1 Week 3 Programming Assignment

1.1 Ranking hospitals in all states

- Write a function called `rankall()` that takes TWO (2) arguments: (a) an outcome name (**outcome**); and (b) a hospital ranking (**num**).
- The function reads the outcome-of-care-measures.csv file and returns a TWO(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 THIRTY(30)-day heart attack death rates.
- The function should return a value for EVERY state (some may be NA). The FIRST (1st) column in the data frame is named hospital, which contains the hospital name, and the SECOND (2nd) column is named state, which contains the TWO(2)-character abbreviation for the state name. The function should use the following template.

```
> rankall <-  function(outcome, num = "best") {
## Read outcome data
## For each state, find the hospital of the given rank
## Return a data frame with the hospital names and
## the (abbreviated) state name
  }
```

- (**Missing Values**) Hospitals that do NOT have data on a particular outcome should be excluded from the set of hospitals when deciding the rankings.
- (**Alphabetical Ties**) If there is MORE THAN ONE (1) hospital for a given ranking, then the hospital names should be sorted in alphabetical order and the FIRST (1st) hospital in that set should be returned (i.e. if hospitals b, c, and f are tied for a given rank, then hospital b should be returned).
- ***NOTE***: For the purpose of this part of the assignment (and for efficiency), your function should NOT call the `rankhospital()` function from the previous section.

- (**Valid Inputs**) 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.
- Save your code for this function to a file named `rankall.R`. To run the test script for this part, make sure your working directory has the file `rankall.R` in it.

1.2 Getting Started

- As with the previous programs, we can reduce the data set to five specific columns.
- We will also check for valid inputs for **outcome**.
- One key difference is that we will not be subsetting by **state**.

1.3 Output Data frame

To start off - we will construct empty vectors to contain the data as we go. We will later combine them into an output data frame.

RankHosp will be a a container for the selected hospitals from each state.

```
RankHosp=character()
```

1.4 Checking Each State

We will use a for loop to go through each state and find the required hospital. For each iteration, we will subset by state (and only use complete cases)

```
for (state in StateList)
   {
   Hosp2 <- Hosp[Hosp$State==state,]
   Hosp2 <- Hosp2[complete.cases(Hosp2),]

   ......


   }
```

- For each state we will select the ranked hospital. This is very similar to rankhospital.
- However - when we are selecting the ***worst*** hospitals, the location of the worst hospital will change from state to state. We have to be able to reset this for each state.
- We will save the num value, and use instead use a temporary variable that can be re-set with the saved num value at each step.
- We will use the same temporary variable for other two cases for the sake of simplicity, although it is not necessary then.

```
num.temp=num
if (num == "best") {num.temp = 1}
if (num == "worst") {num.temp = nrow(Hosp2)}
```

Now we can order the ***Hosp*** data frame, select the hospital name, and append it to the ***RankHosp*** charactered vector.

```
for (state in StateList)
   {
   #subset by state
   Hosp2 <- Hosp[Hosp$State==state,]
   Hosp2 <- Hosp2[complete.cases(Hosp2),]

   num.temp=num
   if (num == "best") {num.temp = 1}
   if (num == "worst") {num.temp = nrow(Hosp2)}

   OrderedHosp <- Hosp2[order(Hosp2[,3],Hosp2[,1]), ]

   RankHosp=c(RankHosp,OrderedHosp[num.temp,]$Hosp.Name)

   }
```

1.5 Output

We can construct a data frame (RankFrame) ,consisting of the set of hospital names and corresponding states.

```
RankFrame=data.frame(hospital=RankHosp,state=StateList)
return(RankFrame)
```

Putting it all together

```
rankall <- function(outcome, num = "best") {
Hosp<-read.csv("outcome-of-care-measures.csv",
    colClasses = "character")


#-----------------------------------------------------#
#Part 1 : Inputs

# Data set is imported as a data frame called "Hosp"
# We will immediately discard every column
# except columns 2,7,11,17 and 23

Hosp <- Hosp[ , c(2,7,11,17,23)]

# [11] "Hospital.30.Day.Death..Mortality..Rates.from.Heart.Attack"
# [17] "Hospital.30.Day.Death..Mortality..Rates.from.Heart.Failure"
# [23] "Hospital.30.Day.Death..Mortality..Rates.from.Pneumonia"

# We will also give the data frame more manageable column names
# Use capital letters for the sake of clarity

names(Hosp) <- c("Hosp.Name", "State", "Heart.At",
   "Heart.Fa","Pneum")

#-----------------------------------------------------#
#Part 2 : Check that inputed value is OK


  # construct a set of valid outcome names

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

  ## Check that outcome is valid
  if(!is.element(outcome,ValidOutcomes)){
    stop("invalid outcome")
  }


#-----------------------------------------------------#
#Part 3 : Convert data to numeric
# Suppress Warnings
```

```
suppressWarnings(Hosp[,3] <- as.numeric(Hosp[,3]))
suppressWarnings(Hosp[,4] <- as.numeric(Hosp[,4]))
suppressWarnings(Hosp[,5] <- as.numeric(Hosp[,5]))

#-----------------------------------------------------#
#Part 4 : Generate a list of states
StatesList <- as.character(unique(Hosp[,2]))

StateList = sort(StateList)


#-----------------------------------------------------#
#Part 5 : Built a temporary data frame called outcome.df

# Function is a simple "look-up table"
# For specified input, can find required column number

outcome.df <- data.frame(
   InputtedOutcome=c("heart attack","heart failure","pneumonia"),
   UseCol=c(3,4,5))


VarCol <- outcome.df[outcome.df$InputtedOutcome==outcome,]$UseCol

#-----------------------------------------------------#
#Part 6 : Subset data set by outcome

# We will use the column selected by VarCol, and also column 2
# Column 1 is the name of the hospital
# Column 2 is the name of the state

UseCols <- c(1,2,VarCol)

Hosp <- Hosp[,UseCols]

#-----------------------------------------------------#
# Part 7
# Set up two empty character vectors
RankHosp=character()

#-----------------------------------------------------#
# Part 8
#Populate the character vectors from Part 7
for (state in StateList)
```

```
   {
   Hosp2 <- Hosp[Hosp$State==state,]
   Hosp2 <- Hosp2[complete.cases(Hosp2),]

   num.temp=num
   if (num == "best") {num.temp = 1}
   if (num == "worst") {num.temp = nrow(Hosp2)}

   OrderedHosp <- Hosp2[order(Hosp2[,3],Hosp2[,1]), ]

   RankHosp=c(RankHosp,OrderedHosp[num.temp,]$Hosp.Name)


   }

#-----------------------------------------------------#
# Part 9
# Construct a data frame from character vectors
RankFrame=data.frame(hospital=RankHosp,state=StateList)
return(RankFrame)
}
```