Assignment 2

Evan Z-G

2023-10-10

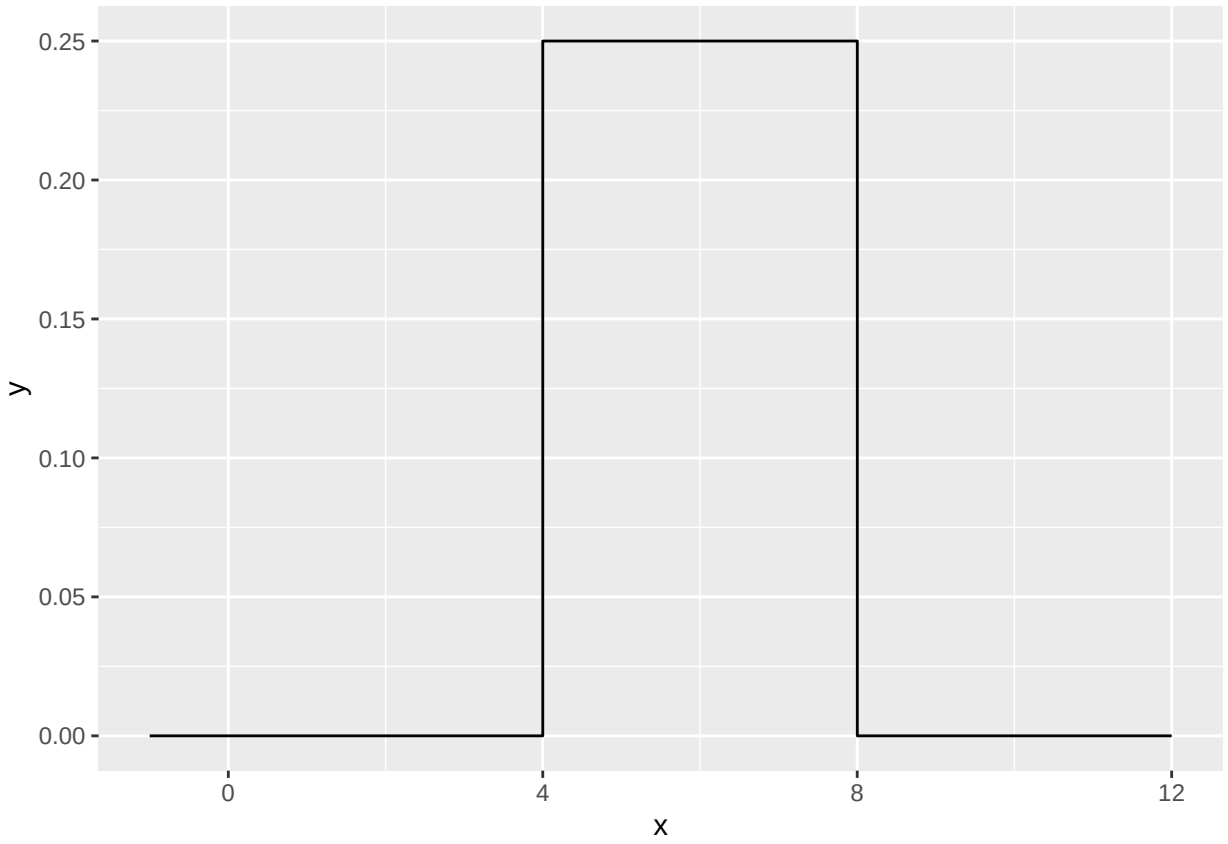
1.a.

```
# define function duniform:
duniform <- function(x,a,b){
  if( a<=x &  x<=b ){
    density <- 1/(b-a)
  }else{
    density <- 0
  }
  return(density)
}

# test function duniform:
duniform(4,2,5)
```

```
## [1] 0.3333333
```

