Collecting Data

Blake Pappas

8/24/2021

Getting started with R Markdown

R documentation

One of the reasons R is so popular among statisticians is that its libraries contain functions for so many statistical procedures. Every R function (if it is from a library hosted on CRAN) has documentation that tells you how to use it.

The **Useage** section of R documentation shows the syntax to use when calling the function. The **Arguments** section tells you what items should be put inside the parenthesis, separated by commas. **Value** describes the output of the function.

To view the documentation for a function, type "?" followed by the name of the function into the console. Try this out for the "quantile" function by typing ?quantile into the console now.

quantile?

Logical Operators and Subsetting

First, create some objects in the workspace.

```
# Create Some Data Vectors
x <- c(12.3, 32.4, 8.6, 9, 11.5)
y <- c("red", "red", "green", "red", "green")
car_no <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 17, 18, 19, 20, 21, 22, 23, 24, 34, 41, 42, 48)
ACC_Schools <- c('Clemson', 'Wake Forest', 'Duke', 'North Carolina', 'NC State', 'Syracuse')

# View the 4th Element of x
x[4]
```

```
## [1] 9
```

```
# View the 5th Element of ACC_Schools
ACC_Schools[5]
```

```
## [1] "NC State"
```

```
# View the 19th Element of car_no
car_no[19]
```

```
## [1] 22
```

```
# View the 2nd and 4th Elements of y
y[c(2, 4)]
```

```
## [1] "red" "red"
```

```
# View Elements 1, 2, 3, 4, 5, 10, 11, 12, 13, 14, 15 of car_no
car_no[c(1:5, 10:15)]
```

```
##  [1]  1  2  3  4  5 10 11 12 14 17 18
```

```
# View the 1st, 3rd, and 5th Elements of ACC_Schools
ACC_Schools[c(1, 3, 5)]
```

```
## [1] "Clemson"  "Duke"     "NC State"
```

Logical operators check some condition on an object (equality, inequality, etc.) and return TRUE or FALSE. Here are a few logical operators in R.

```
x[1] > 10  # Checks whether x[1] is >10
```

```
## [1] TRUE
```

```
x > 10 # Returns a logical vector, returning T/F for each element of x
```

```
## [1]  TRUE  TRUE FALSE FALSE  TRUE
```

```
which( x > 10 )   # Returns the indices of elements of x for which the condition is true
```

```
## [1] 1 2 5
```

```
x[which(x > 10)]   # Returns the subset of x for which the condition is true
```

```
## [1] 12.3 32.4 11.5
```

Other logical operators include <, <=, >=, ==, and !=. Try these out to see what they do.

```
y == 'red'
```

```
## [1]  TRUE  TRUE FALSE  TRUE FALSE
```

```
# Add Your Own Lines of Code
y != 'green'
```

```
## [1]  TRUE  TRUE FALSE  TRUE FALSE
```

```
car_no < 11
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE
```

```
x > 12
```

```
## [1]  TRUE  TRUE FALSE FALSE FALSE
```

```
car_no >= 13
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [25]  TRUE
```

```
x <= 9
```

```
## [1] FALSE FALSE  TRUE  TRUE FALSE
```

```
ACC_Schools == 'Clemson'
```

```
## [1]  TRUE FALSE FALSE FALSE FALSE FALSE
```

Data Frames and (More) Subsetting

A data frame is the object used to store rectangular data sets in R. It can hold a mix of numeric and categorical variables. When you read .csv files, they are automatically stored as data frames in your R workspace. You can also construct one out of vectors of the same length using the function "data.frame".

```
results <- data.frame(size = x, color = y)
results
```

```
##   size color
## 1 12.3   red
## 2 32.4   red
## 3  8.6 green
## 4  9.0   red
## 5 11.5 green
```

```
results <- data.frame(color = y, size = x)
results
```

```
##   color size
## 1   red 12.3
## 2   red 32.4
## 3 green  8.6
## 4   red  9.0
## 5 green 11.5
```

Use the $ to work with a single variable in the data frame.

```
results$size
```

```
## [1] 12.3 32.4  8.6  9.0 11.5
```

```
results$color
```

```
## [1] "red"   "red"   "green" "red"   "green"
```

```
# Add a New Variable Called "diameter":
results$diameter <- c(4.5, 6.2, 8.8, 9.0, 1.3)
head(results)
```

```
##   color size diameter
## 1   red 12.3      4.5
## 2   red 32.4      6.2
## 3 green  8.6      8.8
## 4   red  9.0      9.0
## 5 green 11.5      1.3
```

```
# Pull First 2 Results
head(results, 2)
```

```
##   color size diameter
## 1   red 12.3      4.5
## 2   red 32.4      6.2
```

The subset function is one way to identify a subset of rows of the data frame that satisfy a logical condition. The first argument is the name of your data frame. The 2nd argument is a logical condition.

```
small.results <- subset(results, size < 20)
small.results
```

```
##   color size diameter
## 1   red 12.3      4.5
## 3 green  8.6      8.8
## 4   red  9.0      9.0
## 5 green 11.5      1.3
```

```
large.results <- subset(results, size > 20)
large.results
```

```
##   color size diameter
## 2   red 32.4      6.2
```

