STA445_Assignment_2

Tucker Harris

2023-10-10

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Question 1 a

```r
uniform <- function( a, b, x )
  {

  exDensity = 0

  if ( a <= x && x <= b){
    exDensity = 1 / ( b - a )
  }

  print( paste( 'density = ', exDensity ) )

  }

uniform( 5, 6, 4 )
```

```
## [1] "density =  0"
```

```r
uniform( 4, 6, 5 )
```

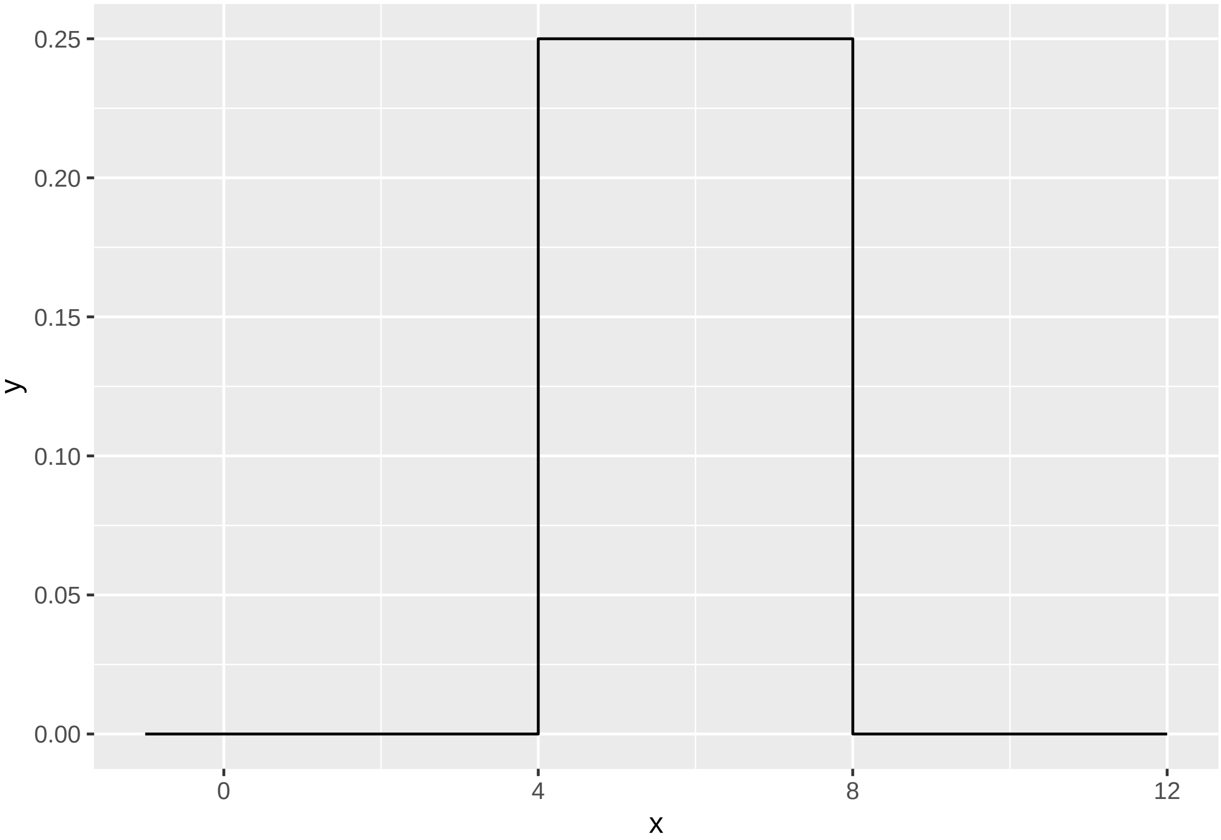
```
## [1] "density =  0.5"
```

```r
uniform( 4, 5, 6 )
```

```
## [1] "density =  0"
```

Question 1 b

```r
duniform <- function(x, a, b){
  output <- NULL
  for( i in 1:length(x) ){  # Set the for loop to look at each element of x
    if( x[i] >= a & x[i] <= b ){
      output[i] = 1 / (b - a)
    }

    else{
      output[i] = 0
    }
  }
  return(output)
}

data.frame( x=seq(-1, 12, by=.001) ) %>%
  mutate( y = duniform(x, 4, 8) ) %>%
  ggplot( aes(x=x, y=y) ) +
  geom_step()
```