1.b.

```
# define function duniform:
duniform <- function(x,a,b){
  output <- NULL
  for( i in (1:length(x)) ){
    if( a<=x[i] & x[i]<=b ){
      output[i] <- 1/(b-a)
    }
    else{
      output[i] <- 0
    }
  }
return(output)
}

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



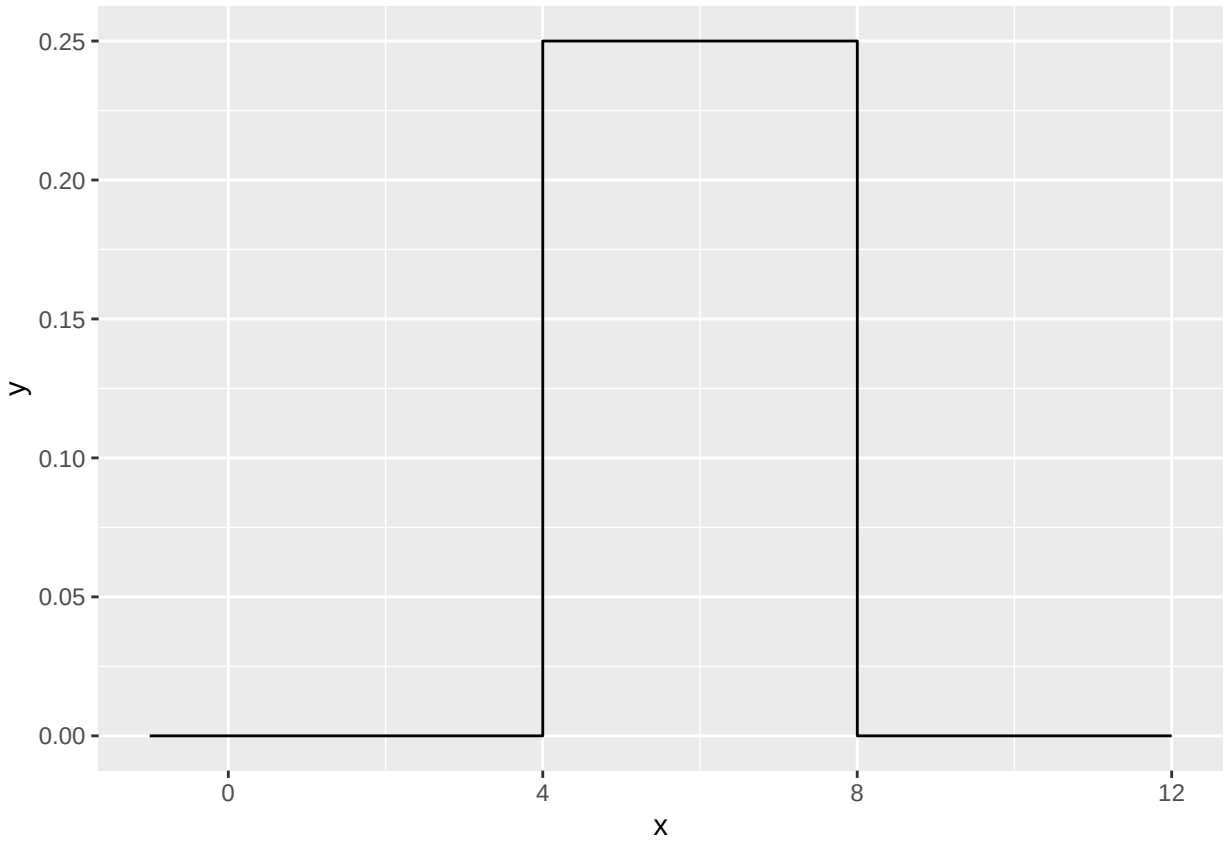
1.c.

```
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) 55.5639 61.23855 63.40445 61.8361
##        uq      max neval
##  64.41925 119.5704   100
```

1.d.

