Stand-alone R Function Example

Linda Wang

January 17, 2016

See detailed information about what this script does after the code.

```r
# Ranking hospitals in all states

rankall <- function(outcome, num = "best") {

    ## Read outcome data
    hospital_data <- read.csv("outcome-of-care-measures.csv", colClasses =
"character")

    ## Check that state and outcome are valid
    var_options <- c("HEART ATTACK","HEART FAILURE","PNEUMONIA")
    if (!toupper(outcome) %in% var_options){
      stop("invalid outcome")
    }
    else {
      col_idx <- 0
      if (toupper(outcome)==var_options[1]){
        col_idx <- 11
      }
      else if (toupper(outcome)==var_options[2]){
        col_idx <- 17
      }
      else {
        col_idx <- 23
      }
    }

    ## For each state, find the hospital of the given rank
    hospital_subset <-
cbind(hospital_data[,col_idx],hospital_data[,2],hospital_data[,7])
    hospital_subset <- hospital_subset[!hospital_subset[,1]=="Not
Available",]
    lvl <- as.factor(as.vector(hospital_subset[,3]))
    hospital_subset[,3] <- as.vector(lvl)
    hospital_subset <-
hospital_subset[order(as.vector(hospital_subset[,3])),]
    hospital_subset_sort <-
split(hospital_subset,as.vector(hospital_subset[,3]))

    rankings <- c()
    state_pick <- c()

    for (i in 1:length(names(hospital_subset_sort))) {
      state_group <-
as.data.frame(matrix(as.vector(as.data.frame(hospital_subset_sort[i])[,1]),nc
ol=3, byrow=FALSE))
      state_group <-
state_group[order(as.vector(as.numeric(as.character(state_group[,1]))),as.vec
tor(state_group[,2])),]
      if (toupper(num)=="BEST"){
        state_pick <-
c(as.vector(state_group[1,2]),as.vector(state_group[1,3]))
      }
      else if (toupper(num)=="WORST"){
        state_pick <-
c(as.vector(state_group[nrow(state_group),2]),as.vector(state_group[nrow(stat
e_group),3]))
      }
      else {
        if (as.numeric(num)>0){
          if (as.numeric(num)<=nrow(state_group)){
            state_pick <-
c(as.vector(state_group[as.numeric(num),2]),as.vector(state_group[as.numeric(
num),3]))
          }
          else {
            state_pick <- c(as.vector("<NA>"),as.vector(state_group[1,3]))
          }
        }
        else {
          stop ("invalid rank")
        }
      }
      rankings <- rbind(rankings,state_pick)
    }

    ## Return a data frame with the hospital names and the (abbreviated)
state name
    rownames(rankings) <- as.vector(rankings[,2])
    colnames(rankings) <- c("hospital","state")
    as.data.frame(rankings)
}
```

Introduction:

Download the file ProgAssignment3-data.zip file containing the data for Programming Assignment 3 from the Coursera web site. Unzip the file in a directory that will serve as your working directory. When you start up R make sure to change your working directory to the directory where you unzipped the data. The data for this assignment come from the Hospital Compare web site (http://hospitalcompare.hhs.gov) run by the U.S. Department of

Health and Human Services. The purpose of the web site is to provide data and information about the quality of care at over 4,000 Medicare-certified hospitals in the U.S. This dataset essentially covers all major U.S. hospitals. This dataset is used for a variety of purposes, including determining whether hospitals should be fined for not providing high quality care to patients (see http://goo.gl/jAXFX for some background on this particular topic).

The Hospital Compare web site contains a lot of data and we will only look at a small subset for this assignment. The zip file for this assignment contains three files
1) outcome-of-care-measures.csv: Contains information about 30-day mortality and readmission rates for heart attacks, heart failure, and pneumonia for over 4,000 hospitals.
2) hospital-data.csv: Contains information about each hospital.
3) Hospital_Revised_Flatfiles.pdf: Descriptions of the variables in each file (i.e the code book).

A description of the variables in each of the files is in the included PDF file named Hospital_Revised_Flatfiles.pdf. This document contains information about many other files that are not included with this programming assignment. You will want to focus on the variables for Number 19 ("Outcome of Care Measures.csv") and Number 11 ("Hospital Data.csv"). You may find it useful to print out this document (at least the pages for Tables 19 and 11) to have next to you while you work on this assignment. In particular, the numbers of the variables for each table indicate column indices in each table (i.e. "Hospital Name" is column 2 in the outcome-of-care-measures.csv file).

Ranking hospitals in all states:

Write a function called rankall that takes two arguments: an outcome name (outcome) and a hospital rank-ing (num). The function reads the outcome-of-care-measures.csv file and returns a 2-column data frame containing the hospital in each state that has the ranking specified in num. For example the function call rankall("heart attack", "best") would return a data frame containing the names of the hospitals that are the best in their respective states for 30-day heart attack death rates. The function should return a value for every state (some may be NA). The first column in the data frame is named hospital, which contains the hospital name, and the second column is named state, which contains the 2-character abbreviation for the state name. Hospitals that do not have data on a particular outcome should be excluded from the set of hospitals when deciding the rankings. Handling ties. The rankall function should handle ties in the 30-day mortality rates in the same way that the rankhospital function handles ties.

The function should use the following template. rankall <- function(outcome, num = "best"){

```
 ## Read outcome data
 ## Check that state and outcome are valid
 ## For each state, find the hospital of the given rank
 ## Return a data frame with the hospital names and the
 ## (abbreviated) state name
 }
```

NOTE: For the purpose of this part of the assignment (and for eficiency), your function should NOT call the rankhospital function from the previous section. The function should check the validity of its arguments. If an invalid outcome value is passed to rankall, the function should throw an error via the stop function with the exact message "invalid outcome". The num variable can take values "best", "worst", or an integer indicating the ranking (smaller numbers are better). If the number given by num is larger than the number of hospitals in that state, then the function should return NA.