Question 1 c

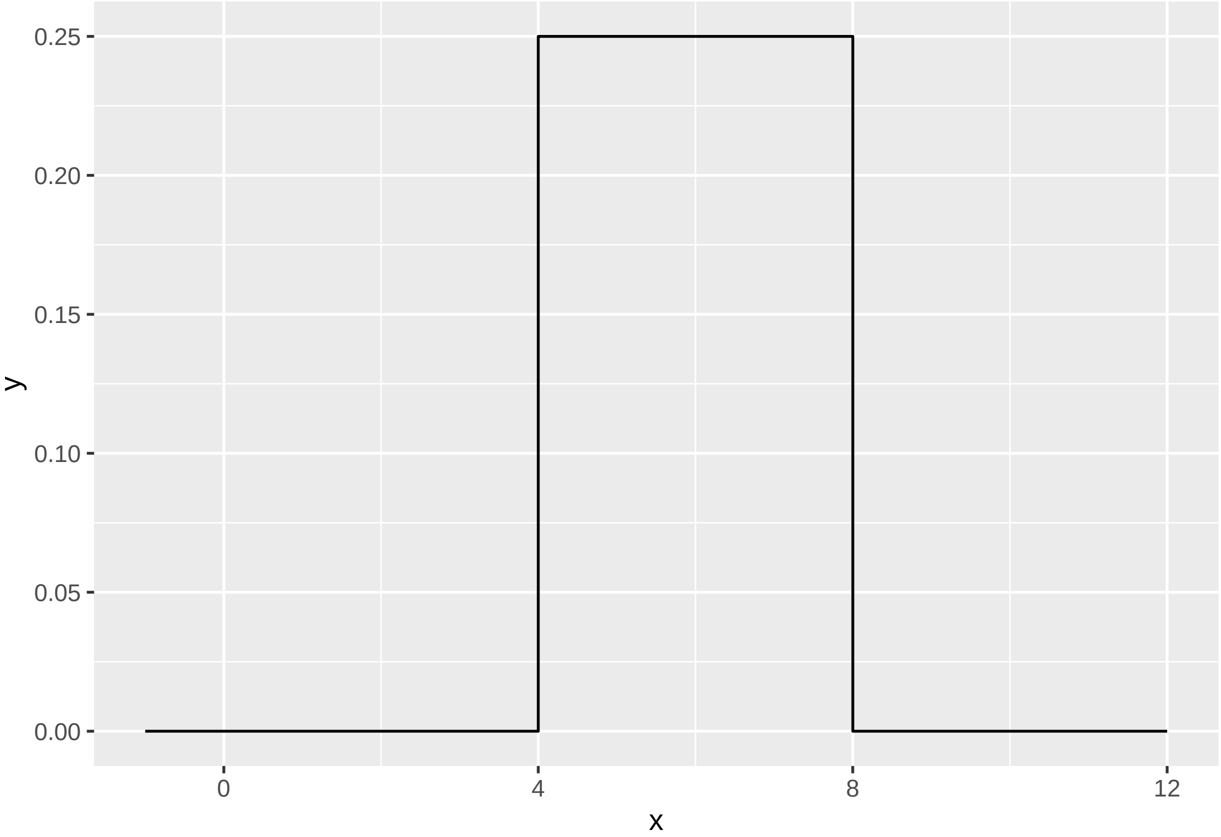
Install microbenchmark

```
library(microbenchmark)

microbenchmark::microbenchmark( duniform( seq(-4,12,by=.0001), 4, 8), times=100)
```

```
## Unit: milliseconds
##                                      expr      min       lq     mean   median
##  duniform(seq(-4, 12, by = 1e-04), 4, 8) 55.3554 61.8581 64.46261 63.0783
##        uq      max neval
##  65.81695 121.4844   100
```

Question d

```
duniform <- function(x, a, b){
  output <- NULL

  output = ifelse(a <= x & x <= b, 1 / ( b - a ), 0 )

  return(output)
}

data.frame( x=seq(-1, 12, by=.001) ) %>%
  mutate( y = duniform(x, 4, 8) ) %>%
  ggplot( aes(x=x, y=y) ) +
  geom_step()
```