```
# define function duniform:
duniform <- function(x,a,b){
  output <- NULL
    output <- ifelse( (a<=x & x<=b), 1/(b-a), 0 )
return(output)
}

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



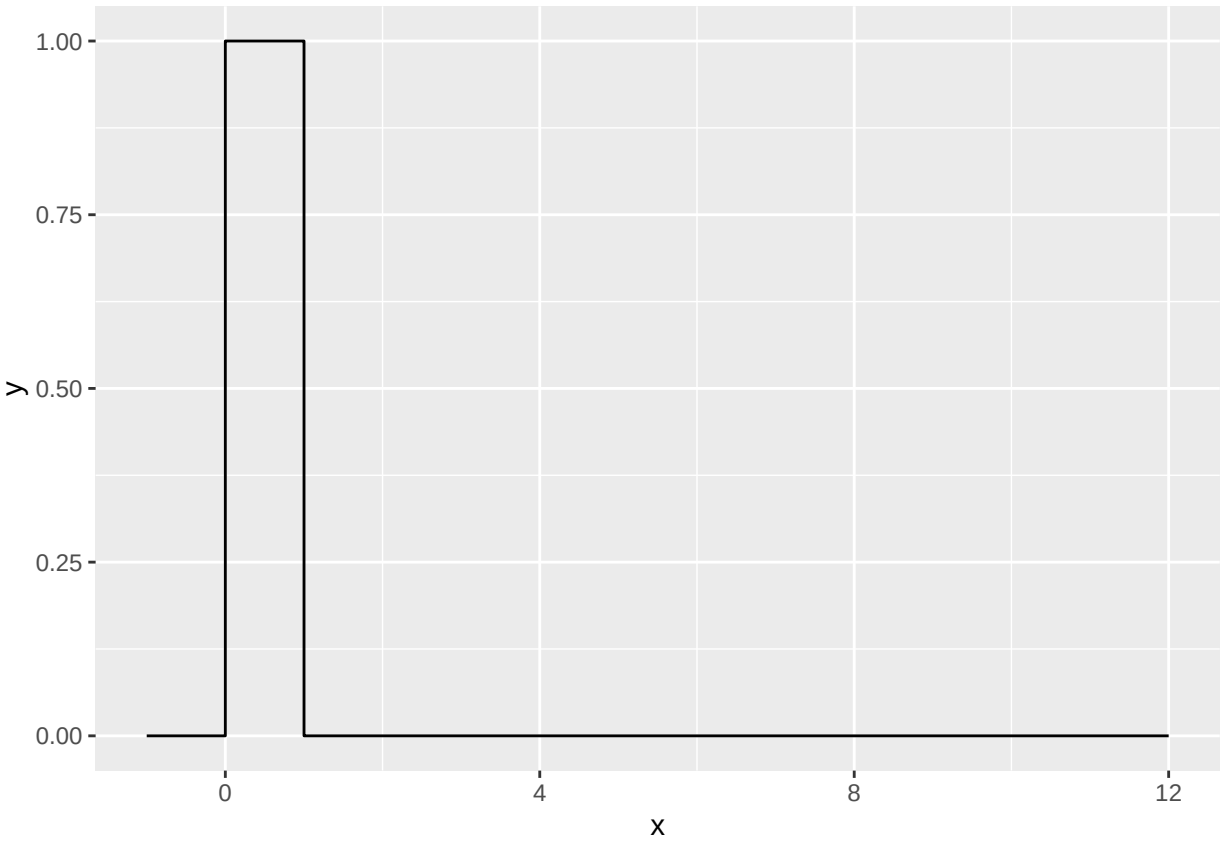

```
library(microbenchmark)
microbenchmark::microbenchmark( duniform( seq(-4,12,by=.0001), 4, 8), times=100)
```

```
## Unit: milliseconds
##                                       expr    min     lq    mean median     uq
##  duniform(seq(-4, 12, by = 1e-04), 4, 8) 4.1717 4.3947 7.07987  5.777 7.7343
##       max neval
##  100.7615   100
```

The second version was easier to write, however the first version was faster.

