Module 3 In Class Activity

**Author:** Andres Felipe Alba Hernández **Department:** Electrical Engineering  
**Date:** September 16, 2018  
**Course:** ISYE670 Data Science for Engineers  
**Professor:** Dr. Christine Nguyen  
**Northern Illinois University**

# PART A

1. Load the islands dataset and obtain the total number of observations.

rm(list=ls())   
data("islands") #load the dataset islands  
#help("islands")  
summary(islands)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 12.0 20.5 41.0 1253.0 183.2 16990.0

**Description:** The areas in thousands of square miles of the landmasses which exceed 10,000 square miles.  
**Usage:** islands **Answer: The total number of observation is:**

1. Calculation of mean and media:

mean(islands)

## [1] 1252.729

median(islands)

## [1] 41

1. Using range calculate max and min value size of the islands: it will be display as min max in the output of the following command.

range(islands)

## [1] 12 16988

1. Standard desviation and Range.

sd(islands) #standard desviation

## [1] 3371.146

range(islands) #range output = (min,max)

## [1] 12 16988

1. Quantile Function:
2. Find the quantiles for: 0%, 25%, 50%, 75%, 100%

quantile(islands)

## 0% 25% 50% 75% 100%   
## 12.00 20.50 41.00 183.25 16988.00

1. Find the quantiles for: .05%, 95%

quantile(islands,probs = c(0.005,0.95))

## 0.5% 95%   
## 12.235 8481.750

1. What does the parameter na.rm do?

data\_test <- c(0.5,10,NaN)  
quantile(data\_test,na.rm = TRUE)

## 0% 25% 50% 75% 100%   
## 0.500 2.875 5.250 7.625 10.000

try(quantile(data\_test)) #How can  
try(quantile(data\_test,na.rm = FALSE))

If the na.rm flag is set as FALSE the NaN values are not removed before the computation, therefore they are not allow to be in the data set. As may be observed above.

1. Interqueartile range: As can be observed in the calculation below it correspond to the different between the 75% quartile and the 25% queartile.

quantile(islands)

## 0% 25% 50% 75% 100%   
## 12.00 20.50 41.00 183.25 16988.00

calculate\_IQR <- (183.25-20.5)  
print("Calculate IQR:")

## [1] "Calculate IQR:"

print(calculate\_IQR)

## [1] 162.75

print("IQR using the command")

## [1] "IQR using the command"

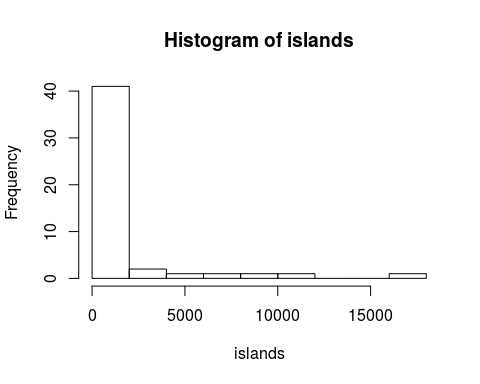
IQR(islands)

## [1] 162.75

#IQR(data\_test,na.rm = TRUE)

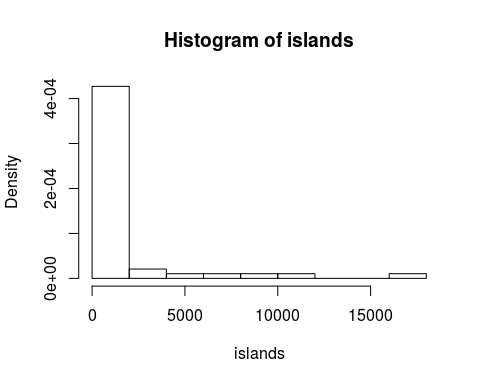
1. Create a histogram of the data, where x = islands. The histogram will help you understand how the data is distributed: From the histogram and the probability plot it can be infered that the islans of less than 2500 square miles are more frequent while values between 2.5k and 11k are almost equially distributed.
2. Using the frequency of each bin:

hist(islands)



