Coursera - Computing For Data Analysis Week 3

October 11, 2013

1 Week 3 Programming Assignment

1.1 Some Useful R Commands

1.1.1 The stop() command

The `stop()` command stops execution of the current expression and executes an error action.

1.1.2 Set Theory Commands

Set theory commands are very useful in simplifying code. The `is.element()` command asks if a specified value is an element in a specified set. If so, the function returns "TRUE", otherwise "FALSE".

```
x <- c(2,4,8)
y <- 1:6
is.element(x,y)
!is.element(x,y)
x %in % y
```

```
> x <- c(2,4,8)
> y <- 1:6
> is.element(x,y)
[1]  TRUE  TRUE FALSE
> !is.element(x,y)
[1] FALSE FALSE  TRUE
>
> x %in% y
[1]  TRUE  TRUE FALSE
```

1.1.3 The `suppresswarnings()` command

1.1.4 The `order()` command

Suppose we want to sort a data frame by multiple columns in R. For example, with the data frame below we would like to sort by column z then by column b :

```
DF <- data.frame(b = factor(c("Hi", "Med", "Hi", "Low"),
      levels = c("Low", "Med", "Hi"), ordered = TRUE),
      x = c("A", "D", "A", "C"), y = c(8, 3, 9, 9),
      z = c(1, 1, 1, 2))
```

```
DF
    b x y z
1  Hi A 8 1
2 Med D 3 1
3  Hi A 9 1
4 Low C 9 2
```

```
DF[with(DF, order(z, b)), ]
```

```
> DF[with(DF, order(z, b)), ]
    b x y z
2 Med D 3 1
1  Hi A 8 1
3  Hi A 9 1
4 Low C 9 2
```

1.1.5 The `which()` and `which.min()` command

```
x <- c(11,4,12,10,4,7)
names(x) = letters[5:1]
which.min(x)

which(x == min(x))
which(x == max(x))
```

```
> x <- c(11,4,12,10,4,7)
> names(x) = letters[5:1]
> which.min(x)
d
2
>
> which(x == min(x))
d a
2 5
> which(x == max(x))
c
3

> Mins = which(x == min(x))
> sort(Mins)
d a
2 5
> sort(names(Mins))
[1] "a" "d"
> sort(names(Mins))[1]
[1] "a"
>
```

1.2 Inputs to the functions

The variable “state” accepts a 2 letter state abbreviation. For example “NY” is New York and “CA” is California. Some code must be written to account for an input piece of text that does not correspond to a state (e.g “XX”).
The variable “outcome” accepts three possible inputs:

- pneumonia
- heart attack
- heart failure

1.3 Hospitals Data Set

- The data for this assignment comes 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).

```
> Hosp<-read.csv("outcome-of-care-measures.csv")
>
> dim(Hosp)
[1] 4706   46
> nrow(Hosp)
[1] 4706
> ncol(Hosp)
[1] 46
>
> names(Hosp)[1:10]
 [1] "Provider.Number" "Hospital.Name"   "Address.1"
 [4] "Address.2"       "Address.3"       "City"
 [7] "State"           "ZIP.Code"        "County.Name"
[10] "Phone.Number"
>
```

As an aside, the three columns we will be using the 11th, the 17th and the 23rd.

```
> names(Hosp)[c(11,17,23)]
[1] "Hospital.30.Day.Death..Mortality..Rates.from.Heart.Attack"
[2] "Hospital.30.Day.Death..Mortality..Rates.from.Heart.Failure"
[3] "Hospital.30.Day.Death..Mortality..Rates.from.Pneumonia"
```

The variables we will use contained in the following vector. We can subset our dataset just for these columns.

```
UseVars=c(2,7,10,11,17,23)

Hosp <- Hosp[ ,UseVars]
```

1.4 List of States Abbreviations

Let us look at the states (includes some US territories such as Guam, Puerto Rico and the US Virgin Islands). Using the `str()` command, we can see this variable is structured as a **factor**.

```
> summary(Hosp[,7])
 AK  AL  AR  AZ  CA  CO  CT  DC  DE  FL  GA  GU  HI  IA  ID  IL  IN  KS
 17  98  77  77 341  72  32   8   6 180 132   1  19 109  30 179 124 118
 KY  LA  MA  MD  ME  MI  MN  MO  MS  MT  NC  ND  NE  NH  NJ  NM  NV  NY
 96 114  68  45  37 134 133 108  83  54 112  36  90  26  65  40  28 185
 OH  OK  OR  PA  PR  RI  SC  SD  TN  TX  UT  VA  VI  VT  WA  WI  WV  WY
170 126  59 175  51  12  63  48 116 370  42  87   2  15  88 125  54  29
>
> str(Hosp[,7])
 Factor w/ 54 levels "AK","AL","AR",..: 2 2 2 2 2 2 2 2 2 2 ..
```

To generate a list of state names (as in abbreviations) can be generated by using the `unique()` and `as.character()` functions.

```
> as.character(unique(Hosp[,2]))
 [1] "AL" "AK" "AZ" "AR" "CA" "CO" "CT" "DE" "DC" "FL" "GA"
[12] "HI" "ID" "IL" "IN" "IA" "KS" "KY" "LA" "ME" "MD" "MA"
[23] "MI" "MN" "MS" "MO" "MT" "NE" "NV" "NH" "NJ" "NM" "NY"
[34] "NC" "ND" "OH" "OK" "OR" "PA" "PR" "RI" "SC" "SD" "TN"
[45] "TX" "UT" "VT" "VI" "VA" "WA" "WV" "WI" "WY" "GU"
```

We can then sort them using the `sort()` command.

```
StateList <- sort(as.character(unique(Hosp[,7])))
```

```
> StateList <- sort(as.character(unique(Hosp[,7])))
> is.element("XX",StateList)
[1] FALSE
> !is.element("XX",StateList)
[1] TRUE
>
```

1.5 R code to check that inputs are valid

```
 # Generate a list of valid state abbreviations from data.

ValidStates <- as.character(unique(Hosp[,2]))

  # construct a set of valid outcome names

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


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

1.6 Subset data set by state

Suppose we wish to subset to Rhode Island “RI”

```
Hosp<-read.csv("outcome-of-care-measures.csv",
 colClasses = "character")

UseVars=c(2,7,11,17,23)

Hosp <- Hosp[ ,UseVars]

Hosp <- Hosp[ Hosp[,2]=="RI",]
```

Let us use easier column names. Not necessary, but helpful.

