Homework 6

Statistical Computing, 36-350

Due Wednesday Oct 28, 2015

Your homework must be submitted in R Markdown format. We will not (indeed, cannot) grade homeworks in other formats. Your responses must be supported by both textual explanations and the code you generate to produce your result. (Just examining your various objects in the "Environment" section of R Studio is insufficient—you must use scripted commands.)

Flu in the US

We will practice data transformations of a data set consisting of aggregate influenza level measurements in Delaware, District of Columbia, Maryland, Pennsylvania, Virginia, and West Virginia. Using the following code, read in the data frame `flumat`.

```
load(url("http://www.stat.cmu.edu/~ryantibs/statcomp/homework/flu.Rdata"))
```

Call `head(flumat)` to examine the first few rows of `flumat`. This data frame contains three columns:

`wili`: weekly influenza occurrences ("wili" stands for weighted influenza like illness percentage) for the these six states, during the time period between 1997 and the first part of 2013. There are some missing values in the first few seasons, don't panic! This is normal.

`season`: flu is a seasonal epidemic, and the *seasons* are defined as week 21 of the current year through the following year's week 20. Season 1 consists of the commensurate weeks of 1997-1998, season 2 is that of 1998-1999, and so forth.

`week`: calendar week numbers of each observations.

1. Notice the seasons are out of order. Reorder the data in increasing order of season. Check that it worked, by producing a plot of the flu measurements over time, and comparing to http://www.stat.cmu.edu/~ryantibs/statcomp/homework/wiliplot.pdf, found on the course website. Label and format your plot legibly.

2. Say we are interested in flu occurrence in December (week 49 through 52). Extract these rows and call this data frame `flumat.12`. Display the first ten rows. Using `tapply()` or some *ply function of your choice, extract the mean December wili in each season, and plot them over time (line plot). Report the seasons with the highest three and lowest three average December flu measurements, and plot them as points in red and blue respectively in the same plot. What is the overall distribution of December averages? The mean and standard deviation of these averages?

3. `flumat` is currently arranged in increasing order of season. Rearrange `flumat` to be arranged by week number, while preserving the increasing order of seasons, and call this new data frame `flumat.by.week`. Display the first and last 10 rows.

4. `flumat` is currently in *long* format, which means that the data are stacked vertically by season. Sometimes it is convenient to transform such data into *wide* format with horizontal stacking. In our case, this will mean that each season's flu occurrences are stacked horizontally as columns – i.e. each column contains a season's wili values. Do this transformation and call it `flumat.wide`, and display the first ten rows using `head()`. Note: the resulting matrix should have 52 rows – the 1st row with 21st calendar week's measurements, the second row with that of week 22, and so forth. (Hint: probably the most straightforward way of doing this is to use a for loop, and fill in the wide matrix one row at a time, by week. A more streamlined, but also more advanced strategy, would be to use the `reshape()` function in R.)

Bonus: Already loaded in your computer is new partially observed flu occurrence for season 17 – `flumat.new` – that is formatted in the same way as `flumat`. Merge it with the wide-formatted `flumat.wide` using `merge()` while preserving the formatting details of `flumat.wide`. Display the first six rows of the new data frame. (Hint: You may want to make a new column in `flumat.wide` and use it to *align* the merge.)

Bug hunt

In this part of the homework you will debug several pieces of almost-functional code. Start by setting `eval=TRUE` for a block of code, then use debugging techniques developed in class to find any problems in that block. After correcting those problems, explain in words what changes you made and why. Your final submission should have the setting `eval=TRUE`, and everything should run.

5. Find the nearest zero of a function using the Newton-Raphson method [http://en.wikipedia.org/wiki/Newton's_method], coded in `zero.finder()`. Be sure that `zero.finder` provides useful error messages when it encounters a problem. Hint: add extra intermediate outputs to diagnose what the function is doing.

```
zero.finder <- function (fn, start, tol=1e-6, delta=1e-4) {
  pp <- start
  repeat {
    grad.fn <- (fn(pp+delta) - fn(pp-delta))/(2*delta)
    pp <- pp + fn(p)/grad.fn
    if (abs(fn(pp)) < tol) break
  }
  pp
}

zero.finder (function(x) {1 - x^2}, 1.5)  # should be 1
zero.finder (function(x) {1 - x^2}, 0.5)  # should be 1
zero.finder (function(x) {x^3 - 7*x^2 + 7*x + 15}, 3.5)  # should be 3
zero.finder (function(x) {x^3 - 7*x^2 + 7*x + 15}, -1.5)  # should be -1
zero.finder (function(x) {x^3 - 7*x^2 + 7*x + 15}, 4.5)  # should be 5
zero.finder (function(x) {1/(1-x^2)}, 1)  # what should happen here?
```

6. The Fibonacci sequence is better defined dynamically, and one attempt at this is in `dynamic.fibonacci()`. It should return the numbers corresponding to `1,1,2,3,5,8,13 ...` for those values, and NA for any disallowed values.

```
dynamic.fibonacci <- function (nn) {
  if ( !(is.integer(nn)) | (nn<1)){
    print ("Bad input")
    return (NA)
  }
  my.fib <- c(1,1)
  for (kk in 3:nn) my.fib[kk] <- my.fib[kk-1] + my.fib[kk-2]
  return my.fib[nn]
}
dynamic.fibonacci(7)   #Should be 13
dynamic.fibonacci(5)   #Should be 5
dynamic.fibonacci(1)   #Should be 1
dynamic.dibonacci(1.5) #Should be NA
dynamic.dibonacci(0)   #Should be NA
```

7. Returning to the flu data above, how well does the week 50 wili predict the week 51 wili? We'll extract the relevant data using two boolean vectors, then perform a simple regression. Does the regression coefficient make sense? A plot may help here.

```
flumat.copy = flumat                #We'll work on a copy of the data
week50 = (flumat.copy$week = 50)    #The first boolean vector
week51 = (flumat.copy$week = 51)    #The second boolean vector
week50wili = flumat.copy$wili[week50,]
week51wili = flumat.copy$wili[week51,]
flureg = lm(week51wili~week50wili)
```