Assignment 3

Isaac Bragna

2024-10-18

Exercise 1

Write a function that calculates the density function of a Uniform continuous variable on the interval (a,b).

```
duniform <- function(x, a, b){
  density <- case_when(a<x&b>x ~ 1/(b-a),
                       x<a|x>b ~ 0)
  return(density)
}
duniform(1, 2, 8)      # x<a
```

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

```
duniform(11, 6, 15)    # a<x<b
```

```
## [1] 0.1111111
```

```
duniform(13, 3, 7)     # b<x
```

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

Part B

```
duniform2 <- function(x, a, b){
  density <- ifelse(a<x&b>x, 1/(b-a), 0)
  return(density)
}
duniform2(c(0,2,4,6), 1, 5)
```

```
## [1] 0.00 0.25 0.25 0.00
```

```
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) 7.3711 9.17 11.61373 10.58895 12.7423
##      max neval
##  70.2433   100
```

```
microbenchmark::microbenchmark( duniform2( seq(-4,12,by=.0001), 4, 8), times=100)
```

```
## Unit: milliseconds
##                                          expr    min      lq     mean median     uq
##  duniform2(seq(-4, 12, by = 1e-04), 4, 8) 4.4243 6.3199 7.999308 6.9196 8.5017
##      max neval
##  64.4526   100
```

Exercise 2

For your duniform() function provide default values of 0 and 1 for the arguments a and b. Demonstrate that your function is appropriately using the given default values by producing a graph of the density without specifying the a or b arguments.

```
duniform <- function(x, a=0, b=1){
  density <- case_when(a<x&b>x ~ 1/(b-a),
                       x<a|x>b ~ 0)
  return(density)
}
duniform(0.5)
```

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

```
duniform(4)
```

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

```
duniform(11,6,15)
```

```
## [1] 0.1111111
```

Exercise 3

Create a function that takes an input vector of numerical values and produces an output vector of the standardized values.

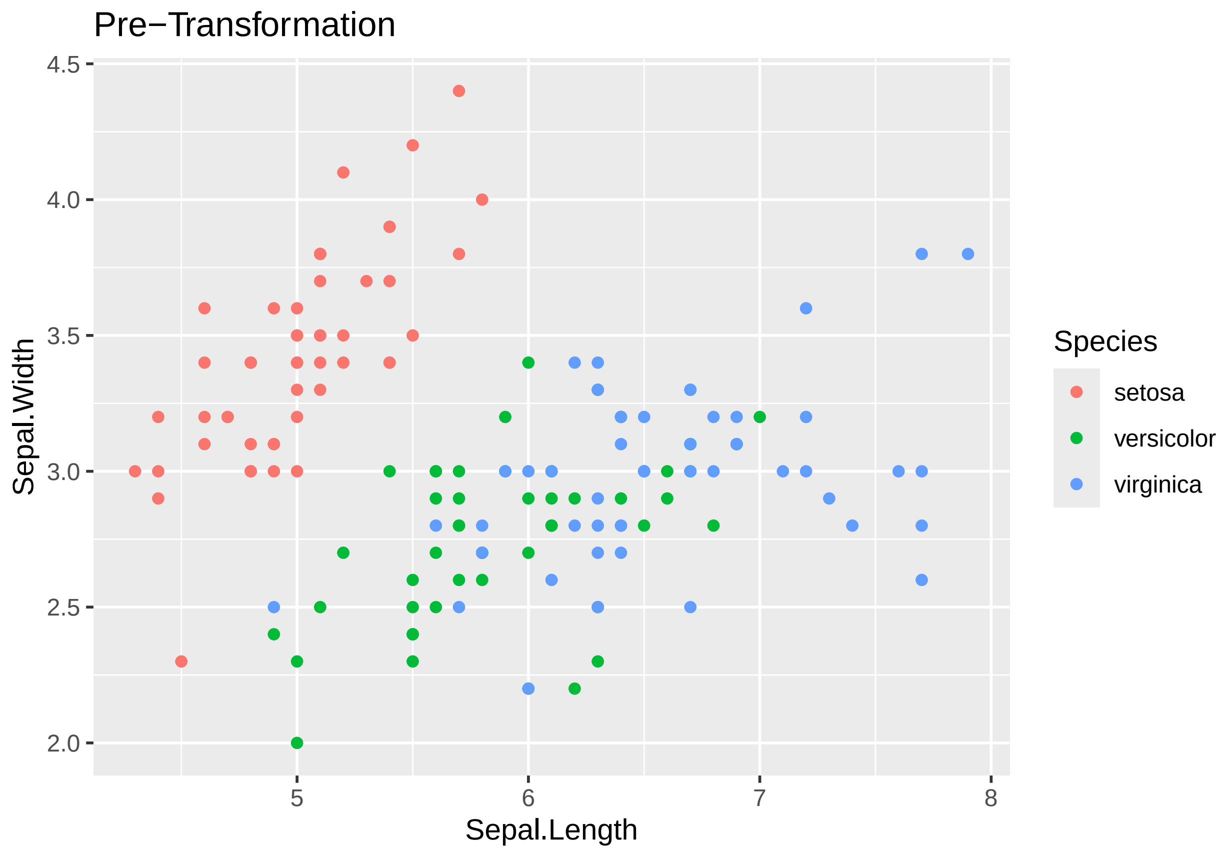
```
standardize <- function( x ){
  s_dev <- sd(x)
  xbar <- mean(x)
  stdzd <- c()
  for( i in x ){
    z = (i-xbar)/s_dev
    stdzd <- c(stdzd, z)
  }
  return(stdzd)
}
standardize( c(1,2,3,4,5,6))
```

```
## [1] -1.3363062 -0.8017837 -0.2672612  0.2672612  0.8017837  1.3363062
```

