Data Analytics II

QBS 103: Foundations of Data Science

August 1, 2024

Lesson Objectives

At the end of this lecture you should be able to:

1. Calculate appropriate summary statistics for a data frame
2. Build a function to produce a "Table One"
3. Implement R script dependencies

Additional Resources

Creating formatted tables in markdown: https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html

Identifying Appropriate Summary Statistics

First, lets generate some random data.

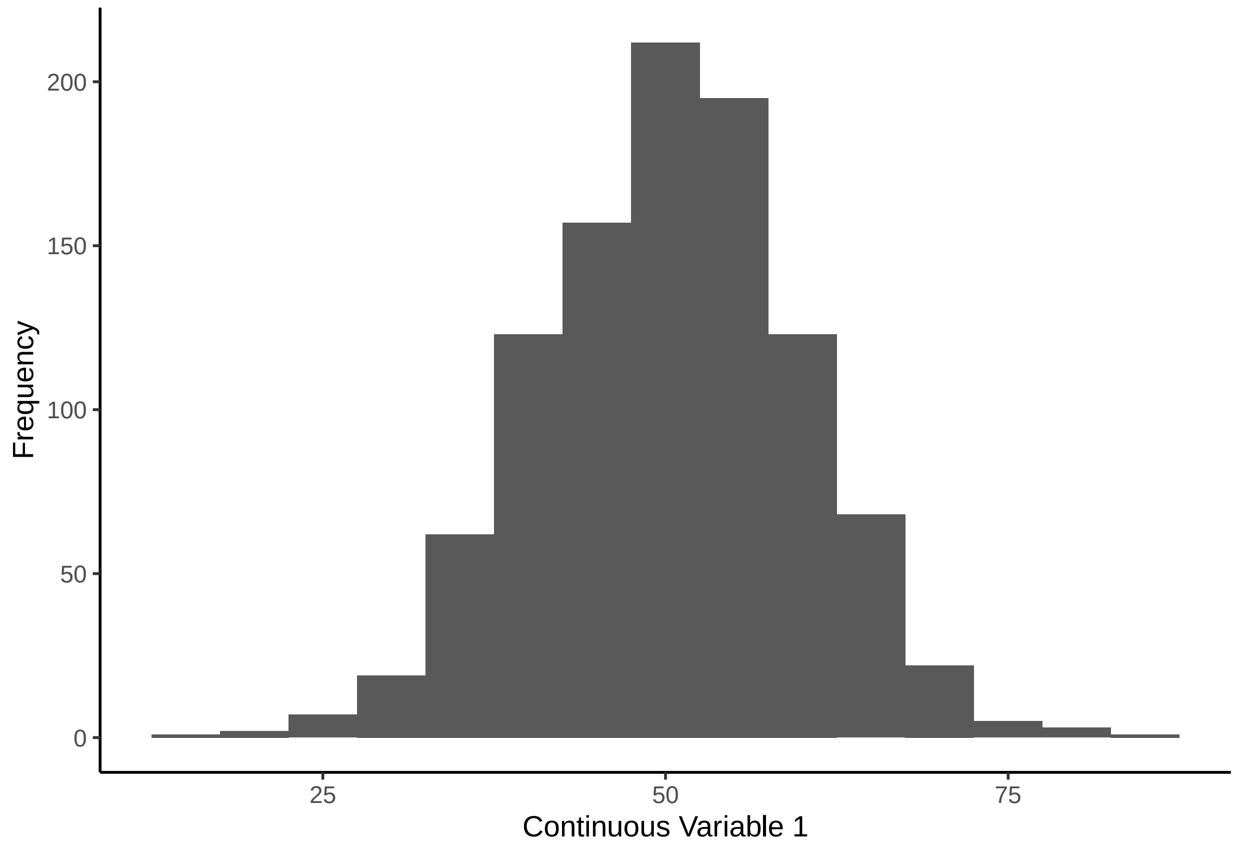
```r
library(tidyverse)
set.seed(3927)

randomData <- data.frame(Cont1 = rnorm(n = 1000,mean = 50,sd = 10),
                         Cont2 = rnorm(n = 1000, mean = 23,sd = 2),
                         Cont3 = rexp(n = 1000,rate = 0.25),
                         Cat1 = factor(rbinom(n = 1000,size = 1,prob = 0.50),
                                       labels = c(F,T)),
                         Cat2 = factor(rbinom(n = 1000,size = 1,prob = 0.25),
                                       labels = c(F,T)),
                         Cat3 = factor(rbinom(n = 1000,size = 2,prob = 0.40),
                                       labels = c('A','B','C')))
```

Continuous Variables

For normally distributed continuous variables, we report mean and standard deviation.

```r
ggplot(data = randomData,aes(x = Cont1)) +
  geom_histogram(binwidth = 5) +
  labs(x = 'Continuous Variable 1',y = 'Frequency') +
  theme_classic()
```