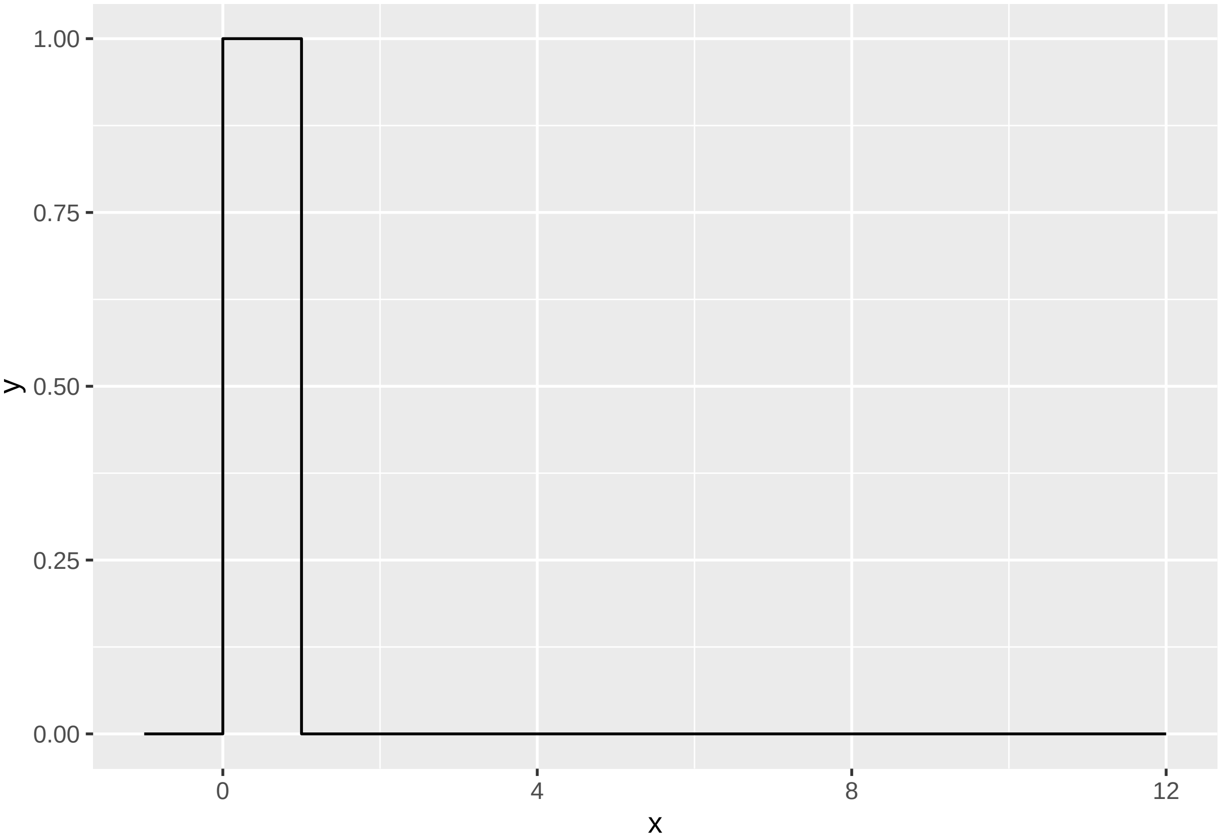
```
microbenchmark::microbenchmark( duniform( seq(-4,12,by=.0001), 4, 8), times=100)
```

```
## Unit: milliseconds
##                                      expr    min     lq     mean median     uq
##  duniform(seq(-4, 12, by = 1e-04), 4, 8) 4.2414 4.5519 7.236214 5.9951 7.9625
##       max neval
##  104.4564   100
```

