

```

> # 1 point
> # Question 1: Create a variable named "myName" and assign to have a value of your
> # preferred name. Create a variable named "myEmail" and assign it to have a value
> # of your email.
> myName <- "Arinjay Jain"
> myEmail <- "ajain80@hawk.iit.edu"
> # Question 2: Create a vector of integers from 99 to 10000 (inclusive). Assign
> # the variable myVector. Randomly reorder that vector.
> # Write your own functions to sum, calculate the min value, the max value and the media
n value.
> # You do not need to implement your own sorting algorithms.
> # Return the sum, min, max, and median of this vector and assign it below.
> #
> # Note: in practice, you should usually use the predefined functions that R provides to
> # compute summary statistics. However, we can use this as an opportunity to practice ou
r R
> # while having an easy way to check for mistakes by comparing our function output with
the
> # default R function output.
> myVector <- c(sample(99:10000))
> mySumFunc <- sum(myVector)
> mySumFunc
[1] 50000149
> myMinFunc <- min(myVector)
> myMinFunc
[1] 99
> myMaxFunc <- max(myVector)
> myMaxFunc
[1] 10000
> myMedianFunc <- median(myVector)
> myMedianFunc
[1] 5049.5

> # 1 point
> # Question 3: Write a function that accepts a number as an input returns
> # TRUE if that number is divisible by 127 FALSE if that number is not divisible
> # by 127. For example, divis(127*5) should return TRUE and divis(80)
> # should return FALSE. Hint: %% is the modulo operator in R.
> divis <- function(inputnum){
+   if (inputnum %% 127 == 0) return(TRUE)
+   return(FALSE)
+ }
> num<-readline("Enter your number:" )
Enter your number:12
> divis(as.numeric(num))
[1] FALSE

# 1 point
> # Question 4: Using the function you wrote for Question 3 and the vector you
> # defined in Question 2, determine how many integers between 100 and 10000 are
> # divisible by 127. Assign it to the variable below.
> countDivis <- function(myVector){
+   count<-0
+   for (i in myVector) {
+     if (divis(i)) count = count+1
+   }
+   return(count)
+ }
> countDivis(myVector)
[1] 78

```

```

> # 1 point
> # Question 5: Using the vector of names below, write code to return the 9th
> # last name in the vector.
> names <- c("Kermit Chacko",
+           "Eleonore Chien",
+           "Genny Layne",
+           "Willene Chausse",
+           "Taylor Lyttle",
+           "Tillie Vowell",
+           "Carlyn Tisdale",
+           "Antione Roddy",
+           "Zula Lapp",
+           "Delphia Strandberg",
+           "Barry Brake",
+           "Warren Hitchings",
+           "Krista Alto",
+           "Stephani Kempf",
+           "Sebastian Esper",
+           "Mariela Hibner",
+           "Torrie Kyler")
> ninthLastName <- c(tail(names, n=9)[1])
> value <- "L"
> countL<-0
> for (k in 1:length(names)){
+   if (grepl(value,names[k])){ countL = countL+1}
+ }
> countLastNameStartsWithL <- countL
> ninthLastName
[1] "Zula Lapp"
> countLastNameStartsWithL
[1] 3

```

```

> # 1 point
> # Question 7: Using the vector "names" from Question 5, write code to create a
> # list that allows the user to input a first name and retrieve the last name.
> # For example, nameMap["Krista"] should return "Alto".
> nameMap <- function(firstname){
+   for (k in 1:length(names)){
+     if (grepl(firstname,names[k])){
+       name<-unlist(strsplit(names[k]," "))
+       return(name[2])
+     }
+   }
+ }
> fname<-readline("Enter User First Name:")
Enter User First Name:Zula
> print(nameMap(fname))
[1] "Lapp"