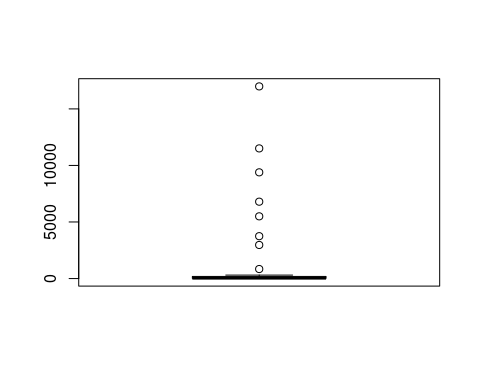
1. Using the proportion of each bin:

hist(islands,prob=TRUE)



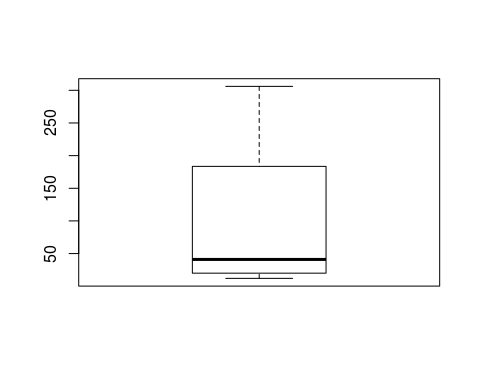
1. Create box-plots. Use boxplot(x). This is another way for you to visualize how the data is distributed, and whether there are any outliers.
2. Create a boxplot using all the data in islands Definition of boxplot (extracted from wikipedia): In descriptive statistics, a box plot or boxplot is a method for graphically depicting groups of numerical data through their quartiles.

boxplot(islands)



1. What does the parameter outline mean? In this case the values out of the usual distribution are not drawn. As may be observe in the boxplot below this allow to appreciate the distribution of the quartiles having the average in a value close to 50 (widther line). The area inside the box correspond to the IQR while the whiskers describe the other quartiles.
2. Create a boxplot without outliers

boxplot(islands, outline = FALSE)



1. Using the function boxplot, find the outliers of islands:

boxplot(islands, plot=FALSE)$out

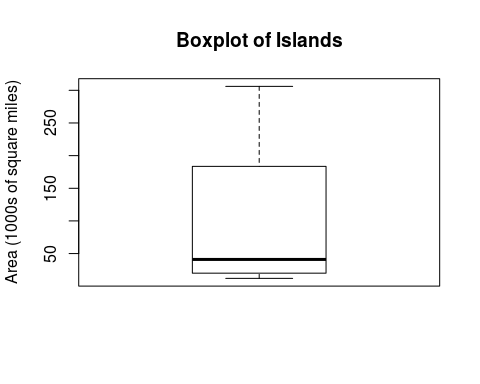
## Africa Antarctica Asia Australia Europe   
## 11506 5500 16988 2968 3745   
## Greenland North America South America   
## 840 9390 6795

1. What are the outliers? Outliers are points of data, observations, that are distant from the others. They may indicade variability in the mesuarments or also they may indicate errors. R can calculate the value with the largest difference between it and sample mean with the command below.

#outlier(islands) #it works when I installed but not when I try to produce the doc

1. Give the plot you just created a title and y-axis label

boxplot(islands, outline = FALSE)  
title(main="Boxplot of Islands",ylab = "Area (1000s of square miles)")



1. Create a stem and leaf plot of islands: Use stem(x) function.
2. Where is the output? The steam plot split the data using the fist number and how may time appear with other numbers.

stem(islands)

##   
## The decimal point is 3 digit(s) to the right of the |  
##   
## 0 | 00000000000000000000000000000111111222338  
## 2 | 07  
## 4 | 5  
## 6 | 8  
## 8 | 4  
## 10 | 5  
## 12 |   
## 14 |   
## 16 | 0

1. How would you interpret the output? Hint: The histogram you made could be helpful From this we can observed that the mayority of the data start with 0000.XX which means low values. Comparing this with the boxplot it can be observed that the majority of the population is in the bottom half.

**PART B**

1. The data is imported using the read.table() command, it is necessary to use tab as the separator. The first row correspond to the header: “Pre-test Quiz 1 Quiz 2 Midterm Quiz 3 Quiz 4 Final”

dt2 <- read.table("data\_ICA3.txt", sep="\t") #The variable dt2 represent the dataset(table)of the scores.

