Assignment 2

Cameron Atkins

2023-09-29

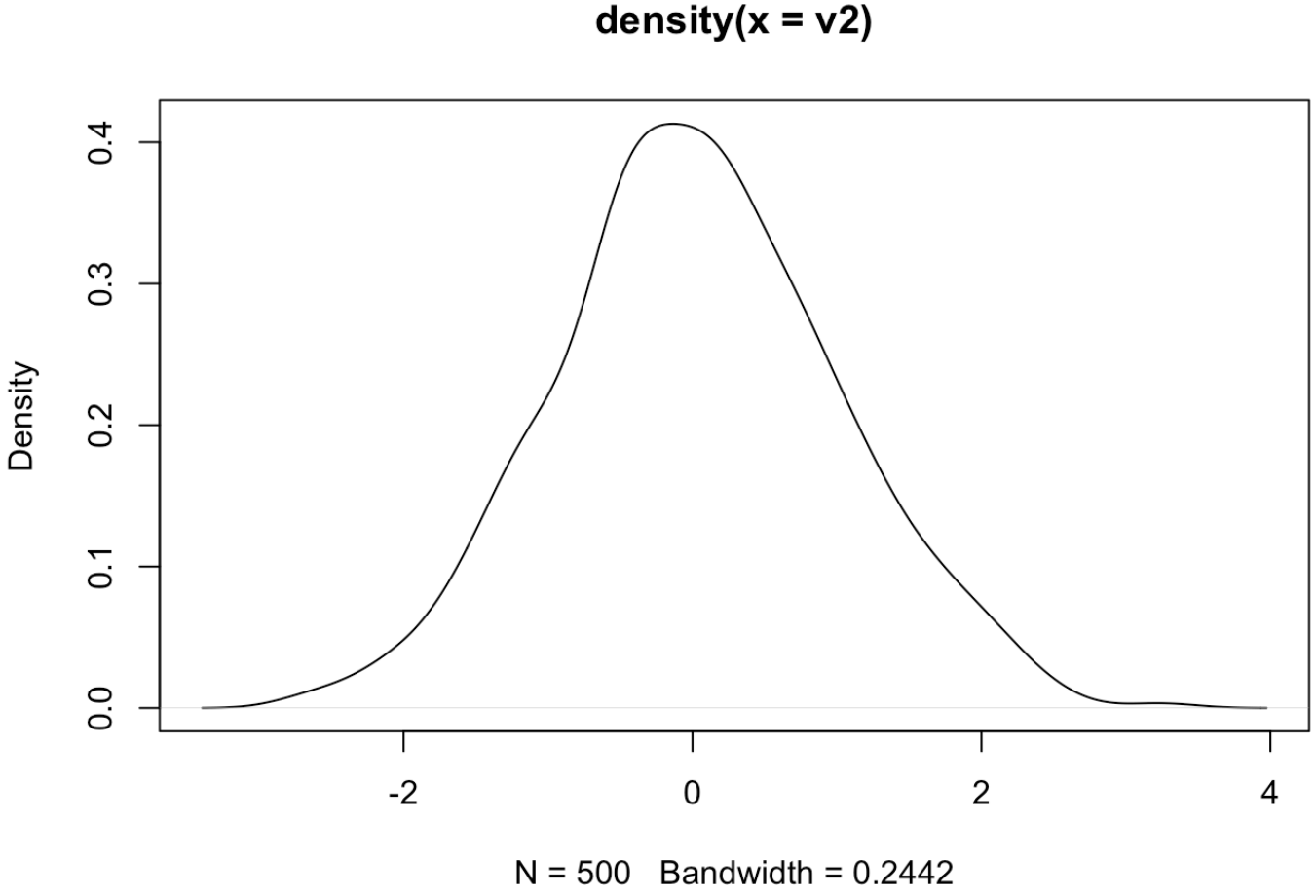
Problem 1

Build a function that takes a vector of size 500 (call the function and argument whatever you would like). If the vector is normally distributed, find the 95% confidence interval and print that to the console (use the cat() function). If the vector is not normally distrubted, print a message back to the console that the vector is not normally distributed. Regardless of distribution, plot the density of the vector.

Run a test case on both functions where the data is normally distributed and one where the data is not. The best way would be to use some r() functions (R has numerous distributions you can use to create vectors from with random values).