```

```

> # 2 points
> # Question 8: Load in the "Adult" data set from the UCI Machine Learning
> # Repository. http://archive.ics.uci.edu/ml/datasets/Adult
> # Load this into a dataframe. Rename the variables to be the proper names
> # listed on the website. Name the income attribute (">50K", "<=50K") to be
> # incomeLevel
> datafram<-adult
> names(datafram)[1]<-"Age"
> names(datafram)[2]<-"Workclass"
> names(datafram)[3]<-"Final Weight"
> names(datafram)[4]<-"Education"
> names(datafram)[5]<-"Education-Num"
> names(datafram)[6]<-"Marital-Status"
> names(datafram)[7]<-"Occupation"
> names(datafram)[8]<-"Relationship"
> names(datafram)[9]<-"Race"

```

```

> names(datafram)[10]<-"Sex"
> names(datafram)[11]<-"Capital-Gain"
> names(datafram)[12]<-"Capital-Loss"
> names(datafram)[13]<-"HoursPerWeek"
> names(datafram)[14]<-"Native-Country"
> names(datafram)[15]<-"IncomeLevel"
> # 2 points
> # Question 9: Create a new variable called workSector. Label all government
> # employees as "government", all self-employed employees as "selfEmployed",
> # all Private employees as "Private" and everyone else as "Other".
> # Enter the number of government employees in the text here, as well as showing the code below:
> countGovEmp<-0
> for (i in 1:length(datafram$workclass)){
+   if (grepl("gov",datafram$workclass[i])){
+     datafram$worksector[i]<-"Government"
+     countGovEmp<-countGovEmp+1
+   }else if (grepl("Self",datafram$workclass[i])){
+     datafram$worksector[i]<-"SelfEmployed"
+   }else if (grepl("Private",datafram$workclass[i])){
+     datafram$worksector[i]<-"Private"
+   }else {
+     datafram$worksector[i]<-"Other"
+   }
+ }
> print(countGovEmp)
[1] 4351

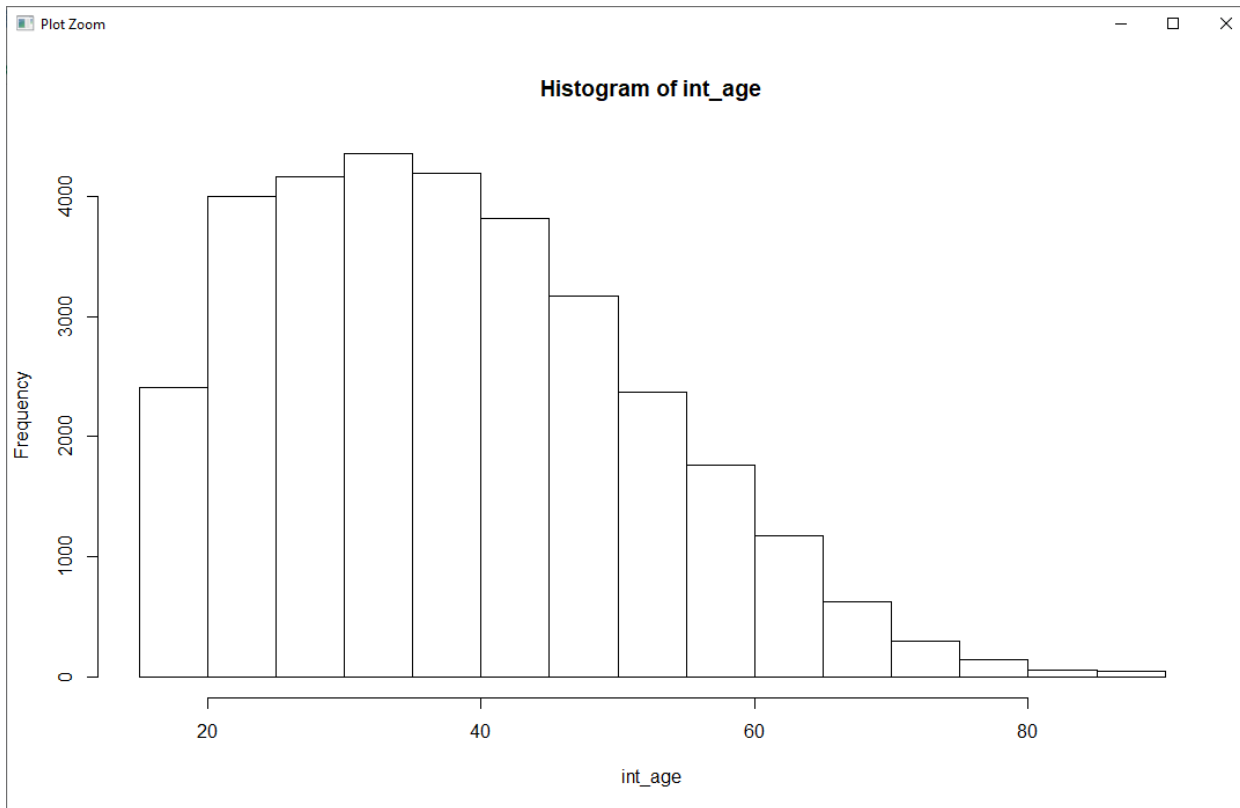
```

```

# 2 points
> # Question 10: Create a histogram of the 'age'. Hint: You may need to convert
> # age to be numeric first. Save this histogram and include it with your
> # submission
> Age<-as.numeric(datafram$Age)
> hist(Age)