1. The data imported trough the read.table() command will provide a list with the data where each element in the list correspond to one column in the table:

mode(dt2)

## [1] "list"

1. How many observations are there? The number of observations correspont to the number of rows in the table for this case is 50 observations.

dt2

## V1 V2 V3 V4 V5 V6 V7  
## 1 Pre-test Quiz 1 Quiz 2 Midterm Quiz 3 Quiz 4 Final  
## 2 56 49 95 64 78 72 90  
## 3 58 66 82 100 79 91  
## 4 63 45 61 100 83 48 87  
## 5 50 44 100 92 45 69 78  
## 6 58 64 81 59 64 97  
## 7 61 56 57 100 75 44 87  
## 8 63 58 56 80 40 56 85  
## 9 67 64 79 72 76 75 90  
## 10 72 62 100 80 89 89  
## 11 56 50 68 68 45 62 91  
## 12 64 52 78 75 100  
## 13 60 50 78 74 56 70 87  
## 14 64 93 70 100 81 98  
## 15 55 38 34 76 100 100 100  
## 16 57 54 53 100 53 83  
## 17 60 82 71 74 35 94  
## 18 71 47 69 68 88  
## 19 56 53 67 99 100 57 93  
## 20 61 63 80 79 49 90  
## 21 54 57 95 39 78 84 86  
## 22 50 59 87 78 100 82 93  
## 23 83 64 100 52 86  
## 24 60 60 51 91 87 71 76  
## 25 61 60 76 99 79 2 95  
## 26 57 55 61 84 80 18 92  
## 27 60 85 43 48 31 85  
## 28 59 59 80 68 93 94  
## 29 69 77 65 82 47 84  
## 30 63 56 64 69 64 82 90  
## 31 57 66 81 98 88 77 91  
## 32 63 79 66 43 76 68 93  
## 33 68 75 95 100 30 92  
## 34 63 51 50 86 80 100 98  
## 35 60 70 92 94 97 24 90  
## 36 59 63 84 92 56 63 89  
## 37 83 89 90 81 91  
## 38 81 41 57 47 100  
## 39 55 41 66 89 70 68 80  
## 40 54 49 55 65 47 29 88  
## 41 63 47 100 100 100 93  
## 42 56 58 71 70 100 88 98  
## 43 55 61 69 66 63 92  
## 44 51 81 63 92 89  
## 45 68 77 65 100 99 56 81  
## 46 63 61 89 94 94 35 92  
## 47 61 71 58 97 100 69 100  
## 48 54 54 93 100 100 63 92  
## 49 61 41 81 95 37 52 88  
## 50 89 80 82 87 95  
## 51 58 78 98 30 93

#mode(dt2)  
#summary(dt2)  
#length(dt2[1])

1. How many variables are there? The number of variables correspond to the number of columns in the data. For this case it is 7 columns.
2. names(x): This command names show you the labes of each column (variable) into the table.

names(dt2)

## [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7"

1. Changing the names example:

print("Scores with original name")

## [1] "Scores with original name"

names(dt2)

## [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7"

names(dt2)[1] <- "Pre-test"  
names(dt2)

## [1] "Pre-test" "V2" "V3" "V4" "V5" "V6"   
## [7] "V7"

dt2

## Pre-test V2 V3 V4 V5 V6 V7  
## 1 Pre-test Quiz 1 Quiz 2 Midterm Quiz 3 Quiz 4 Final  
## 2 56 49 95 64 78 72 90  
## 3 58 66 82 100 79 91  
## 4 63 45 61 100 83 48 87  
## 5 50 44 100 92 45 69 78  
## 6 58 64 81 59 64 97  
## 7 61 56 57 100 75 44 87  
## 8 63 58 56 80 40 56 85  
## 9 67 64 79 72 76 75 90  
## 10 72 62 100 80 89 89  
## 11 56 50 68 68 45 62 91  
## 12 64 52 78 75 100  
## 13 60 50 78 74 56 70 87  
## 14 64 93 70 100 81 98  
## 15 55 38 34 76 100 100 100  
## 16 57 54 53 100 53 83  
## 17 60 82 71 74 35 94  
## 18 71 47 69 68 88  
## 19 56 53 67 99 100 57 93  
## 20 61 63 80 79 49 90  
## 21 54 57 95 39 78 84 86  
## 22 50 59 87 78 100 82 93  
## 23 83 64 100 52 86  
## 24 60 60 51 91 87 71 76  
## 25 61 60 76 99 79 2 95  
## 26 57 55 61 84 80 18 92  
## 27 60 85 43 48 31 85  
## 28 59 59 80 68 93 94  
## 29 69 77 65 82 47 84  
## 30 63 56 64 69 64 82 90  
## 31 57 66 81 98 88 77 91  
## 32 63 79 66 43 76 68 93  
## 33 68 75 95 100 30 92  
## 34 63 51 50 86 80 100 98  
## 35 60 70 92 94 97 24 90  
## 36 59 63 84 92 56 63 89  
## 37 83 89 90 81 91  
## 38 81 41 57 47 100  
## 39 55 41 66 89 70 68 80  
## 40 54 49 55 65 47 29 88  
## 41 63 47 100 100 100 93  
## 42 56 58 71 70 100 88 98  
## 43 55 61 69 66 63 92  
## 44 51 81 63 92 89  
## 45 68 77 65 100 99 56 81  
## 46 63 61 89 94 94 35 92  
## 47 61 71 58 97 100 69 100  
## 48 54 54 93 100 100 63 92  
## 49 61 41 81 95 37 52 88  
## 50 89 80 82 87 95  
## 51 58 78 98 30 93