Here is a second example of the subset function:

```
green.results <- subset(results, color == 'green')
green.results
```

```
##   color size diameter
## 3 green  8.6      8.8
## 5 green 11.5      1.3
```

```
notgreen.results <- subset(results, color != 'green')
notgreen.results
```

```
##   color size diameter
## 1   red 12.3      4.5
## 2   red 32.4      6.2
## 4   red  9.0      9.0
```

An equivalent way of subsetting using a logical condition:

```
which(results$color == 'green')  # This returns the rows for which the condition is true
```

```
## [1] 3 5
```

```
green.results2 <- results[which(results$color == 'green'), ]
green.results2
```

```
##   color size diameter
## 3 green  8.6      8.8
## 5 green 11.5      1.3
```

Exercises

Answer the following questions in the space below.

Exercise 1

View the documentation for the function "t.test". The "Value" section describes all of the pieces of information that are stored in the output of the function. How many components are in the output?

```
?t.test
```

```
## starting httpd help server ... done
```

Answer: 10

Note: Data in R will typically have a class such as integer, numeric, factor, and so forth.

Statistical programmers can also define their own classes (called S3 classes) for the output of their functions. "t.test" is one function where this is the case: its output has a class called "htest". These "htest" objects will always have the components you counted up for this exercise.

Exercise 2

1. Use the "read.csv" function to load the "insurance.csv" data set into a data frame called "insurance". Write your code in the chunk below. Also include code to set your working directory.

Code:

```
insurance <- read.csv("insurance.csv")
```

2. Print the first five rows of the insurance data using the "head" function.

Code:

```
head(insurance, 5)
```

```
##   age    sex  bmi children smoker    region expenses smoker_bin
## 1  30   male 35.3        0    yes southwest 36837.47          1
## 2  54 female 31.9        3     no southeast 27322.73          0
## 3  51 female 18.1        0     no northwest  9644.25          0
## 4  48 female 32.2        1     no southeast  8871.15          0
## 5  41 female 31.6        1     no northeast  7358.18          0
```

Exercise 3

Answer the following questions about the insurance data set.

1. How many rows does it have? How many columns? The functions "dim," "nrow," and "ncol" might help. Create a new code chunk that returns the answer and type your answer below.

```
dim(insurance)
```

```
## [1] 47  8
```

```
nrow(insurance)
```

```
## [1] 47
```

```
ncol(insurance)
```

```
## [1] 8
```

Answer: The insurance dataset has 47 rows and 7 columns.

2. What type of variable is "region"? What about "expenses"? Use the "class" function.

```
class(insurance$region)
```

```
## [1] "character"
```

```r
class(insurance$expenses)
```