```

Question 10:-



```
> # 2 points
> # Question 11: Determine the top 3 occupations with the highest average hours-per-week
> # Hint: One way to do this is to use tapply
> # List the occupations in the comments, as well as showing the code you used to determine that.
> higherOccupation<-sort(tapply(datafram$HoursPerWeek,datafram$Occupation,mean), decreasing = TRUE)
> top3highOcc<-head(higherOccupation,3)
> print(top3highOcc)
Farming-fishing      Exec-managerial      Transport-moving
      46.98994              44.98770              44.65623

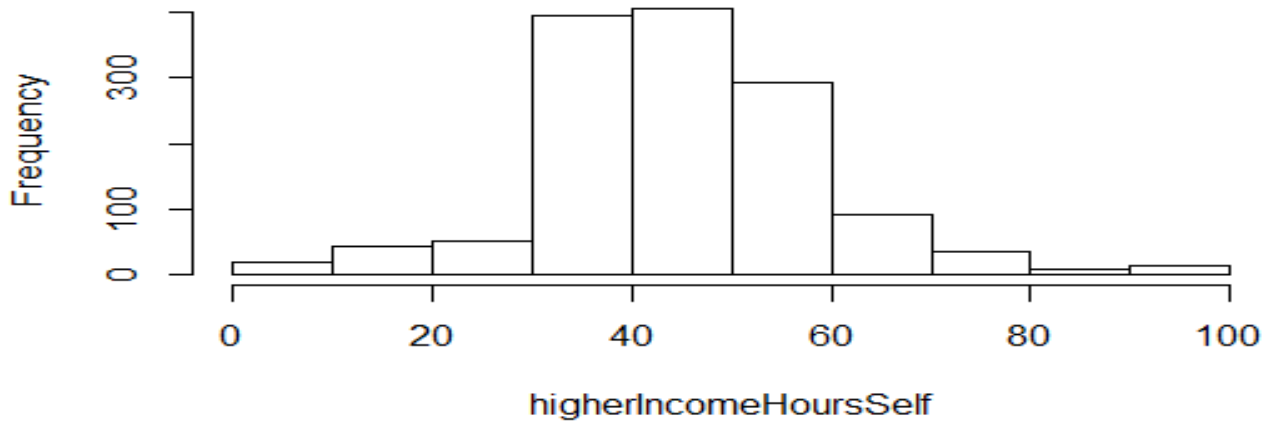
> # 2 points
> # Question 12: Your friend works for the government and claims that in order to make more money, you have to work longer hours. Use this data set to determine if your friend is right. State your conclusion in the comments.

higherIncomeHoursGov<-NULL
higherIncomeHoursSelf<-NULL
higherIncomeHoursPrivate<-NULL
higherIncomeHoursOther<-NULL
for (x in 1:length(datafram$IncomeLevel)){
  if (grepl(">50K", datafram$IncomeLevel[x]) & grepl("Government", datafram$worksector[x])){
    higherIncomeHoursGov<-append(higherIncomeHoursGov,datafram$HoursPerWeek[x])
  } else if (grepl(">50K", datafram$IncomeLevel[x]) & grepl("SelfEmployed", datafram$worksector[x])){
    higherIncomeHoursSelf<-append(higherIncomeHoursSelf,datafram$HoursPerWeek[x])
  } else if (grepl(">50K", datafram$IncomeLevel[x]) & grepl("Private", datafram$worksector[x])){
    higherIncomeHoursPrivate<-append(higherIncomeHoursPrivate,datafram$HoursPerWeek[x])
  } else if (grepl(">50K", datafram$IncomeLevel[x]) & grepl("Other", datafram$worksector[x])){
    higherIncomeHoursOther<-append(higherIncomeHoursOther,datafram$HoursPerWeek[x])
  }
}
```