1. Finding Missing values in my dataset:
2. Try finding NA values for the Pre-test using == “NA” or == NA or == “ “ . What is the output? \*\*The only one that works properly is ==“”

rm(dt2)  
dt2 <- read.table("data\_ICA3.txt", sep="\t") #The variable dt2 represent the dataset(table)of the scores.  
names(dt2)[1] <- "Pre-test"  
#dt2$`Pre-test`==NA  
#dt2$`Pre-test`=="NA"  
dt2$`Pre-test`=="" #This is the only one that give me the right result

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
## [12] FALSE FALSE TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE  
## [23] TRUE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE TRUE  
## [34] FALSE FALSE FALSE TRUE TRUE FALSE FALSE TRUE FALSE FALSE TRUE  
## [45] FALSE FALSE FALSE FALSE FALSE TRUE TRUE

b)Try using the is.na() function. Use is.na(scores$’Pre-test’) on your data. What is the output? **The output should give TRUE at the positions where the data is missing.**

#Other things I tested:  
 #v\_test <- dt2$`Pre-test` #this give me a type integer. Factor with levels  
 #v\_test2 <- dt2[1] #This give me a list  
 #typeof(v\_test)  
 #typeof(v\_test2)  
#Now testing the suggested command.   
#is.na(dt2$`Pre-test`) #this one does not work proper, I do not know why.  
#is.na(dt2[1]) #this one also does not work.  
dt2$`Pre-test`=="" #this one does not work

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
## [12] FALSE FALSE TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE  
## [23] TRUE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE TRUE  
## [34] FALSE FALSE FALSE TRUE TRUE FALSE FALSE TRUE FALSE FALSE TRUE  
## [45] FALSE FALSE FALSE FALSE FALSE TRUE TRUE

c)Use is.na(x) on the entire data set. What is the output? **The output should give TRUE at the positions where the data is missing.**

#dt2  
#is.na(dt2) #THIS IS NOT WORKING, whyy? #commenting out this line for now  
dt2=="" #This one work properly

## Pre-test V2 V3 V4 V5 V6 V7  
## [1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [2,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [3,] FALSE FALSE FALSE FALSE TRUE FALSE FALSE  
## [4,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [5,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [6,] FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
## [7,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [8,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [9,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [10,] TRUE FALSE FALSE FALSE FALSE FALSE FALSE  
## [11,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [12,] FALSE FALSE FALSE FALSE TRUE TRUE FALSE  
## [13,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [14,] TRUE FALSE FALSE FALSE FALSE FALSE FALSE  
## [15,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [16,] FALSE FALSE FALSE TRUE FALSE FALSE FALSE  
## [17,] TRUE FALSE FALSE FALSE FALSE FALSE FALSE  
## [18,] TRUE TRUE FALSE FALSE FALSE FALSE FALSE  
## [19,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [20,] FALSE FALSE FALSE TRUE FALSE FALSE FALSE  
## [21,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [22,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [23,] TRUE FALSE FALSE FALSE FALSE TRUE FALSE  
## [24,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [25,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [26,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [27,] TRUE FALSE FALSE FALSE FALSE FALSE FALSE  
## [28,] FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
## [29,] TRUE FALSE FALSE FALSE FALSE FALSE FALSE  
## [30,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [31,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [32,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [33,] TRUE FALSE FALSE FALSE FALSE FALSE FALSE  
## [34,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [35,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [36,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [37,] TRUE TRUE FALSE FALSE FALSE FALSE FALSE  
## [38,] TRUE TRUE FALSE FALSE FALSE FALSE FALSE  
## [39,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [40,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [41,] TRUE FALSE FALSE FALSE FALSE FALSE FALSE  
## [42,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [43,] FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
## [44,] TRUE FALSE FALSE FALSE FALSE TRUE FALSE  
## [45,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [46,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [47,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [48,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [49,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [50,] TRUE TRUE FALSE FALSE FALSE FALSE FALSE  
## [51,] TRUE TRUE FALSE FALSE FALSE FALSE FALSE

