
- Free Software Implementation: R (named in honor of founders Robert Gentleman and Ross Ihaka)
- Object-oriented language: In R, we work with "objects" and create new ones
- Each object has a "type" (vector, factor, matrix, function, data frame, etc.)
- Important to be aware of what type each object is.

Vector Objects in R

- **Vector**: String of elements *having the same mode*
- Can be numeric, character, logical
- R vectors are not row vectors nor column vectors
- Can store objects to your chosen name with the assignment operator: $<-$
- Often vectors will have *attributes* that you can examine (see examples)

Other Types of Objects in R

- **Factor**: Similar to a character vector; has multiple *levels*
- Can coerce object into a different mode, e.g., with `as.character()`
- **Matrix**: Array of elements (rows and columns), *all of same mode*
- Can extract individual elements, or rows, or columns of a matrix using square brackets (see examples)
- **Data Frame**: Similar to matrix, but columns can be *of different modes*

- **List**: string of objects with possibly many different modes/structures
- Often built-in R functions will return lists as their outputs.
- **Function**: A command in R that accepts input (arguments) and returns *one* object (which may be a list) as output
- Any intermediate calculations not saved
- Some arguments may be optional and have default values (see examples).
- Some functions do different things depending on what type of object is input (example: `diag()`)
- Other objects: Ordered objects, time series objects, arrays, etc.

Useful Commands in R

- Getting Help: `help`, `help.search`, `help.start`, `?objectname`
- Usually you want help information for functions; sometimes for built-in data sets
- Help files often describe a function's purpose, its arguments, and what it outputs.
- Often gives references to similar functions and provides examples of code.

Managing Objects in R

- R objects organized into a hierarchy of directories
- Use `search()` to show current search path directories
- `ls()` lists objects in current workspace
- `objects(3)` gives the objects in the 3rd element of `search()`
- Other useful commands: `attach()` to place data frames or lists in the search path, `detach()` to remove them from search path, `rm()` to remove individual objects from the workspace

Saving Work in R

- Can save current workspace when you quit R (saved as `.Rdata` file)
- Another option: just save all code you need in a Notepad file
- Can copy and paste code into R (or use the `source` command) whenever you want
- Several ways to get back to a saved workspace:
  1. Can open the saved `.Rdata` file
  2. Go the File menu, then "Load Workspace"
  3. Use the `attach` command
- Be careful of saving objects under identical names, and of keeping duplicate objects in two directories!
- It's also bad practice to give your own R objects the same name as built-in R functions.

Getting Data into R

- Built-in functions to enter vectors manually: `c()`, `seq()`, `rep`
- R can generate random values from common distributions, e.g., `rnorm`, `runif`, `rchisq`
- Can create matrices with the `matrix` function
- Matrix can be written by rows or by columns

Reading Data from an External File into R

- Use `read.table` for "nice" rectangular data files
- Data entries may be separated by spaces, tabs, or commas, etc.
- First line may or may not be a header (with column names)
- Important arguments for `read.table` function:
  1. `file`: specifies file name (either use *full path* or change to correct directory); use quotes!
  2. `header`: Is first row a header (T/F)? Note default settings.
  3. `sep`: What character separates data elements? Note default settings.
  4. `row.names` and `col.names`: Can provide specially; otherwise, will be generic names or will be taken from header.
  5. `as.is`: If `TRUE`, will prevent R from converting character columns to factors.
  6. `nrows`: Tells R the maximum number of rows to read
  7. `skip`: Tells R how many rows to skip before reading data (default is 0)

- The usual specification for missing data values in R is `NA`. Other markers for missing data can be specified with the `na.strings` option.
- It's sometimes easier to "clean up" text file before reading it into R.

More About Reading Data from an External File

- A raw data table can be copied and pasted to the keyboard by using `tempfile()` to create a temporary file and the `cat` command to fill it in.
- See `http://people.stat.sc.edu/hitchcock/recognitiondataR.tx` for an example
- For "messy", unwieldy data file, the `scan` command works better than `read.table`
- `scan` does well for data files in which columns are of many different types
- The `what` argument tells R which variables are which types
- The `multi.line` argument can specify whether data records spill onto more than one line.
- The `widths` options lets you specify the field widths for each data record.

Getting Results out of R

- The `write` function is a useful way to print a matrix (of results, typically) to an external text file.
- Need to use the transpose function `t()` when specifying the matrix name.
- Specifying the number of columns is critical, otherwise R defaults to 5 columns no matter what.
- The `write` function also can output data frames to a text file
- The `write.table` function works similarly
- Other useful functions: `dump`, `dput`, `sink`
- The `print` command will print any R object to the R screen
- Use `paste` with `print` when printing strings combining character and numeric elements