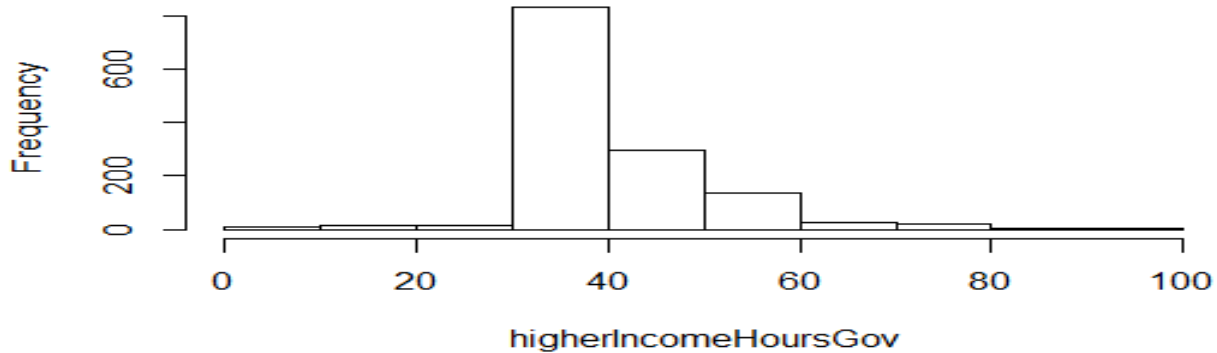
```
}  
}
```

```
hist(higherIncomeHoursGov)  
hist(higherIncomeHoursSelf)  
hist(higherIncomeHoursPrivate)  
hist(higherIncomeHoursOther)
```