We can extract all the individual components of mean and standard deviation using the following functions.

```r
# Mean
mean(randomData$Cont1)
```

```
## [1] 50.34092
```

```r
# Standard deviation
sd(randomData$Cont3)
```

```
## [1] 3.76435
```

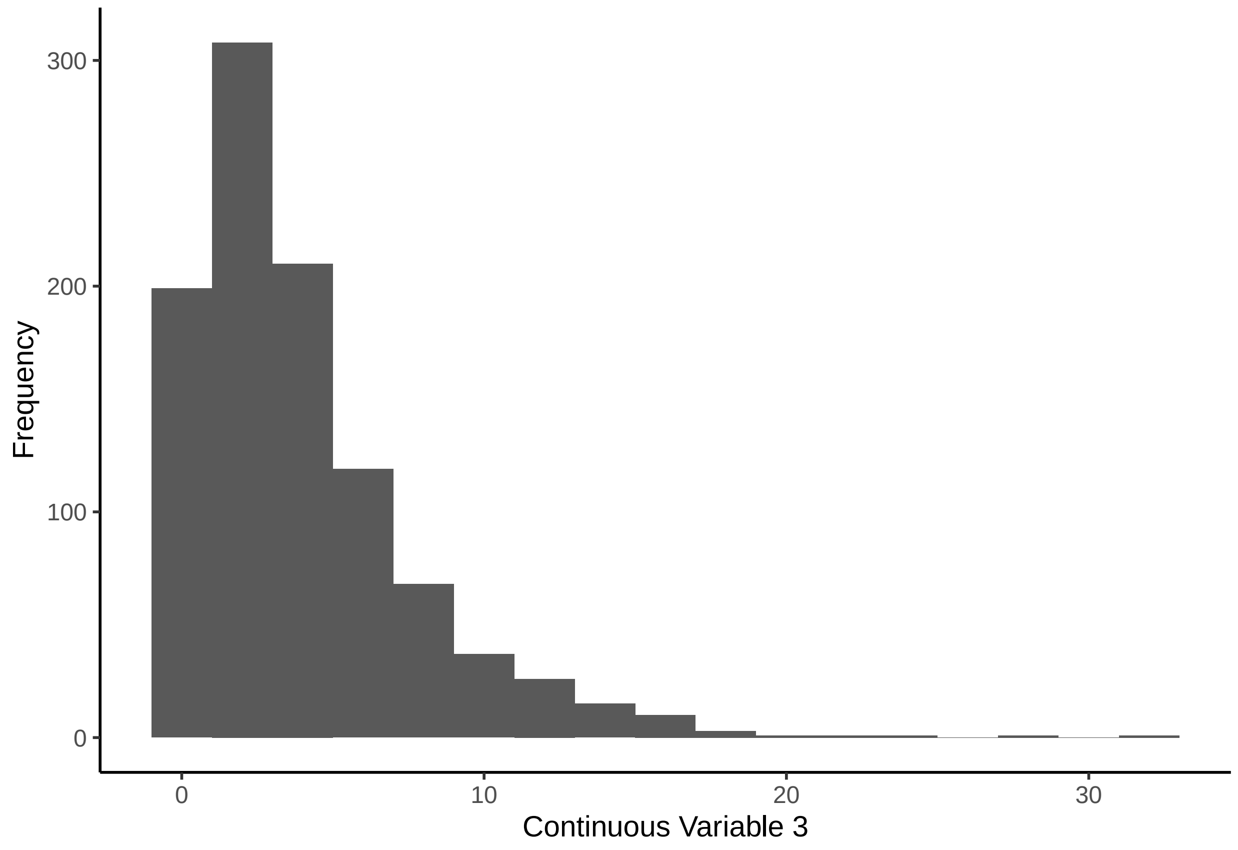
We can then combine these values to make them nicely formatted.

```r
print(paste0('Mean (sd): ',round(mean(randomData$Cont1),digits = 2),' (',
             round(sd(randomData$Cont1),digits = 2),')'))
```

```
## [1] "Mean (sd): 50.34 (9.61)"
```

For non-normally distributed continuous variables, we report median and interquartile range (IQR).

```r
ggplot(data = randomData,aes(x = Cont3)) +
  geom_histogram(binwidth = 2) +
  labs(x = 'Continuous Variable 3',y = 'Frequency') +
  theme_classic()
```



We can extract all the individual components of median and IQR using the following functions.

```r
# Median
median(randomData$Cont3)
```

```
## [1] 2.927949
```

```r
# Quartile values
quantile(randomData$Cont3)
```

```
##          0%         25%         50%         75%        100%
##  0.001975371  1.293967997  2.927949040  5.380989123 31.038499748
```

```r
# IQR (Q3 - Q1)
IQR(randomData$Cont3)
```

```
## [1] 4.087021
```

