Assignment 3

M Hill

10/11/2024

Exercise 1

(a)

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

  n <- NULL

  if (x >= a & x <= b)
  {
    n <- (1/b-a)
  }
  else
  {
    n <- 0
  }

  return(n)

}

duniform(2,6,8)
```

```
## [1] 0
```

```
duniform(7,6,8)
```

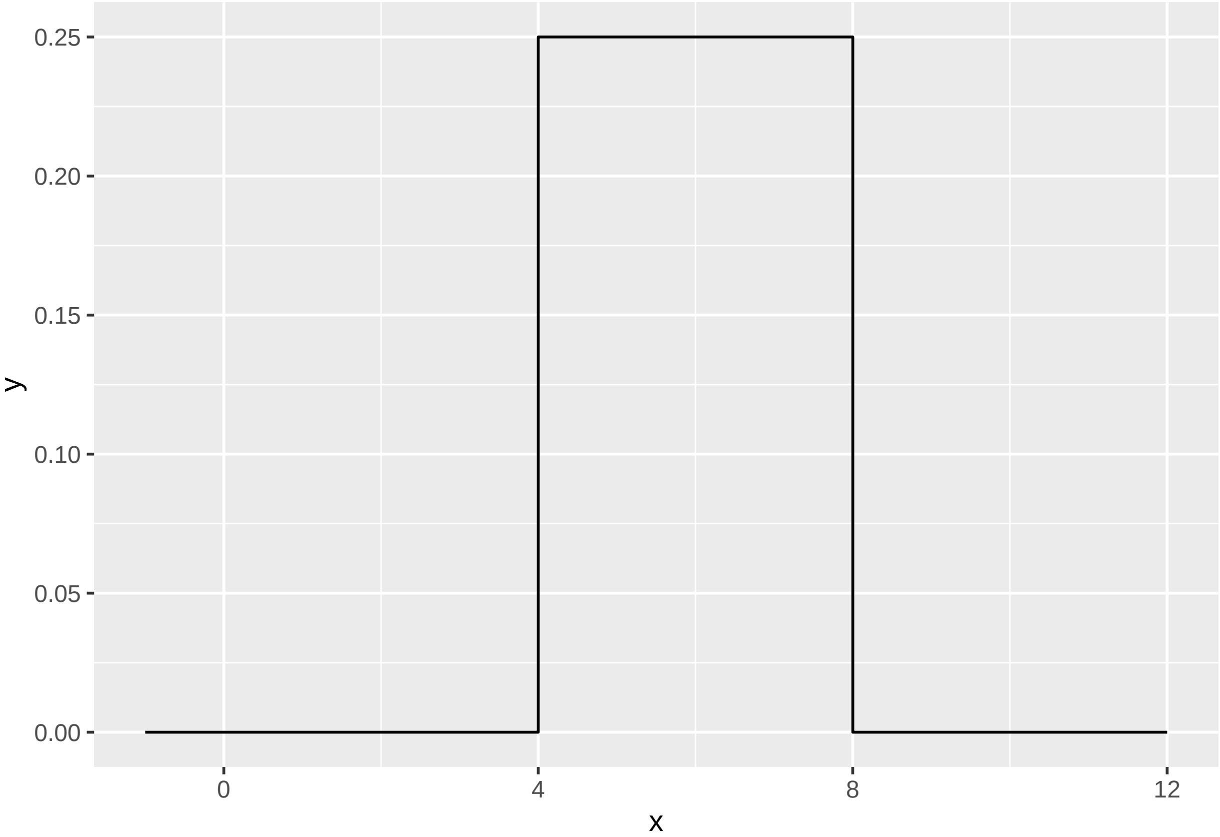
```
## [1] -5.875
```

```
duniform(9,6,8)
```

```
## [1] 0
```

(b)

```
duniform <- function(x,a,b)
{
  n <- c()
  for (i in 1:length(x))
  {
    if (x[i] >= a & x[i] <= b)
    {
      n[i] <- (1/(b-a))
    }
    else
    {
      n[i] <- 0
    }
  }

  return(n)