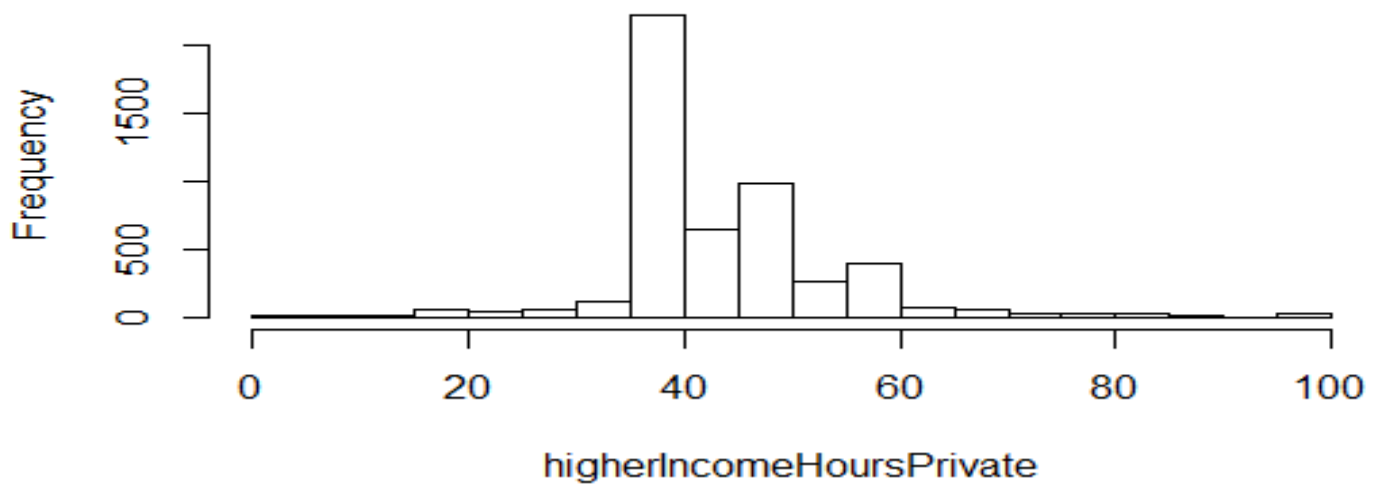
Histogram of higherIncomeHoursSelf



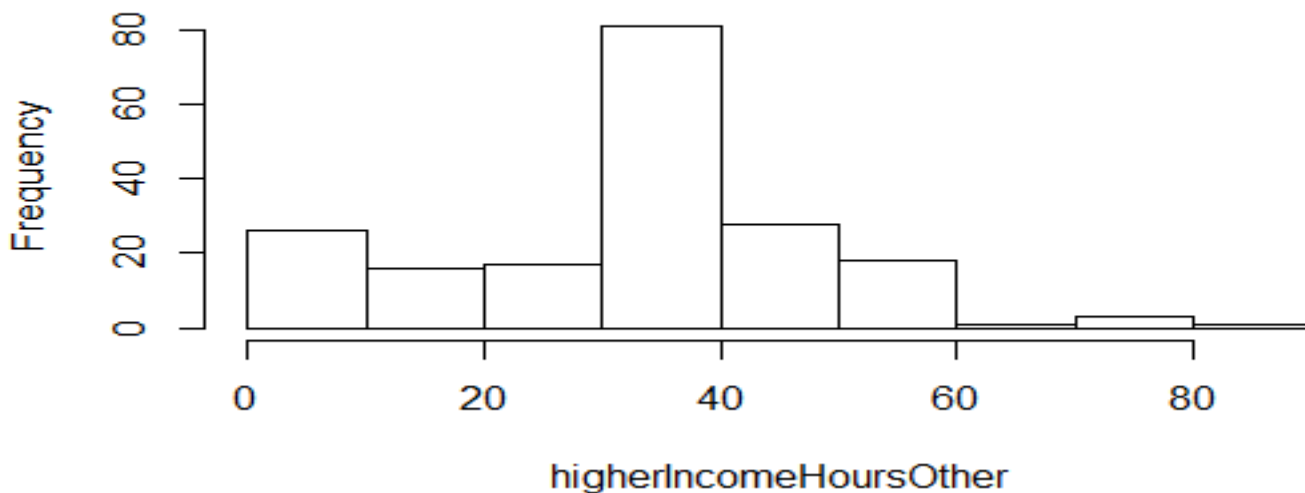
Histogram of higherIncomeHoursGov



Histogram of higherIncomeHoursPrivate



Histogram of higherIncomeHoursOther



```
> conclusion<-"According to data analysis, we can say that income level (>50K) is not varied in the same trend as the number of hours per week. Here we can see in the work Sector "Other" have higher income on low working hours. So this statement 'to make more money, you have to work longer hours' is not valid."  
> print(conclusion)
```

```
[1] "According to data analysis, we can say that income level (>50K) is not varied in the same trend as the number of hours per week. Here we can see in the work Sector "Other" have higher income on low working hours. So this statement 'to make more money, you have to work longer hours' is not valid."
```

```
# 3 points  
# Question 13: Implement a function call charCombos from scratch  
# (only using base R; no using 3rd party libraries for this question!)  
# that counts how many times each parameter z of  
# letters occur sequentially in a string.
```

```
# For example, charCombos('abcbcb', z=2)
# should return ab:1, bc:2, cb: 2
# charCombos('abcbcb', z=3) should return
# abc: 1, bcb: 2, cbc: 1
# Hint, use the substr function
```

```
> charCombos <- function(string,z){
+   combos<-NULL
+   for (i in 1:(nchar(string)-(z-1))) {
+     combos<-append(combos,substr(string,i,i+(z-1)))
+   }
+   return(table(combos))
+ }
```

```
> myTestString <- 'abcbcb'
> charCombos(string = myTestString, z= 2)
```

```
combos
ab bc cb
 1  2  2
> charCombos(string = myTestString, z= 3)
combos
abc bcb cbc
 1   2   1
```

```
# 3 points
```

```