2.

```
# define function duniform:
duniform <- function(x,a=0,b=1){
  output <- NULL
    output <- ifelse( (a<=x & x<=b), 1/(b-a), 0 )
return(output)
}

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

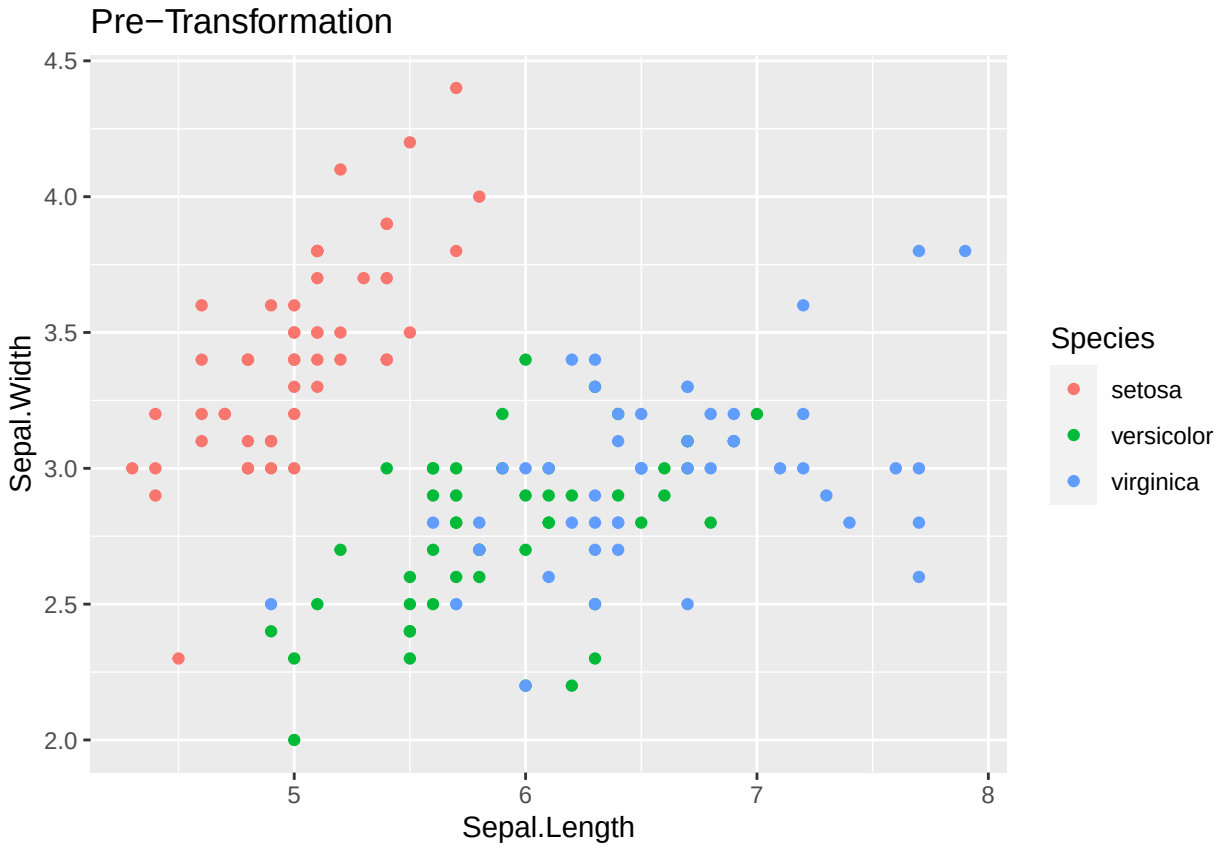