1. How many total missing values or “NA” are in the data? **The total of missing data is 29**

total\_of\_missing <- sum(dt2=="")   
print(total\_of\_missing)

## [1] 29

anyNA(dt2,recursive = TRUE) #For some reason the is.na is not working well for me

## [1] FALSE

e)What happens when you try to use is.nan( ) or is.infinite( ) on the entire data set? To understand, look up these function in help Inf = negative or positive number that are infinity (too big for example number/0) NaN = Non a number (I number like 0/0 will be indefined or non a number)

is.infinite(dt2$`Pre-test`)

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

is.nan(dt2$`Pre-test`)

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

1. To find the number of missing values per column: apply(is.na(scores),2,sum)

apply(dt2=="",2,sum)

## Pre-test V2 V3 V4 V5 V6 V7   
## 14 5 0 2 2 6 0

#other way  
sum(dt2[2]=="")

## [1] 5

1. Repeat the above to find the number of missing values per row

apply(dt2=="",1,sum)

## [1] 0 0 1 0 0 1 0 0 0 1 0 2 0 1 0 1 1 2 0 1 0 0 2 0 0 0 1 1 1 0 0 0 1 0 0  
## [36] 0 2 2 0 0 1 0 1 2 0 0 0 0 0 2 2

#sum(dt2[3,]=="") #This is for the third row for example

1. Using the apply function, find the average of the scores per row: apply(scores,1, mean). What is the output? Does it make sense?

#apply(na.omit(dt2),1,mean)  
#The one in the second argument means row, this data structure have two dimensions  
apply((dt2!=""),1,mean) #Average per row omiting empty positions.

## [1] 1.0000000 1.0000000 0.8571429 1.0000000 1.0000000 0.8571429 1.0000000  
## [8] 1.0000000 1.0000000 0.8571429 1.0000000 0.7142857 1.0000000 0.8571429  
## [15] 1.0000000 0.8571429 0.8571429 0.7142857 1.0000000 0.8571429 1.0000000  
## [22] 1.0000000 0.7142857 1.0000000 1.0000000 1.0000000 0.8571429 0.8571429  
## [29] 0.8571429 1.0000000 1.0000000 1.0000000 0.8571429 1.0000000 1.0000000  
## [36] 1.0000000 0.7142857 0.7142857 1.0000000 1.0000000 0.8571429 1.0000000  
## [43] 0.8571429 0.7142857 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000  
## [50] 0.7142857 0.7142857

1. As can be observed below 328 elements are remining after omiting the empty ones.

sum(dt2!="")

## [1] 328

1. The average per column is found in in a similar way as in h but using the 2 in the second argument of the apply comand.

apply((dt2!=""),2,mean)

## Pre-test V2 V3 V4 V5 V6 V7   
## 0.7254902 0.9019608 1.0000000 0.9607843 0.9607843 0.8823529 1.0000000

1. Then, use this clean set of data to randomly sample. Generate 5 new samples from the existing clean set and save it to a new object called newSample:

scoresOmit <- dt2!=""  
scoresOmit[sample(1:nrow(scoresOmit),5,replace = TRUE)]