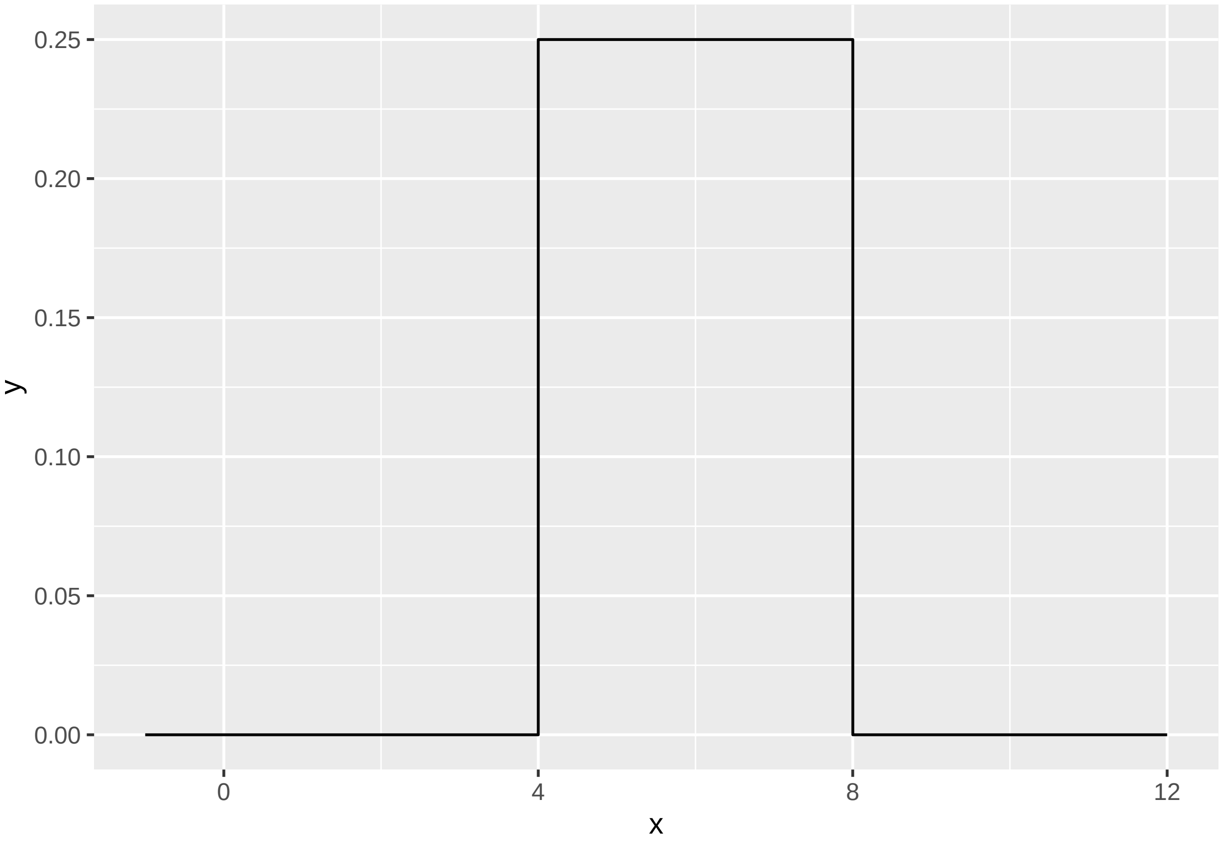
Using the ifelse call, reduced the median by 57.37855 milliseconds. This means the code was reduced in both time and length of code. The ifelse is far more efficient in both facets.

Question 2

```
duniform <- function( x, a = 0, b = 1 ){
  output <- NULL

  output = ifelse(a <= x & x <= b, 1 / ( b - a ), 0 )

  return(output)
}

data.frame( x=seq(-1, 12, by=.001) ) %>%
  mutate( y = duniform(x,) ) %>%
  ggplot( aes(x=x, y=y) ) +
  geom_step()
```