```
# 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")
```

```
> Hosp
                                Hospital.Name State  Heart.Attack Heart.Failure     Pneumonia
3655          MEMORIAL HOSPITAL OF RHODE ISLAND    RI          14.9          12.8          12.8
3656              ROGER WILLIAMS MEDICAL CENTER    RI          16.3          10.8          11.6
3657            ST JOSEPH HEALTH SERVICES OF RI    RI          15.5          11.9          14.6
3658                           NEWPORT HOSPITAL    RI          15.6          13.7          10.0
3659                      RHODE ISLAND HOSPITAL    RI          13.9          10.8          11.3
3660                  SOUTH COUNTY HOSPITAL INC    RI          14.7          15.6          14.9
3661              KENT COUNTY MEMORIAL HOSPITAL    RI          14.4          11.3          10.4
3662 WOMEN AND INFANTS HOSPITAL OF RHODE ISLAND    RI Not Available Not Available Not Available
3663              LANDMARK MEDICAL CENTER,  INC    RI          14.2          12.9          14.8
3664                            MIRIAM HOSPITAL    RI          11.9          11.6          11.9
3665                          WESTERLY HOSPITAL    RI          17.7          10.1          12.5
3666               PROVIDENCE VA MEDICAL CENTER    RI          15.3          11.7          14.0
```

1.7 Programming Task : Finding the best hospital in a state

- 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 "heart attack", "heart failure", or "pneumonia".
- 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.

```
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
}
```

1.8 Lookup Tables

Next we will built a temporary data frame called **outcome.df**, which is a simple "look-up table" For specified input, we can find required column number

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


#For given input for "outcome" return the column number
# > outcome.df
#   InputtedOutcome UseCol

# 1    heart attack      3
# 2   heart failure      4
# 3       pneumonia      5
#
# Pick outcome.df$Cols when outcome.df$InputtedOutcome
#   is equal to "outcome"
# save the column number as VarCol

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

1.9 More Subsetting

Subset data set by selected state and outcome.

```
# Subset by State
Hosp = Hosp[Hosp$State==state,]
```

Next we will subset data set by outcome

```
# We will use the column selected by VarCol, and also column 1
# Column 1 is the name of the hospital
# N.B. UseCols is different from the variable UseCol, which we used previously

UseCols <- c(1,VarCol)

Hosp <- Hosp[,UseCols]
```

Inspecting the data set, we would see that the second column is actually character data. We will transform it to numeric data. We will use the `suppressWarnings()` command to hide warnings. We will reduce the data to complete cases only, using the `complete.cases()` command.

```
suppressWarnings(Hosp[,2] <- as.numeric(Hosp[,2]))
Hosp <- Hosp[complete.cases(Hosp),]
```

1.10 Find the case with the smallest value

In the first instance, we would use the `which.min()` function to determine which case contains the smallest value. This function would only return the first case of the minima, when there is a tie.

a better approach is to use the following code. This will return all the minima.

```
which(Hosp[,2] == min(Hosp[,2])
```

The best hospital is the hospital that corresponds to this case. Return the name only!

```
BestHosp <- Hosp[which(Hosp[,2] == min(Hosp[,2]) ), ]
```

Tie breaker - in the case of two hospitals being selected

```
BestHosp <- Hosp[which(Hosp[,2] == min(Hosp[,2]) ), ]
return(sort(BestHosp$Hosp.Name)[1])
```

(N.B. Order function is better approach - amend.)

2 Rank Hospitals

- 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

  le 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 "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.
- Hospitals that do not have data on a particular outcome should be excluded from the set of hospitals when deciding the rankings.

2.1 Key points

- We will be able to use a lot of code from the previous exercise. The key difference is selecting which row at the end.
- There is an additional input - num. It can take either a numeric value or the character values “best” or “worst”. There is no requirement to do validity checking for this exercise.
- The last two points are already considered.

2.2 Following on from `best.R`

At the end of the last exercise we had subsetted data frame Hosp by state and outcome. We considered only complete cases.

We are given an input **num**

- If the input for **num** is numeric, we leave it alone.
- If the input for **num** is “best” - then we rewrite it with a numeric value of 1
- If the input for **num** is “worst” - then we rewrite it with a numeric value of the number of rows of the data frame Hosp. (again - after we reduced it to complete cases only)

```
Hosp <- Hosp[complete.cases(Hosp),]

if (num == "best") {num = 1}
if (num == "worst") {num = nrow(Hosp)}
```

- We can use the order function to reorder the Hosp data frame in ascending order of the second column (i.e. the outcome column).
- To make sure that hospitals with tied ranks are in alphbetical order, rather than case order, we will then also order the data frame by name (column 1).
- We can then select the **num** row of the ordered data frame and return the hospital name.

```
OrderedHosp <- Hosp[order(Hosp[,2] ,Hosp[,1] ), ]

return (OrderedHosp[num,]$Hosp.Name)
```