Programming Assignment 3: Hospital Quality rankall

Ahmed Ezzat Afifi

Thursday, October 23, 2014

Read the function specification very carefully!

you get the idea we are not going to do something very new by now. except we will be ranking over all states not an exact one

and as rankhospital we should take care of invalid arguments

again, we are not going to-reinvent the wheel. this is the beauty of designing a good algorithm before starting to code :)

a clever trick we will be doing here

in R and python you can do things like this

```
#Printing the first 10 states
#In some languages like Java or C# or PHP this is called foreach
for(i in state.abb[1:10]){ #This is a bulit in vector in R that has the states abbreviates
  print(i)
}
```

```
## [1] "AL"
## [1] "AK"
## [1] "AZ"
## [1] "AR"
## [1] "CA"
## [1] "CO"
## [1] "CT"
## [1] "DE"
## [1] "FL"
## [1] "GA"
```

As opposed to this for loop. here we

```
#Here i is only a number and we are getting the data from state.abb by index
for(i in 1:10){
  print(state.abb[i])
}
```

```
## [1] "AL"
## [1] "AK"
## [1] "AZ"
## [1] "AR"
## [1] "CA"
## [1] "CO"
## [1] "CT"
## [1] "DE"
## [1] "FL"
## [1] "GA"
```

what if we want to use the both methods at the same time? sometimes “like in rankall” we want both the for iterator to have both the value and the position we can simply define a number by 1 outside the for loop, then increment it at the end of the loop

```
z <- 1
for(state in state.abb){
  print(state); print(z)
  z <- z + 1;
  #Now we can use data with both data[data$State == state]
  #And something like data[z, ]
}
```

```
## [1] "AL"
## [1] 1
## [1] "AK"
## [1] 2
## [1] "AZ"
## [1] 3
## [1] "AR"
## [1] 4
## [1] "CA"
## [1] 5
## [1] "CO"
## [1] 6
## [1] "CT"
## [1] 7
## [1] "DE"
## [1] 8
## [1] "FL"
## [1] 9
## [1] "GA"
## [1] 10
## [1] "HI"
## [1] 11
## [1] "ID"
```

```
## [1] 12
## [1] "IL"
## [1] 13
## [1] "IN"
## [1] 14
## [1] "IA"
## [1] 15
## [1] "KS"
## [1] 16
## [1] "KY"
## [1] 17
## [1] "LA"
## [1] 18
## [1] "ME"
## [1] 19
## [1] "MD"
## [1] 20
## [1] "MA"
## [1] 21
## [1] "MI"
## [1] 22
## [1] "MN"
## [1] 23
## [1] "MS"
## [1] 24
## [1] "MO"
## [1] 25
## [1] "MT"
## [1] 26
## [1] "NE"
## [1] 27
## [1] "NV"
## [1] 28
## [1] "NH"
## [1] 29
## [1] "NJ"
## [1] 30
## [1] "NM"
## [1] 31
## [1] "NY"
## [1] 32
## [1] "NC"
## [1] 33
## [1] "ND"
## [1] 34
## [1] "OH"
## [1] 35
```

```
## [1] "OK"
## [1] 36
## [1] "OR"
## [1] 37
## [1] "PA"
## [1] 38
## [1] "RI"
## [1] 39
## [1] "SC"
## [1] 40
## [1] "SD"
## [1] 41
## [1] "TN"
## [1] 42
## [1] "TX"
## [1] 43
## [1] "UT"
## [1] 44
## [1] "VT"
## [1] 45
## [1] "VA"
## [1] 46
## [1] "WA"
## [1] 47
## [1] "WV"
## [1] 48
## [1] "WI"
## [1] 49
## [1] "WY"
## [1] 50
```

Anyway, So our function goes like this

```
rankall <- function(outcome, num = "best") {
  #Read data as usual
  data <- read.csv("outcome-of-care-measures.csv", colClasses = "character")

  #Save or potential outcomes
  outcomevalues = c("heart attack", "heart failure", "pneumonia")



  if (!is.element(outcome, outcomevalues)) {
    stop("invalid outcome")
  }

  column <- c(11, 17, 23)[outcomevalues == outcome]
```

```
suppressWarnings(data[[column]] <- as.numeric(data[[column]]))

data <- data[!is.na(data[[column]]),]


hospitals <- c() #Initialize an emptey vector to save hospitals in
states <- c() #Same but for states

#Since we will not loop with an iterator
#We will loop with the value of state.abb
#So the state variable will be the state abbreviation
#This will make it easy to subset the data by state
#But we will need too an iterator
#To append our hospitals
#And states vectors
#So we will define a counter
#Just like we do with while loops if you are familiar with them
#And then increment it at the end of every loop iteration
#So now we can use both the easy way to subset data and the easy way to append our vectors

i <- 0
for (state in state.abb) {

  #Subset the data to only keep for the value of 'state' in the for loop
  #The easy way to subset data by state
  stateData <- data[data$State == state, ]

  # Sort by minimum column, and hospital name -- best one on top
  stateData <- stateData[order(stateData[column], stateData$Hospital.Name), ]

  # Check what to return
  if (num == "best") {
    #Return the top of the ordered data
    value <- stateData$Hospital.Name[1]
  }
  else if (num == "worst") {
    #return the last element of the ordered data
    value <- stateData$Hospital.Name[nrow(stateData)]
  }
  else if (num <= nrow(stateData)) {
    #Return the number of row by index
    value <- stateData$Hospital.Name[num]
  }
  else {
    value <- NA
  }
```

```
    #Append our hospitals data
    hospitals[i] <- value
    states[i] <- state
    i <- i+1 #Here we increment the i so we can keep track of it
  }

  df <- data.frame(hospital=hospitals, state=states)
  df
}
```