Topic 5: Basics of Dealing with Data in R

1. Data Types
2. Factors
3. Basic Use of Data Frames
4. Basic Functions to Manipulate Data
5. Other data structures (FYI)

1. Data Types

- Two basic types of variable (columns)
  - Numeric – quantitative measures (discrete or continuous)
  - Factors – qualitative or categorical information
- Numeric
  - If appropriate, can force R to make factor to number
    - `as.numeric()`
  - Different classes of numbers
    - `class()` shows wither integer or double etc... (not important in most cases)
- Factor
  - Often appropriate to change a number to a factor!
  - Datasets often have numbers to label different levels of a factor
  - Change number to factor: `as.factor()`

2. Factors

- First determine if R is considering variable as factor with `str()`
- Important because functions will run different if factor vs. numeric
  - Plotting
  - Modeling
  - Errors for some functions: e.g. R will not do `mean(factor_variable)`
- Use `as.factor()` to change numeric to factor & check again with str()
- See factor levels with `levels()`
- R often defaults to alpha-ordering of levels
  - Change order: (move 6th level to the first)

```
clover$Strain <- factor(clover$Strain,levels(clover$Strain)[c(6,1:5)])
```

3. Using Data Frame

- Referencing data frame
  - Consider name of data frame that is short and informative
  - Need to refer to data frame any time referencing variables (columns) of data
    - Can attach data frame (not recommended)
    - **Use $ operator (as in prior examples, `dataframename$variablename`)**
    - To specify particular variable that may share same name among multiple datasets
- Attaching data frame
  - Do not need $ operator to refer to variables (keeps references to variables shorter)
  - Should detach data frame
  - May create potential issues if working with multiple data frames

Using Data Frame continued

- Functions requiring data
  - May not need to specify variable with $
  - Often use variable names then specify data set
    - `functionname(variablename1, variablename2, data = dataframename)`

- Appropriately referencing data and variables in functions takes practice and becoming familiar with certain functions

4. Basics to Managing Data

- To create a new column in data frame
  - `dataframename$newcolumn <- somevalue`
  - New column will be added to last of columns of data frame
  - Use `str()` to verify new column was added as intended
- Example1: Double a particular numeric variable
  - `dataframename$newcolumn <- 2*dataframename$originalcolumn`
- Example2: Add natural log transformation of a numeric variable
  - `dataframename$logcolumn <- log(dataframename$originalcolumn)`
- Example3: Difference between two columns
  - `dataframename$diffcolumn = dataframename$columnA- dataframename$columnB`

Basics to Managing Data, sorting (FYI)

- Sort using order()

```
data(mtcars)
mtcars
mtcars[order(mtcar$cyl),]
```

- Sort from plyr package

```
library(plyr)
arrange(mtcars, cyl)
# to include column names
myCars <- cbind(vehicle=row.names(mtcars), mtcars)
arrange(myCars, cyl, mpg)  #sorts first by cyl then by mpg
# sort with displacement in descending order
arrange(myCars, cyl, desc(disp))
```

5. Other data structures (FYI)

- Vectors & Matrices are common objects to store data (but others exist)
- Vectors are a sequence of values
  - R actually has no scalar, just vectors of length one
  - Functions may treat vectors differently than a column in a data frame

```
Y <- c(5,4,3,2,1)  #Y is a vector
Y
sort(Y) #sort() applies to vectors unlike order() or arrange()
```

- Matrices store information by both a column and row
  - Can reference elements in two-dimensional indexing

```
B <- matrix(c(2, 4, 3, 1, 5, 7), nrow=3, ncol=2)
B
```