# Question 14: In the traditional English language, students are taught
# "Always use a 'u' after a 'q'!". Using the function from
# question 13 (won't get full credit otherwise)
# and a link to a dictionary of english word provided below
# determine the percentage of times that 'q' is indeed
# immediately followed by a 'u'.
# Specifically, words containing q that are immediately followed
# by a u divided by total number of words containing q.
# Hint: Ensure you don't count q followed by whitespace or other
# non-alphanumeric characters. For example, don't count
# something like "Shaq upended the game."
# Note for words with multiple q's or multiple q-u's, count them once.
# This can be rather naively done and achieve short run times. If you are
# issues with the length of this run-time, check your code.
```

```
> bigListOfWords <- readLines('https://raw.githubusercontent.com/dwyl/english-words/master/words.txt')
> allQwords<-NULL
> allQuwords<-NULL
> for (i in 1:length(bigListOfWords)){
+   if (grepl("q",bigListOfWords[i]) || grepl("Q",bigListOfWords[i])){
+     if(!(grepl("q ",bigListOfWords[i]) || grepl("q[^[:alnum:]]",bigListOfWords[i]))){
+       allQwords<-append(allQwords,charCombos(bigListOfWords[i],nchar(bigListOfWords[i])))
+     }
+   }
+   if (grepl("qu",bigListOfWords[i]) || grepl("QU",bigListOfWords[i]) || grepl("Qu",bigListOfWords[i])){
+     allQuwords<-append(allQuwords,charCombos(bigListOfWords[i],nchar(bigListOfWords[i])))
+   }
+ }
> pctQU <-(length(allQuwords)/length(allQwords))*100
> print(pctQU)
[1] 96.9881
```

```
> # 3 points
```

```
# Question 15: Find the top 5
# most commonly used letters after q that are NOT equal to u sorted in descending
# order of frequency.
> allletters<-NULL
> for (i in 1:length(bigListOfWords)){
+   let<-unlist(strsplit(bigListOfWords[i], ""))
+   for (k in 1:length(let)){
+     if (grepl("q", let[k],ignore.case = TRUE)){
```

```
+       if(!(grep("u", let[k+1],ignore.case = TRUE)) && k != length(let) && !(grep("^[[:alpha:]]", let[k+1],ignore.case = TRUE)))
+         allletters<-append(allletters,let[k+1])
+     }
+ }
> lowercase<-unlist(lapply(allletters, tolower))
> top5 <-head(sort(table(lowercase),decreasing = TRUE),5)
> print(top5)
lowercase
 a  i  s  r  e
40 23 17 14 11
```