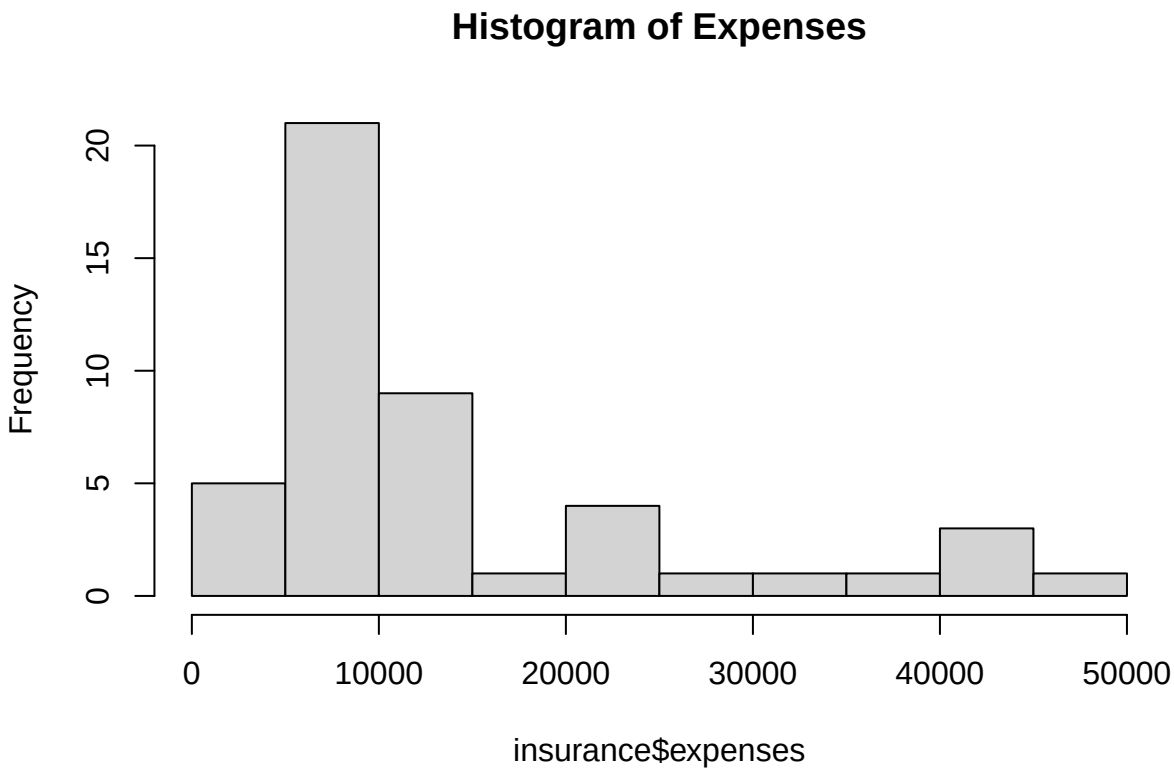
```
## [1] "numeric"
```

Answer: The variable "Region" is a character. The variable "expenses" is numeric.

3. Use the "hist" function to make a histogram of the variable "expenses". Customize the title using the argument "main".

Answer:

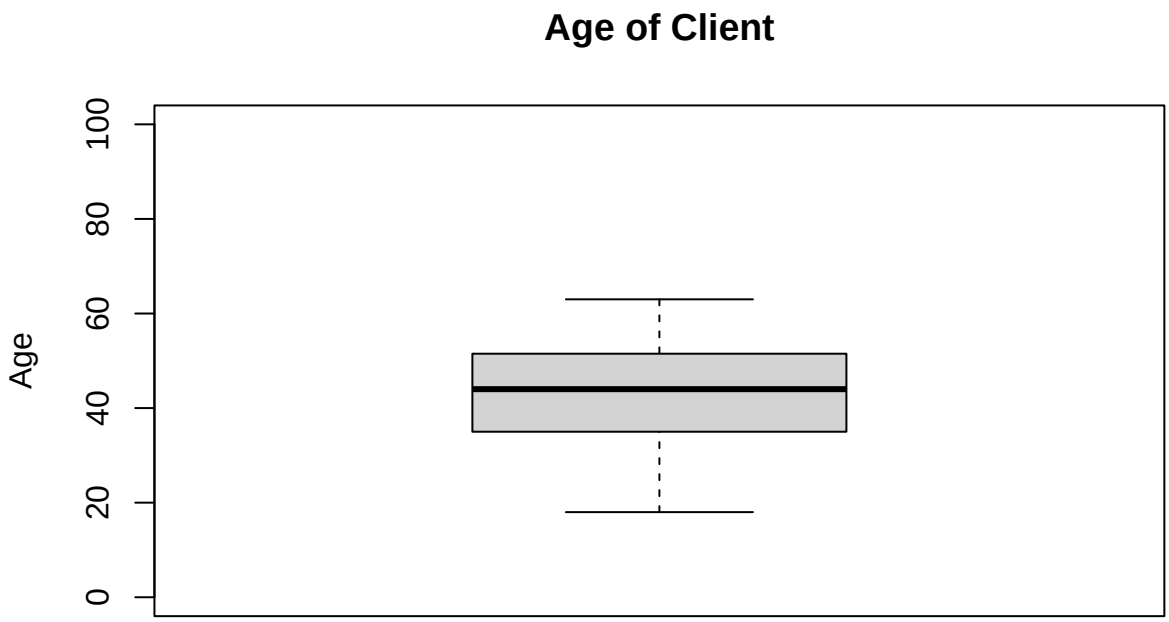
```r
hist(insurance$expenses, main = "Histogram of Expenses")
```



4. Use "boxplot" function to make a boxplot of the variable "age". Customize the scale of the y axis using the "ylim" argument.

Answer:

```r
boxplot(insurance$age,main = 'Age of Client', ylab = 'Age', ylim = c(0, 100))
```



Exercise 4

Suppose the data were collected in the following manner. An analyst at Statewide Insurance took a random sample from a database of all health insurance claims paid by the company in the year 2019. Each row of the data set represents one claim.

Do you think these data can be used to find an unbiased estimate of the following quantities? If you think there is potential for bias, give a brief explanation and state whether you think the quantity might be underestimated or overestimated.

1. The average age of customers insured by Statewide Insurance.

Answer: I think there is potential for bias in this data set when it comes to age. My reason for this is that older people are more likely to file insurance claims than younger people. When more claims are filed by older people, this drives up the average age of those customers of Statewide Insurance, thus overstating the average customer age.

2. The average dollar amount of a claim.

Answer: This data cannot be used to find an unbiased estimate. Insurance claims tend to be positively skewed due to unusually large expense amounts in a select few individuals. Therefore, it is likely that such a quantity is overestimated.*

3. The proportion of smokers in the country where Statewide Insurance is located.

Answer: I think these data can be used to find an unbiased estimate of the proportion of smokers in the country where Statewide Insurance is located.