}

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



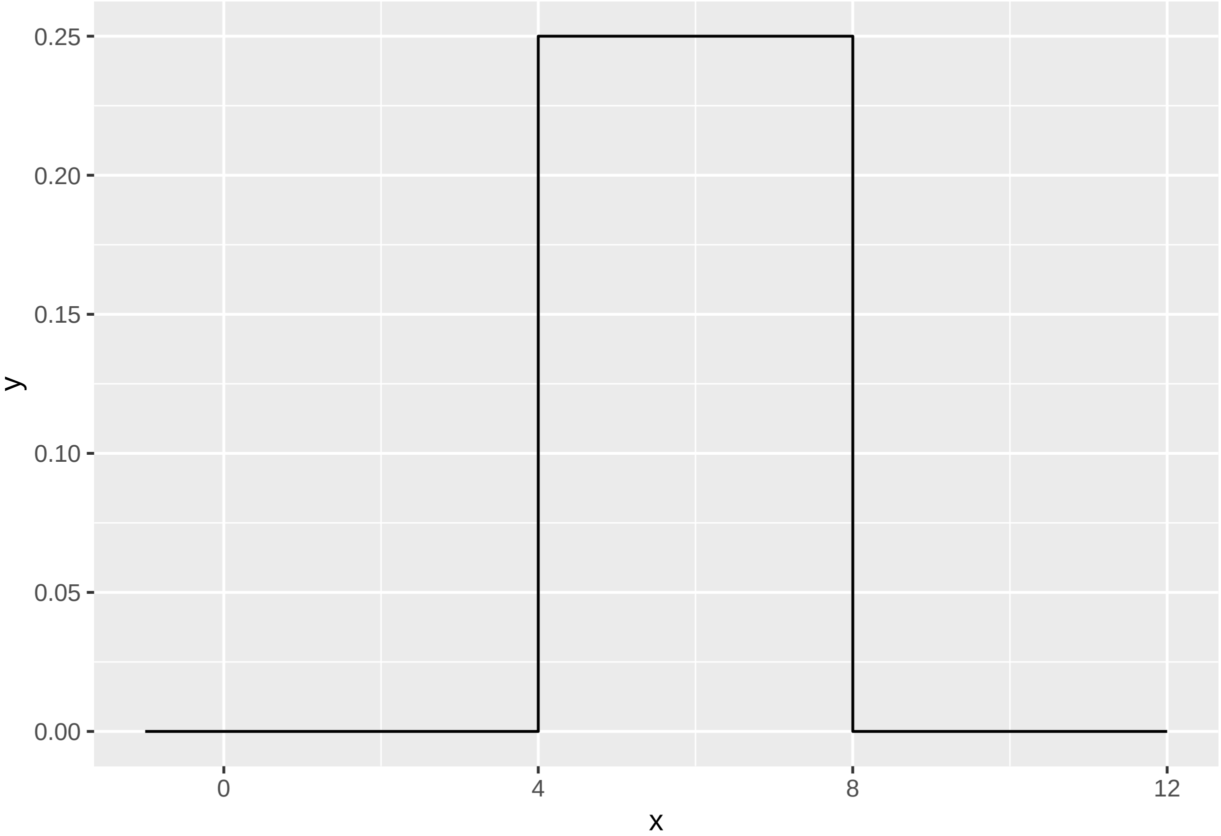
(c)

```
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) 58.4279 60.5576 63.96366 61.82495
##        uq      max neval
##  64.21005 117.6809   100
```

(d)

```
duniform <- function(x,a,b)
{
  n <- ifelse(x >= a & x <= b,1/(b-a),0)
  return(n)
}

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) 4.0842 7.98965 11.06605 9.2527 10.9551
##      max neval
##  97.8744   100
```