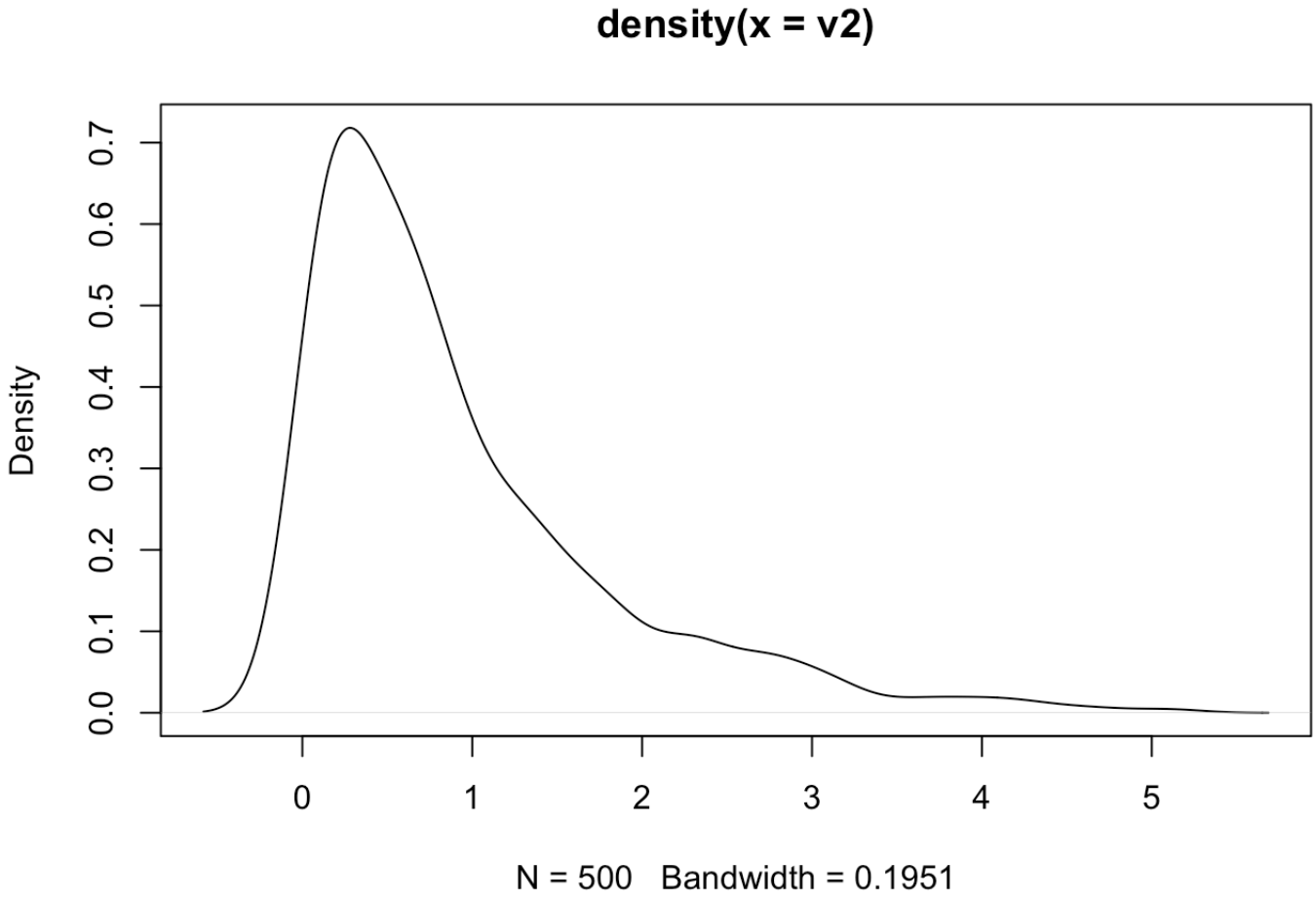
```
confInterval <- function(v1){
  vectorMean <- mean(v1)
  standardError <- (qnorm(0.975)*sd(v1))/(sqrt(length(v1)))
  confidence_interval <- c(vectorMean - standardError, vectorMean + standardError)
  cat("95% Confidence Interval:", confidence_interval, "\n")
}

checkIfNormal <- function(v2){
  isNormal <- shapiro.test(v2)

  if (isNormal$p.value >= 0.05){
    cat("Vector is normally distributed\n")
    confInterval(v2)
  }

  else{
    cat("Vector is not normally distributed\n")
  }

# The Density Plot of vector passed to function
  plot(density(v2))
}

# Test functions with data that is normal
set.seed(123)
isNormal <- rnorm(500)
checkIfNormal(isNormal)
```

```
## Vector is normally distributed
## 95% Confidence Interval: -0.05067498 0.1198559
```




```
# Test functions with data that is not normal
set.seed(456)
notNormal <- rexp(500, rate = 1)
checkIfNormal(notNormal)
```

```
## Vector is not normally distributed
```



