Project 2

Andrew Abisha Hu

8/16/2018

Instructions:

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

• 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. • hospital-data.csv: Contains information about each hospital.

• 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 (of Care Measures.csv“) and Number 11 (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. Name" is column 2 in the outcome-of-care-measures.csv file).

Part I

Write a function called best that take two arguments: the 2-character abbreviated name of a state and an outcome name. The function reads the outcome-of-care-measures.csv file and returns a character vector with the name of the hospital that has the best (i.e. lowest) 30-day mortality for the specified outcome in that state. The hospital name is the name provided in the Hospital.Name variable. The outcomes can be one of attack“, failure”, or “. Hospitals that do not have data on a particular outcome should be excluded from the set of hospitals when deciding the rankings. Handling ties. If there is a tie for the best hospital for a given outcome, then the hospital names should be sorted in alphabetical order and the first hospital in that set should be chosen (i.e. if hospitals ”, “, and ” are tied for best, then hospital " should be returned).

The function should use the following template.

```
best <- function(state, outcome) {
## Read outcome data
```

```
## Check that state and outcome are valid
## Return hospital name in that state with lowest 30-day death
## rate
}
```

The function should check the validity of its arguments. If an invalid state value is passed to best, the function should throw an error via the stop function with the exact message state“. If an invalid outcome value is passed to best, the function should throw an error via the stop function with the exact message outcome”.

```
library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.1

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

best <- function(state, outcome) {
        ## Read outcome data
        data <- read.csv("/users/andrewhu/Documents/GitHub/Coursera-
DataScienceSpecialization/02_RProgramming/oocm.csv",colClasses = "character")

        data<- select(data,c(2,7,11,17,23)) #subset for hospital, state,
heart attack, heart failure and pneumonia
        colnames(data) <- c("hospital", "state", "heart attack", "heart
failure", "pneumonia") #naming the columns

        # Check that state and outcome are valid
        if (!state %in% data[ ,"state"]){ stop('invalid state')}
        else if (!outcome %in% c("heart attack", "heart failure",
"pneumonia")) {stop('invalid outcome')} #for invalid
        else {
        state_logical <- which(data[, "state"] == state) #Create logical
indicator for having state
        df_state <- data[state_logical, ]    # subset data for having state
using the logical factor
        ot <- as.numeric(df_state[, eval(outcome)])#turn the outcome into
numeric
        min_eval <- min(ot, na.rm = TRUE) #assign the min value
        result  <- df_state[, "hospital"][which(ot == min_eval)]
        output  <- result[order(result)] }
return(output)
```

```
}
best("SC", "heart attack")

## Warning in best("SC", "heart attack"): NAs introduced by coercion

## [1] "MUSC MEDICAL CENTER"
```

Part II

Ranking hospitals by outcome in a state

Write a function called rankhospital that takes three arguments: the 2-character abbreviated name of a state (state), an outcome (outcome), and the ranking of a hospital in that state for that outcome (num). The function reads the outcome-of-care-measures.csv file and returns a character vector with the name of the hospital that has the ranking specified by the num argument. For example, the call rankhospital("MD", "heart failure", 5) would return a character vector containing the name of the hospital with the 5th lowest 30-day death rate for heart failure. The num argument can take values "", "", 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. Hospitals that do not have data on a particular outcome should be excluded from the set of hospitals when deciding the rankings.

The function should use the following template.

```
rankhospital <- function(state, outcome, num = "best") {
## Read outcome data
## Check that state and outcome are valid
## Return hospital name in that state with the given rank
## 30-day death rate
}
```