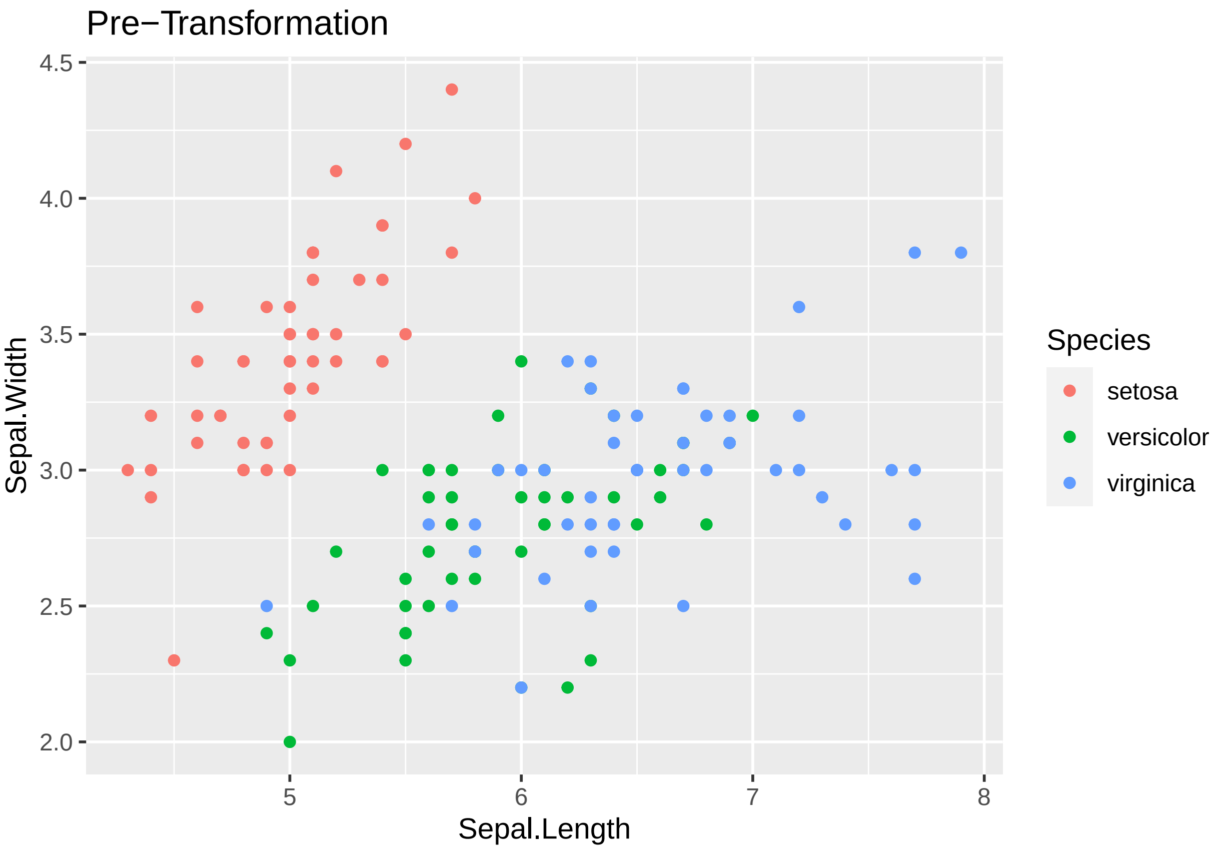


```
data.frame( x=seq(-1, 12, by=.001) ) %>%
  mutate( y = duniform(x, a = 4, b = 8) ) %>%
  ggplot( aes(x=x, y=y) ) +
  geom_step()
```