Problem 2

Create a function that accepts a dataset. For each column in the dataset, print a statement “ is of type: ”. Test out your function on the dataset attached to this assignment (show the steps of reading in the dataset, you may need to use the str() function and make a class conversion for one of the variables).

Solution

```
# Read in the dataset
SampleData <- read.csv('dfex.csv')

# Structure of Sample data before converting Date Column
str(SampleData)
```

```
## 'data.frame':    5 obs. of  4 variables:
##  $ x3: int  1 2 3 4 5
##  $ x4: chr  "Dog" "Cat" "Horse" "Honey Bee" ...
##  $ x5: logi  TRUE TRUE FALSE TRUE FALSE
##  $ x6: chr  "2020-03-24" "2021-02-18" "1999-09-08" "1987-04-15" ...
```

```
# Converting Column x6 from chr to date
SampleData$x6 <- as.Date(SampleData$x6)

ColumnNameType <- function(ds){
  columns <- colnames(ds)

  for (name in columns){
    type <- class(ds[[name]])

    cat(name, "is of type: ", type, "\n")
  }
}

# Structure of Sample data after converting Date Column to type Date
str(SampleData)
```

```
## 'data.frame':    5 obs. of  4 variables:
##  $ x3: int  1 2 3 4 5
##  $ x4: chr  "Dog" "Cat" "Horse" "Honey Bee" ...
##  $ x5: logi  TRUE TRUE FALSE TRUE FALSE
##  $ x6: Date, format: "2020-03-24" "2021-02-18" ...
```

```
# Test the function with the sample dataset
ColumnNameType(SampleData)
```

```
## x3 is of type:  integer
## x4 is of type:  character
## x5 is of type:  logical
## x6 is of type:  Date
```

Problem 3

What does each function do? What does each function return? Come up with better names for these functions

Solution

A. function f1 accepts two parameters, a string and prefix. The function returns true if “string” begins with “prefix” and false if it doesn’t. A better name for this function would be beginsWithPrefix. Below is an example of fuction f1 in use.

```
f1 <- function(string, prefix) {
  substr(string, 1, nchar(prefix)) == prefix
}

f1("mathematics", "math")
```

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

```
f1("mathematics", "biol")
```

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

B. function f2 accepts one parameter, a vector or list. If the list has only one element, the function return NULL. If the list or vector has more than one element, the last element is removed.

A better name for this function would be removeLastElement.

Below are two instances of function f2 executing.

```
f2 <- function(x) {
  if (length(x) <= 1) return(NULL)
  x[-length(x)]
}

names <- c("Djokovic", "Federer", "Nadal", "Tiafoe", "Alcaraz", "Murray")
nums <- c(4005)

f2(names)
```

```
## [1] "Djokovic" "Federer"  "Nadal"    "Tiafoe"   "Alcaraz"
```

```
f2(nums)
```

```
## NULL
```

C. function f3 accepts two vectors as parameters. Using the built-in replicate function(rep), function f3 returns the second vector, y, matching the number of elements in the first vector, x. If the length of the y is less than the length of x, then elements in y will be repeated to until its length matches the length of x. If the length of x is greater than the length of y, elements in y will be removed to match the length of x.

A better name for this function would be matchesLength.

Below are four instances of f3 executing with vectors of various lengths.

```
f3 <- function(x, y) {
rep(y, length.out = length(x))
}

a <- c(1, 2, 3, 4, 5)
b <- c(6)
c <- c(7, 8, 9, 10, 11, 12, 13, 14, 15)

f3(a, b)
```

```
## [1] 6 6 6 6 6
```

```
f3(a, c)
```

```
## [1]  7  8  9 10 11
```

```
f3(b, c)
```

```
## [1] 7
```

```
f3(c,a)
```

```
## [1] 1 2 3 4 5 1 2 3 4
```