## [1] FALSE FALSE TRUE TRUE TRUE

1. To combine this set with the existing set, you can use rbind(x,y)

r\_scores <- rbind(scoresOmit,dt2)  
r\_scores

## Pre-test V2 V3 V4 V5 V6 V7  
## 1 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 2 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 3 TRUE TRUE TRUE TRUE FALSE TRUE TRUE  
## 4 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 5 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 6 TRUE TRUE TRUE TRUE TRUE FALSE TRUE  
## 7 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 8 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 9 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 10 FALSE TRUE TRUE TRUE TRUE TRUE TRUE  
## 11 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 12 TRUE TRUE TRUE TRUE FALSE FALSE TRUE  
## 13 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 14 FALSE TRUE TRUE TRUE TRUE TRUE TRUE  
## 15 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 16 TRUE TRUE TRUE FALSE TRUE TRUE TRUE  
## 17 FALSE TRUE TRUE TRUE TRUE TRUE TRUE  
## 18 FALSE FALSE TRUE TRUE TRUE TRUE TRUE  
## 19 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 20 TRUE TRUE TRUE FALSE TRUE TRUE TRUE  
## 21 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 22 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 23 FALSE TRUE TRUE TRUE TRUE FALSE TRUE  
## 24 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 25 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 26 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 27 FALSE TRUE TRUE TRUE TRUE TRUE TRUE  
## 28 TRUE TRUE TRUE TRUE TRUE FALSE TRUE  
## 29 FALSE TRUE TRUE TRUE TRUE TRUE TRUE  
## 30 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 31 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 32 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 33 FALSE TRUE TRUE TRUE TRUE TRUE TRUE  
## 34 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 35 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 36 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 37 FALSE FALSE TRUE TRUE TRUE TRUE TRUE  
## 38 FALSE FALSE TRUE TRUE TRUE TRUE TRUE  
## 39 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 40 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 41 FALSE TRUE TRUE TRUE TRUE TRUE TRUE  
## 42 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 43 TRUE TRUE TRUE TRUE TRUE FALSE TRUE  
## 44 FALSE TRUE TRUE TRUE TRUE FALSE TRUE  
## 45 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 46 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 47 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 48 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 49 TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## 50 FALSE FALSE TRUE TRUE TRUE TRUE TRUE  
## 51 FALSE FALSE TRUE TRUE TRUE TRUE TRUE  
## 52 Pre-test Quiz 1 Quiz 2 Midterm Quiz 3 Quiz 4 Final  
## 53 56 49 95 64 78 72 90  
## 54 58 66 82 100 79 91  
## 55 63 45 61 100 83 48 87  
## 56 50 44 100 92 45 69 78  
## 57 58 64 81 59 64 97  
## 58 61 56 57 100 75 44 87  
## 59 63 58 56 80 40 56 85  
## 60 67 64 79 72 76 75 90  
## 61 72 62 100 80 89 89  
## 62 56 50 68 68 45 62 91  
## 63 64 52 78 75 100  
## 64 60 50 78 74 56 70 87  
## 65 64 93 70 100 81 98  
## 66 55 38 34 76 100 100 100  
## 67 57 54 53 100 53 83  
## 68 60 82 71 74 35 94  
## 69 71 47 69 68 88  
## 70 56 53 67 99 100 57 93  
## 71 61 63 80 79 49 90  
## 72 54 57 95 39 78 84 86  
## 73 50 59 87 78 100 82 93  
## 74 83 64 100 52 86  
## 75 60 60 51 91 87 71 76  
## 76 61 60 76 99 79 2 95  
## 77 57 55 61 84 80 18 92  
## 78 60 85 43 48 31 85  
## 79 59 59 80 68 93 94  
## 80 69 77 65 82 47 84  
## 81 63 56 64 69 64 82 90  
## 82 57 66 81 98 88 77 91  
## 83 63 79 66 43 76 68 93  
## 84 68 75 95 100 30 92  
## 85 63 51 50 86 80 100 98  
## 86 60 70 92 94 97 24 90  
## 87 59 63 84 92 56 63 89  
## 88 83 89 90 81 91  
## 89 81 41 57 47 100  
## 90 55 41 66 89 70 68 80  
## 91 54 49 55 65 47 29 88  
## 92 63 47 100 100 100 93  
## 93 56 58 71 70 100 88 98  
## 94 55 61 69 66 63 92  
## 95 51 81 63 92 89  
## 96 68 77 65 100 99 56 81  
## 97 63 61 89 94 94 35 92  
## 98 61 71 58 97 100 69 100  
## 99 54 54 93 100 100 63 92  
## 100 61 41 81 95 37 52 88  
## 101 89 80 82 87 95  
## 102 58 78 98 30 93

1. To reassign “NA” values to 0

scores <- dt2  
#levels(scores)  
#scores[scores==""] <- 0  
 #scores[scores==""] <- NA  
scores[is.na(scores)] <- 0

1. The mean will change because the NA values are reemplace by zeros increasing the number of samples and the mean will be the
2. In this case n will be bigger.