```
library(microbenchmark)
microbenchmark::microbenchmark( duniform( seq(-4,12,by=.0001)), times=100)
```

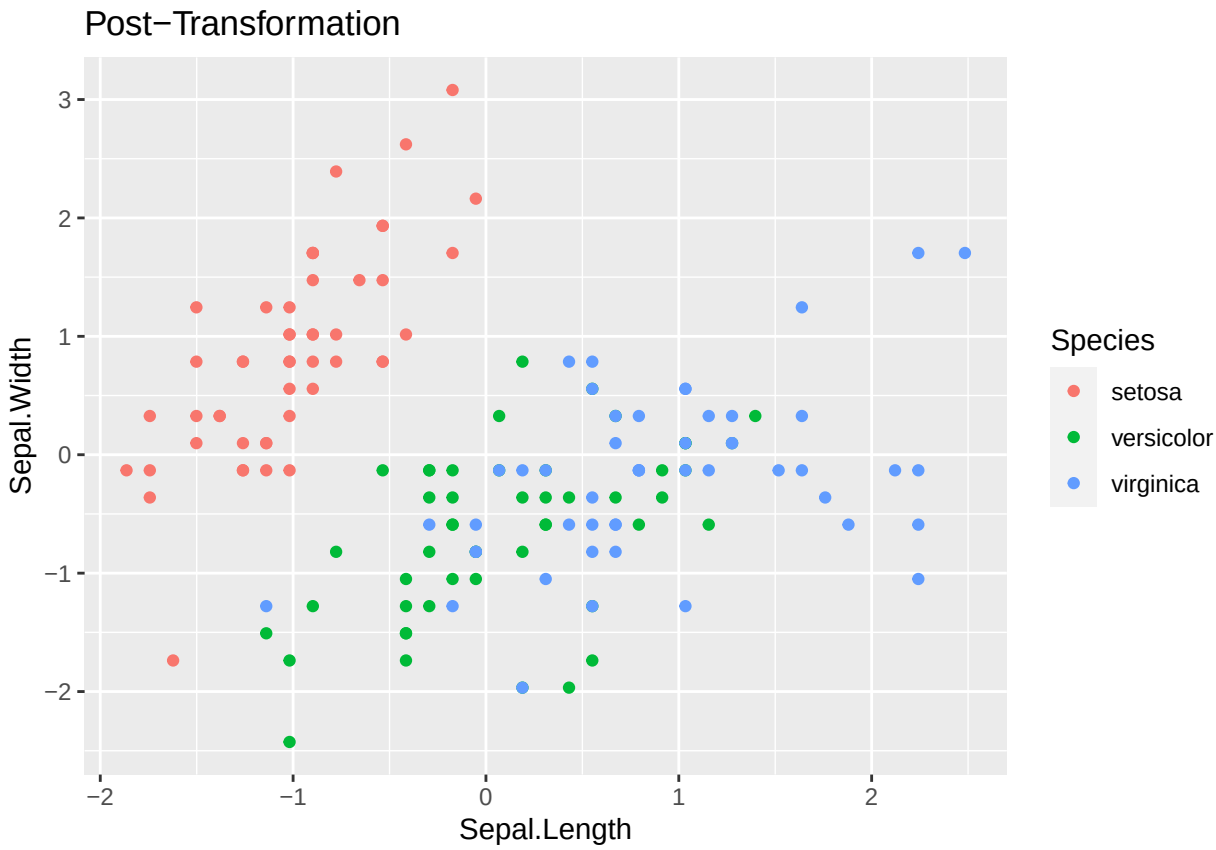
```
## Unit: milliseconds
##                               expr    min     lq   mean  median      uq
##  duniform(seq(-4, 12, by = 1e-04)) 4.1687 4.5111 7.1541 5.89295 8.02185
##       max neval
##  102.0813   100
```

3.

```
# define function standardize:
standardize <- function(x){
  output <- NULL
  for (i in 1:length(x)){
    output[i] <- (x[i] - mean(x)) / sd(x)
  }
  return(output)
}

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




```
# 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')
```



4.

```
fizzBuzz <- function(n){
  output = ""
  for (i in 1:n){
    if (i %% 3 == 0 & i %% 5 == 0){
      output <- paste(output, "Fizz Buzz")
    }
    else if (i %% 3 == 0){
      output <- paste(output, "Fizz")
    }
    else if (i %% 5 == 0){
      output <- paste(output, "Buzz")
    }
    else {
      output <- paste(output, as.character(i))
    }
  }
  return(output)
  }

fizzBuzz(15)
```

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

5.

```
myFill <- function(x){
  new_vect = c()
  for( i in (1:length(x)) ){
    if (is.na(x[i])){
      new_vect[i] <- new_vect[i-1]
    }
    else {new_vect[i] <- x[i]
    }
  }
  return(new_vect)
}

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

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