We can string these values together as follows to print out a clean and easy to read summary.

```r
print(paste0('Median [IQR]: ',round(median(randomData$Cont3),digits = 2),' [',
             round(quantile(randomData$Cont3,1/4),digits = 2),', ',
             round(quantile(randomData$Cont3,3/4),digits = 2),']'))
```

Table 1: Table 1

Variable	n = 40
Age mean (sd)	35 (2)
Sex n (%)	
Female	10 (25.0)
Male	30 (75.0)

```
## [1] "Median [IQR]: 2.93 [1.29, 5.38]"
```

Categorical Variables

For categorical covariates, we typically report the count (n) and percentage of each level of that variable.

```r
data.frame('n' = c(table(randomData$Cat1)),
           'perc' = c(round(table(randomData$Cat1)/1000*100,digits = 2)))
```

```
##         n perc
## FALSE 495 49.5
## TRUE  505 50.5
```

The "Table One"

In the epidemiology and data science communities, a table summarizing the demographics of a study population is typically referred to as a "table one". Such a table might look like this.

```r
library(knitr)
library(kableExtra)

# Define Table
table1 <- data.frame('Variable' = c('**Age** mean (sd)','**Sex** n (%)',
                                    "Female","Male"),
                     "Value" = c('35 (2)','','10 (25.0)','30 (75.0)'))

# Print table using kable
kable(x = table1, caption = 'Table 1',format = 'latex',
      col.names = c("Variable", "n = 40"),
      align = c('l','r'),escape = T) %>%
   add_indent(positions = c(3,4)) %>%
  kable_classic()
```

Refresher: Building a Function

To define a function in R, we use the following syntax:

```r
# Define function to calculate mean (sd)
meanSD <- function(x) {
  # Calculate individual values
  myMean <- mean(x)
  mySD <- sd(x)
  # Combine values
  paste0(round(myMean,digits = 2),' (',round(mySD,digits = 2),')')
}

meanSD(x = randomData$Cont1)
```

```
## [1] "50.34 (9.61)"
```

When we run a function, no intermediate values are saved. The only output from the function will be the final value you return.

We can also provide default values for terms in a function.

```r
# Define a function to calculate a mean or a median
contSummary <- function(x,normal = T) {

  # Calculate mean (sd) if normally distributed (the default)
  if (normal == T) {
      # Calculate individual values
    myMean <- round(mean(x),2)
    mySD <- round(sd(x),2)
    # Combine values
    paste0(myMean,' (',mySD,')')
  }
  # Calculate median (IQR) if non-normally distributed
  else {
    # Calculate individual values
    myMedian <- round(median(x))
    myIQR1 <- round(quantile(x,1/4),digits = 2)
    myIQR2 <- round(quantile(x,3/4),digits = 2)
    # Combine values
    paste0(myMedian,' [',myIQR1,', ',myIQR2,']')
  }
}

# Run function on normally distributed variable
contSummary(x = randomData$Cont1,normal = T)
```

```
## [1] "50.34 (9.61)"
```

```r
# Run function on non-normally distributed variable
contSummary(x = randomData$Cont3,normal = F)
```

```
## [1] "3 [1.29, 5.38]"
```

If we don't specify the "normal" term which we set a default for, the function will assume that it is normally distributed.

```r
contSummary(x = randomData$Cont1)
```

```
## [1] "50.34 (9.61)"
```

Table 2: Table 1

Variable	n = 1000
Cont1 Mean (sd)	50.34 (9.61)
Cont2 Mean (sd)	23.01 (1.96)
Cat1 n (%)	
FALSE	495 (49.5)
TRUE	505 (50.5)
Cont3 Median [IQR]	3 [1.29, 5.38]
Cat2 n (%)	
FALSE	745 (74.5)
TRUE	255 (25.5)
Cat3 n (%)	
A	352 (35.2)
B	491 (49.1)
C	157 (15.7)

```r
contSummary(x = randomData$Cont3)
```

```
## [1] "3.98 (3.76)"
```

In Class Activity

Design your own function to build a table one. Users should input a data frame and how different variables should be treated (i.e. normally or non-normally distributed, categorical).

R Script Dependencies

When organizing your code, sometimes it may be easier to define a function that you will be using across multiple scripts and then call it independently. We can do this using the *source* function in base R.

The source function will run through an entire R script and you will see all the generated variables and functions appear in your current R Environment. So, if you have a file with your saved functions, you can just run that file rather than defining the function in each R script that you need it in.

```r
# Define location of the file with the function on your computer
source('TableOne.R')

buildTableOne(data = randomData,varList = c('Cont1','Cont2','Cat1','Cont3','Cat2','Cat3'),
              nonnormVars = 'Cont3',catVars = c('Cat1','Cat2','Cat3'))
```