The function should check the validity of its arguments. If an invalid state value is passed to rankhospital, the function should throw an error via the stop function with the exact message state". If an invalid outcome value is passed to rankhospital, the function should throw an error via the stop function with the exact message outcome".

```
library(dplyr)
rankhospital <- function(state, outcome, rank) {
        ## Read outcome data
        data <- read.csv("/users/andrewhu/Documents/GitHub/Coursera-
DataScienceSpecialization/02_RProgramming/oocm.csv",colClasses = "character")

        data<- select(data,c(2,7,11,17,23)) #subset for hospital, state,
heart attack, heart failure and pneumonia
        colnames(data) <- c("hospital", "state", "heart attack", "heart
failure", "pneumonia")
        # Check that state and outcome are valid

        if (!state %in% data[ ,"state"])
```

```
                { stop('invalid state')
        } else if (!outcome %in% c("heart attack", "heart failure",
"pneumonia"))
                {stop('invalid outcome')
        } else if (is.numeric(rank)){

                state_logical <- which(data[, "state"] == state) #logical
variable for having state

                df_state <- data[state_logical, ]    # subset data for having
state

                df_state[,eval(outcome)] <- as.numeric(df_state[,
eval(outcome)])

                df_state <- df_state[order(df_state[, eval(outcome)],
df_state[, "hospital"]), ]

                output  <- df_state[,"hospital"][rank]


        }
          else if (!is.numeric(rank)){
                if (rank == "best") {

                output<- best(state,outcome)

        }       else if (rank == "worst") {

                state_logical <- which(data[, "state"]==state)

                df_state <-data[state_logical,]

                df_state[, eval(outcome)] <-
as.numeric(df_state[,eval(outcome)]) #turn the outcome value into numeric

                df_state <- df_state[order(df_state[,eval(outcome)],
df_state[,"hospital"], decreasing =T),] #order the hospital: if some hospital
have same values, use alphabetically order

                output <- df_state[,"hospital"][1] #only return the first one
        }
                else
                {stop("invalid rank")
        }


                }
```

```
return(output)
}

rankhospital("NC", "heart attack", "worst")

## Warning in rankhospital("NC", "heart attack", "worst"): NAs introduced by
## coercion

## [1] "WAYNE MEMORIAL HOSPITAL"
```

Part 3: 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.

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

        #test

        outcome= "heart attack"
        ## Read outcome data
        data <- read.csv("/users/andrewhu/Documents/GitHub/Coursera-
DataScienceSpecialization/02_RProgramming/oocm.csv",colClasses = "character")

        data<- select(data,c(2,7,11,17,23))
        colnames(data)<- c("hospital", "state", "heart attack", "heart
failure", "pneumonia")

        disease <- c("heart attack", "heart failure", "pneumonia")


        if (!outcome %in% disease ) {stop ("invalid outcome")}
        ##subsetting data
        if (outcome == "heart attack") {
                data<- data[c("hospital", "state", "heart attack")]
        }
        else if (outcome == "heart failure") {
                data<- data[c("hospital", "state","heart failure")]
        }
```

```
        else if (outcome == "pneumonia") {
                data<- data[c("hospital","state", "pneumonia")]
        }

       colnames(data)[3] <- "disease.rate" #assign a simple name for column 3

        data$disease.rate <-as.numeric(as.character(data$disease.rate))

        data$hospital<- as.character(data$hospital) #transform for order
using

        State_list <- as.character((data$state))

        State_list <- State_list[order(State_list)] #order by state name

        final_df <- data.frame() #create a blank df first

        for (i in seq_len(length(State_list))){


                State_df <- subset(data, state == State_list[i])

                State_df <- State_df[order(State_df$disease.rate,
State_df$hospital),]

        N <- sum(!is.na(State_df$disease.rate))
        if (num =="best") { num <- 1}
        else if (num =="worst") {num <- N}
        else{}


        hos <- State_df[num, "hospital"]
        df1 <- data.frame(hos, State_list[i])
        colnames(df1) <- c("hospital", "state")
        result_df <- rbind(final_df, df1)
        }

        return(result_df)
}

rankall("heart attack", "best")

## Warning in rankall("heart attack", "best"): NAs introduced by coercion

##                 hospital state
## 1 WYOMING MEDICAL CENTER    WY
```

** Reference**

[]https://github.com/DanieleP/PA3-tutorial

[]https://rpubs.com/fhlgood/rw3a3

[]https://www.cnblogs.com/yifeili/p/5437384.html