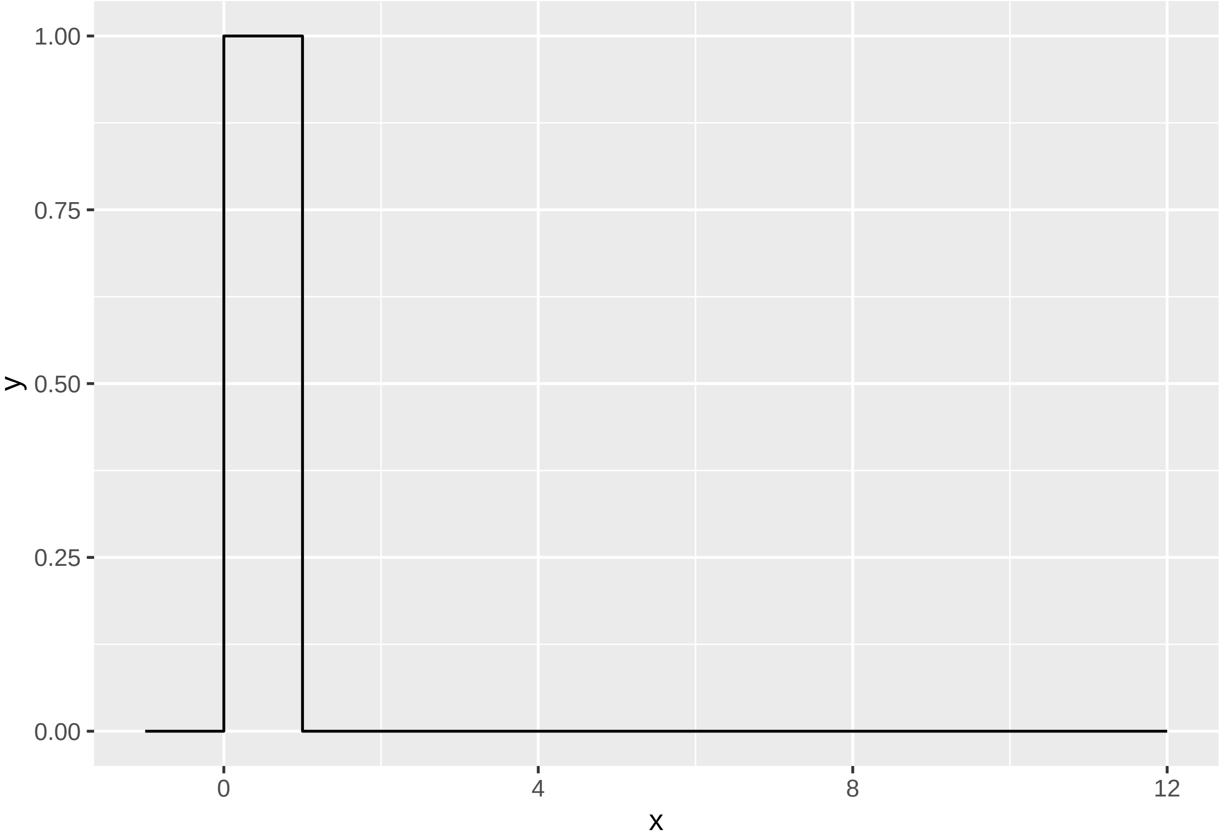
(e)

The ifelse was easier to write than the for loop, once i got some of the syntax down. And it ran much faster than the for loop I wrote.

Exercise 2

```
duniform <- function(x,a=0,b=1)
{
  n <- ifelse(x >= a & x <= b,1/(b-a),0)
  return(n)
}
```

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



Setting a=0 and b=1, creates default that the function will use when it is called without those arguments.