Apply this function to each numeric column in a data frame

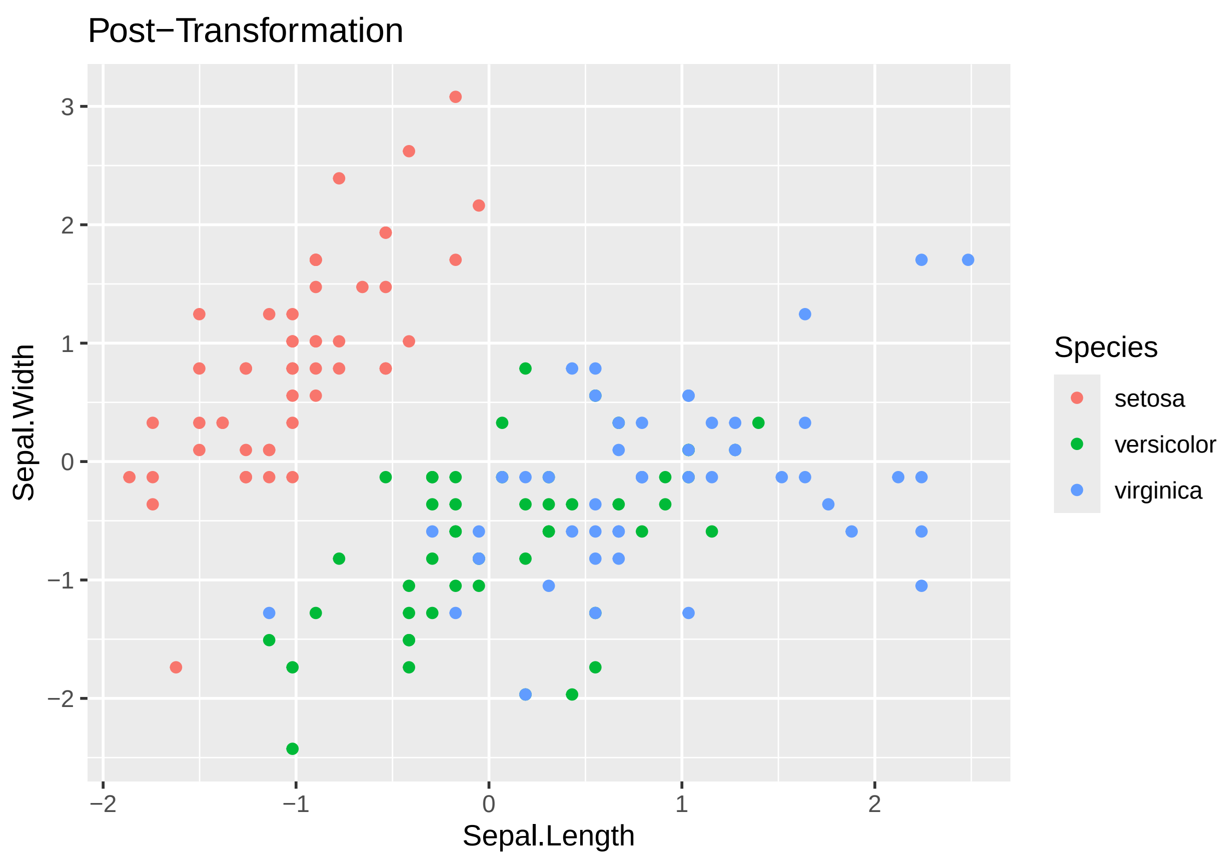
```
data( 'iris' )

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



```
iris.z <- iris %>% mutate( across(where(is.numeric), standardize) )

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



Exercise 4

```r
fizz_buzz <- function(n){
  list <- seq(n)
  FB_list <- NULL
  for( i in list ){
    fate <- case_when( i %% 15 == 0 ~ 'Fizz-Buzz',
                       i %% 5 == 0 ~ 'Buzz',
                       i %% 3 == 0 ~ 'Fizz',
                       .default =as.character(i))
    FB_list <- paste(FB_list, fate, sep=' ')
  }
  return(FB_list)
}
fizz_buzz(25)
```

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