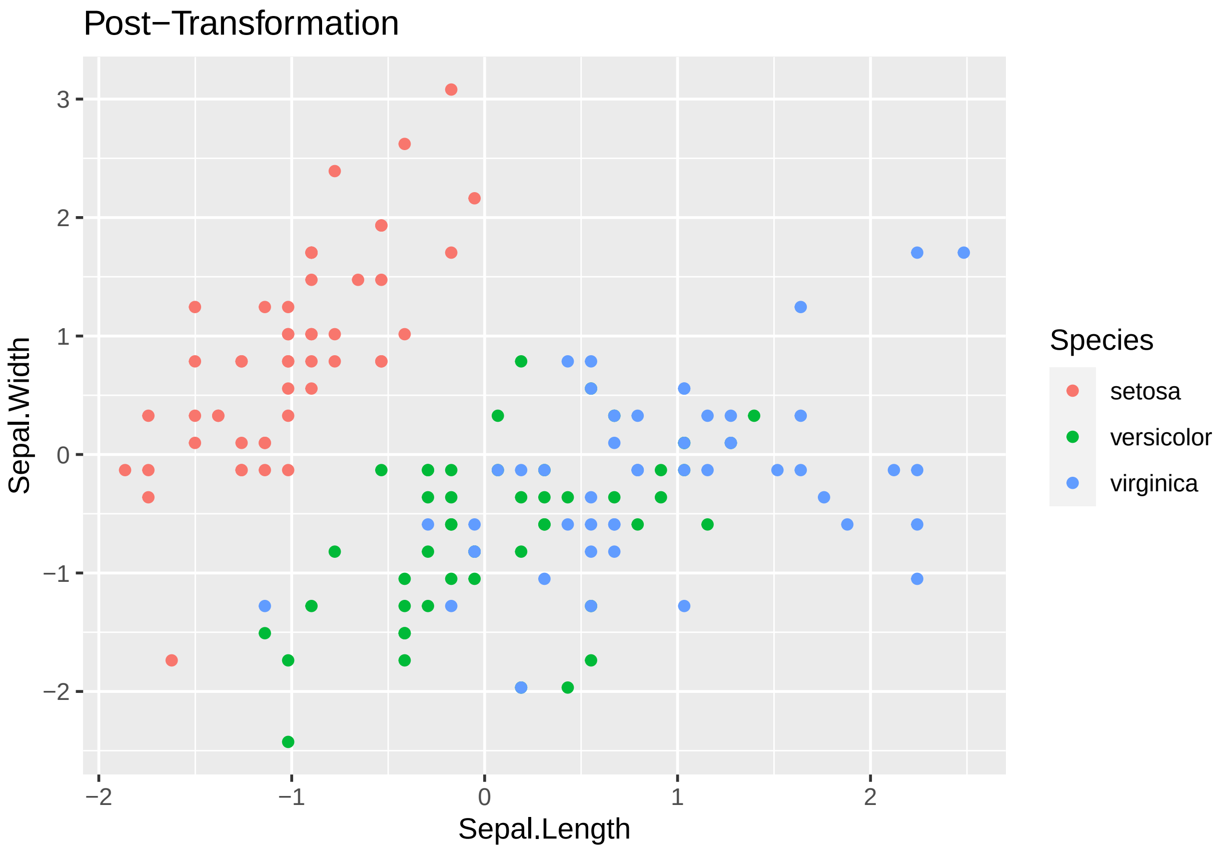


## Question 3

```
standardize <- function(x)
  {
    s = sd(x)
    z = ( x - mean( x ) ) / s

    return(z)
  }

data( 'iris' )
# Graph the pre-transformed data.
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point() +
  labs(title='Pre-Transformation')
```



```r
# Standardize all of the numeric columns
# across() selects columns and applies a function to them
# there column select requires a dplyr column select command such
# as starts_with(), contains(), or where().  The where() command
# allows us to use some logical function on the column to decide
# if the function should be applied or not.
iris.z <- iris %>% mutate( across(where(is.numeric), standardize) )

# Graph the post-transformed data.
ggplot(iris.z, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point() +
  labs(title='Post-Transformation')
```



Question 4

```
fizzBuzz <- function(numbers)
{

  newVector <- c()

  for(i in 1:length( numbers) )
  {
    if(numbers[i] %% 3 == 0 & numbers[i] %% 5 == 0){
      newVector[i] = "Fizzbuzz "
    } else if (numbers[i] %% 3 == 0){
      newVector[i] = "Fizz"
    } else if(numbers[i] %% 5 == 0){
      newVector[i] = "Buzz"
    }
    else{
      newVector[i] = numbers[i]
    }


  }

  return( newVector )

}
```

```r
fizzBuzz( 1:16 )
```

```
##  [1] "1"         "2"         "Fizz"      "4"         "Buzz"      "Fizz"     
##  [7] "7"         "8"         "Fizz"      "Buzz"      "11"        "Fizz"     
## [13] "13"        "14"        "Fizzbuzz " "16"
```

Question 5

```r
test.vector <- c('A',NA,NA, 'B','C', NA,NA,NA)


myFill <- function( x )
{
  # Create a loop that checks each index
  for( index in 1:length( x ) )
  {
    if( is.na( x[ index ] ) )
    {
      x[ index ] = x[ index - 1 ]
    }
  }

  return( x )
}

myFill(test.vector)
```

```
## [1] "A" "A" "A" "B" "C" "C" "C" "C"
```