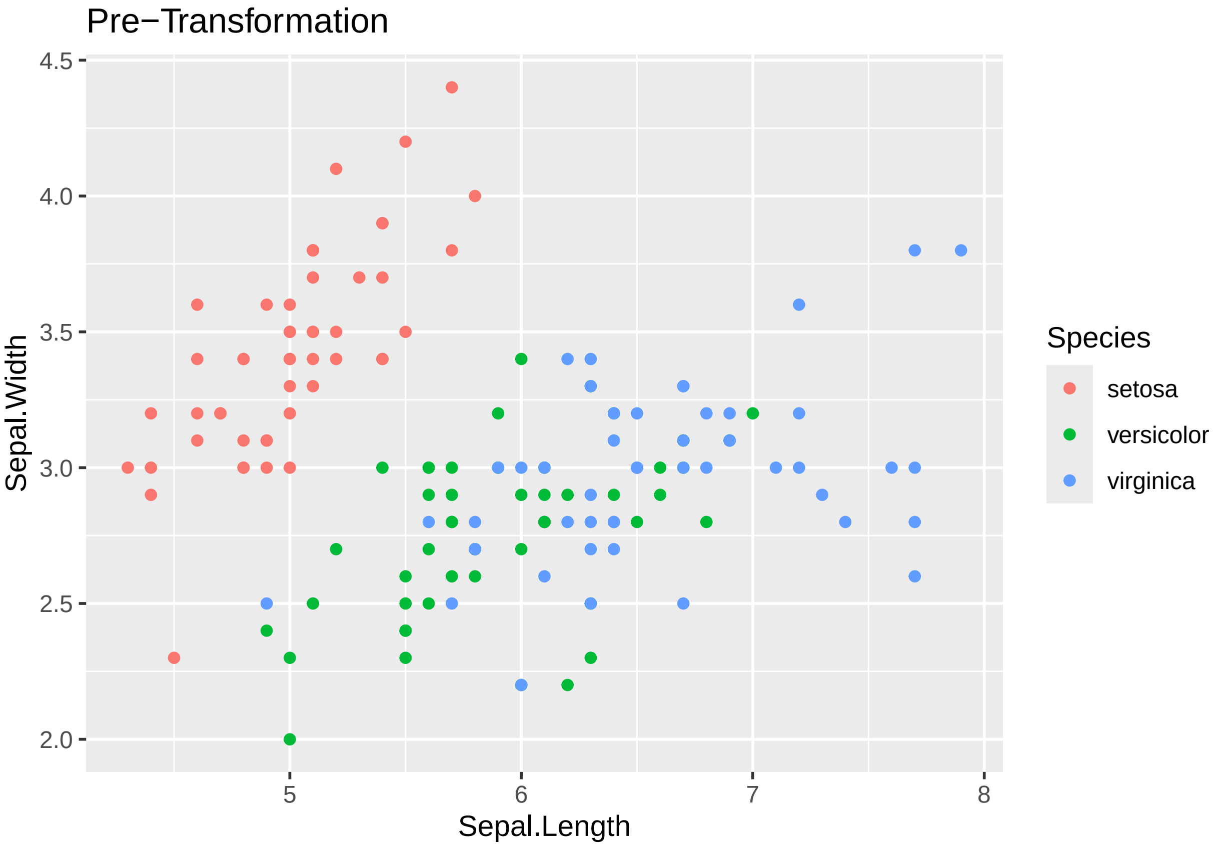
Exercise 3

(a)

