

1. Consider the following integral:

$$\int_a^b (7 - 2x^2)dx.$$

While this is a relatively straightforward integral – split the difference and use power rule for integration – we can approximate the area under this curve using Riemann sums. Below, I describe four rules for computing Riemann sums using rectangles of width δ_x .

- Left Rule uses the left points to create the rectangles.

$$\text{Area} = \delta_x (f(a) + f(a + \delta_x) + f(a + 2\delta_x) + \cdots + f(b - \delta_x))$$

- Right Rule uses the right points to create the rectangles.

$$\text{Area} = \delta_x (f(a + \delta_x) + f(a + 2\delta_x) + \cdots + f(b - \delta_x) + f(b))$$

- Midpoint Rule uses the midpoints between the left and right points to create the rectangles.

$$\text{Area} = \delta_x \left(f\left(a + \frac{\delta_x}{2}\right) + f\left(a + \frac{3\delta_x}{2}\right) + \cdots + f\left(b - \frac{\delta_x}{2}\right) \right)$$

- Trapezoidal Rule averages the rectangles created using left and right endpoints, which results in areas of trapezoids.

$$\text{Area} = \frac{1}{2} \delta_x (f(a) + 2f(a + \delta_x) + 2f(a + 2\delta_x) + \cdots + f(b))$$

The first step, is to create a function that computes the integrand.

```
integrand <- function(x){  
  f <- 7 - 2 * x^2  
  return(f)  
}
```

Next, I set up an example choice of a , b , and the number of rectangles.

```
a <- 0  
b <- 2  
n.rect <- 100  
(delta.x <- (b-a)/n.rect)  
  
## [1] 0.02
```

We can compute the area using the Left Rule as follows.

```
left.points <- a + 0:99*(delta.x)  
(left.area <- sum(delta.x*(integrand(left.points))))  
  
## [1] 8.7464
```

We can compute the area using the Right Rule as follows.

```
right.points <- a + 1:100*(delta.x)  
(right.area <- sum(delta.x*(integrand(right.points))))  
  
## [1] 8.5864
```

We can compute the area using the Midpoint Rule as follows.

```
mid.points <- (left.points+right.points)/2  
(mid.area <- sum(delta.x*(integrand(mid.points))))  
  
## [1] 8.6668
```

- (a) Write code that computes the area using the Trapezoidal Rule.

Solution:

```
# Write code for answer here.
#Reiterating the bounds and questions
a = 0 #Lower Bound
b = 2 #Upper Bound
integrand = function(x){ #Making the Function
  f <- 7 - 2 * x^2
  return(f)
}

#Number of trapezoids
numb.of.trap = 100
#Setting up the width
width.of.trap = (b-a)/numb.of.trap

#Points for trapezoid
point.for.trap = a + 1:99*(width.of.trap)

#Formula for the trapezoid

0.5*width.of.trap*sum(integrand(a) + integrand(b) + sum(2*integrand(point.for.trap)))

## [1] 8.6664
```

- (b) Write a function that takes a, b, and number of rectangles (n.rect) in as input and returns the Trapezoidal Rule by default, but can return left, right, or midpoint when necessary. Use the skeleton below. When you are done, remove eval=FALSE to show that the function provides the expected result for the example above.

Solution:

```
riemann.sums <- function(fnct,                # function to integrate
                        a,                    # lower bound of integral
                        b,                    # upper bound of integral
                        n.rect,               # number of bound of integral
                        method = "Trapezoidal"){ # method to use (trap by default)

  #####
  # Check Input
  #####
  if(!is.numeric(a)){ # if a is not numeric
    stop("The lower bound of the integral (a) must be numeric.")
  }
  if(!is.numeric(b)){ # if b is not numeric
    stop("The lower bound of the integral (a) must be numeric.")
  }
  if(!(is.numeric(n.rect)) | (n.rect%%1!=0)){ # if n.rect is not a whole number
    stop("The number of rectangles must be a positive whole number.")
  }
  #####
  # Compute Area
  #####
  if(method == "Left"){
    # Add necessary code here
    delta.x = (b-a)/n.rect
    left.points <- a + (0:(n.rect-1))*(delta.x)
    area = (left.area <- sum(delta.x*(fnct(left.points))))
  }else if(method == "Right"){
    # Add necessary code here
    delta.x = (b-a)/n.rect
    right.points <- a + (1:(n.rect))*(delta.x)
    area = (right.area <- sum(delta.x*(fnct(right.points))))
  }else if(method == "Midpoint"){
    # Add necessary code here
    delta.x = (b-a)/n.rect
    left.points <- a + (0:(n.rect-1))*(delta.x)
    right.points <- a + (1:(n.rect))*(delta.x)
    mid.points <- (left.points+right.points)/2
    area = (mid.area <- sum(delta.x*(fnct(mid.points))))
  }else if(method == "Trapezoidal"){
    # Add necessary code here
    delta.x = (b-a)/n.rect
    point.for.trap = a + (1:(n.rect-1))*(delta.x)
    area = 0.5*delta.x*sum(fnct(a) + fnct(b) + sum(2*fnct(point.for.trap)))
  }else{
    stop("Please select a valid method (e.g., 'Left', 'Right', 'Midpoint', 'Trapezoidal')")
  }
  #####
  # Return the area
}
```

```

#####
    return(area)
}
#####
# Test the function
#####
riemann.sums(fnct = integrand,
             a = 0,
             b = 2,
             n.rect = 100)

## [1] 8.6664

#####
# Compare to numerical integral
#####
integrate(f = integrand, # integrate() is an R function
          lower = 0,      # that completes numerical
          upper = 2)      # integration

## 8.666667 with absolute error < 9.8e-14

```