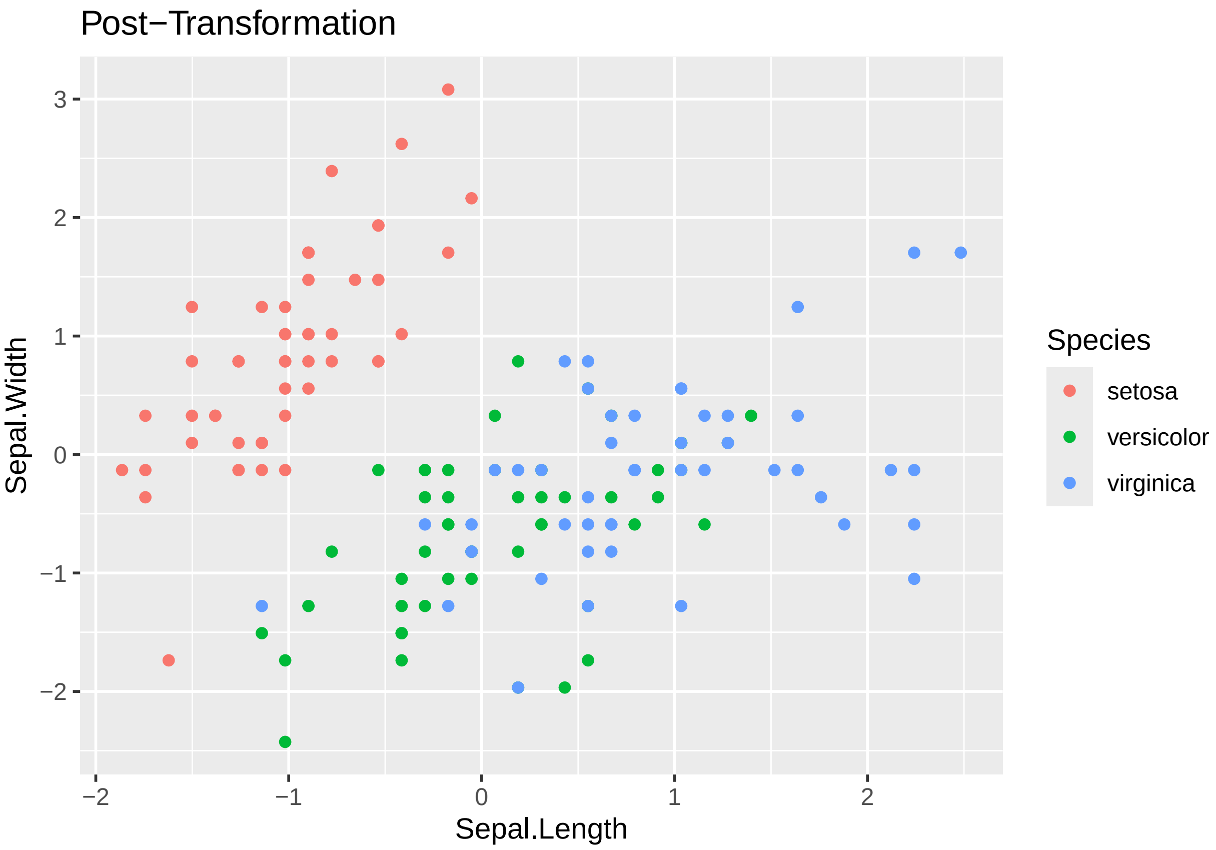
```r
standardize <- function(x)
{
  n <- c()
  mtn <- mean(x)
  std <- sd(x)
  for (i in 1:length(x))
  {

    n[i] <- (x[i] - mtn)/std

  }

  return(n)
}
```

(b)

```r
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')
```



I created a vector within the function that would hold and then return the standardized numeric variables.

Exercise 4

```
fizzBuzz <- function(n)
{
  fizzList <- c()
  for (i in 1:n)
  {
    if (i %% 15 == 0)
    {
      fizzList[i] = "Fizz-Buzz"
    }
    else if (i %% 5 == 0)
    {
      fizzList[i] = "Buzz"
    }
    else if (i %% 3 == 0)
    {
      fizzList[i] = "Fizz"
    }
    else
    {
      fizzList[i] = i
    }
  }
  return(fizzList)
}
fizzBuzz(30)
```

```
##  [1] "1"         "2"         "Fizz"      "4"         "Buzz"      "Fizz"
##  [7] "7"         "8"         "Fizz"      "Buzz"      "11"        "Fizz"
## [13] "13"        "14"        "Fizz-Buzz" "16"        "17"        "Fizz"
## [19] "19"        "Buzz"      "Fizz"      "22"        "23"        "Fizz"
## [25] "Buzz"      "26"        "Fizz"      "28"        "29"        "Fizz-Buzz"
```

The fizzBuzz function loops from one to n. Then, based on the if and else statements places the results, either “Fizz”, “Buzz”, “Fizz-Buzz”, and the number itself into a vector that is returned.