Coding Examples

1. Basic Integration Example

```
# Define the function to integrate
f <- function(x) { x^2 }
# Perform integration
result <- integrate(f, lower = 0, upper = 1)
# Print result
print(result$value)
```

output:

```
> f <- function(x) {x^2}
> result <- integrate(f,lower=0,upper=1)
> print(result$value)
[1] 0.3333333
```

2. Advanced Integration Example Using pracma

```
library(pracma)
# Define the function to integrate
f <- function(x) { sin(x) }
# Perform integration
result <- integral(f, lower = 0, upper = pi)
# Print result
print(result)
```

output:

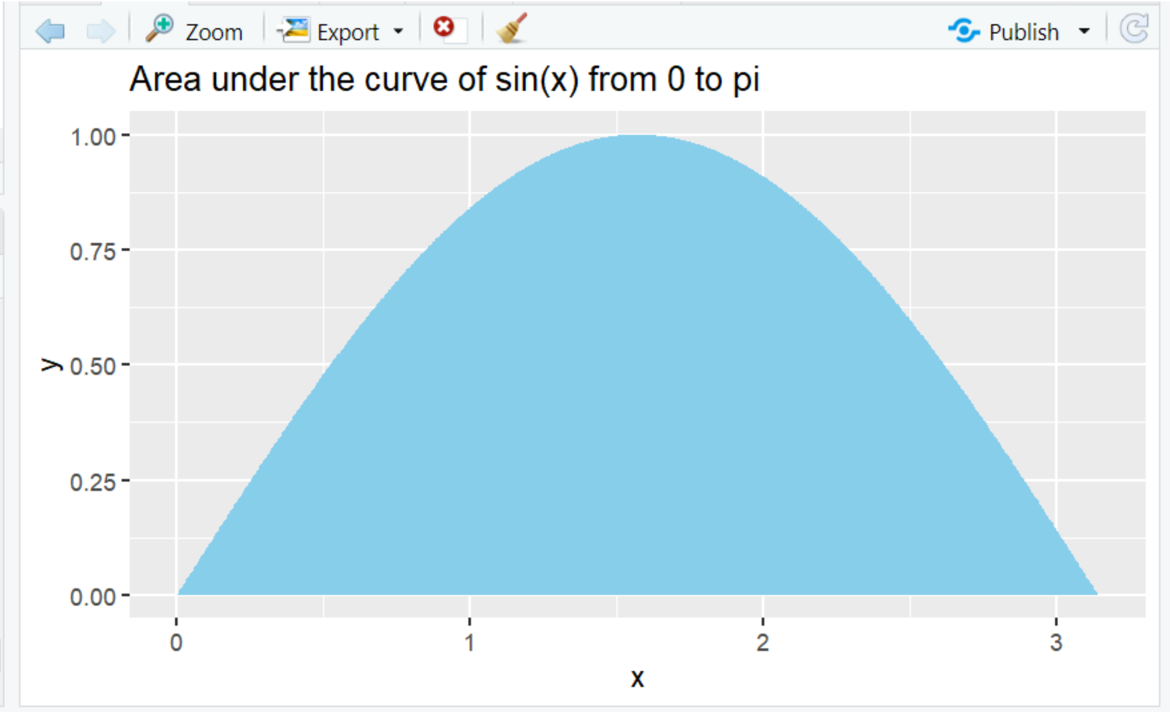
```
f<-function(x) {sin(x)}
> r<-integrate(f,lower=0,upper=pi)
> print(r)
2 with absolute error < 2.2e-14
```

3. Visualizing the Integration Result

```
library(ggplot2)
```

```
# Define the function and create a sequence of x values
f <- function(x) { sin(x) }
x_values <- seq(0, pi, by = 0.01)
y_values <- f(x_values)
# Create a data frame
data <- data.frame(x = x_values, y = y_values)
# Plot the area under the curve
ggplot(data, aes(x = x, y = y)) +
geom_area(fill = "skyblue") +
ggtitle("Area under the curve of sin(x) from 0 to π")
```

output:



Exercises

1. Exercise 1: Integrate a Polynomial Function

o Task: Integrate the function f(x)=3x3+2x2+x+1f(x) = 3x^3 + 2x^2 + x + 1f(x)=3x3+2x2+x+1 from 0 to 2.

o Visualize: Plot the function and the area under the curve.

o Expected Output:

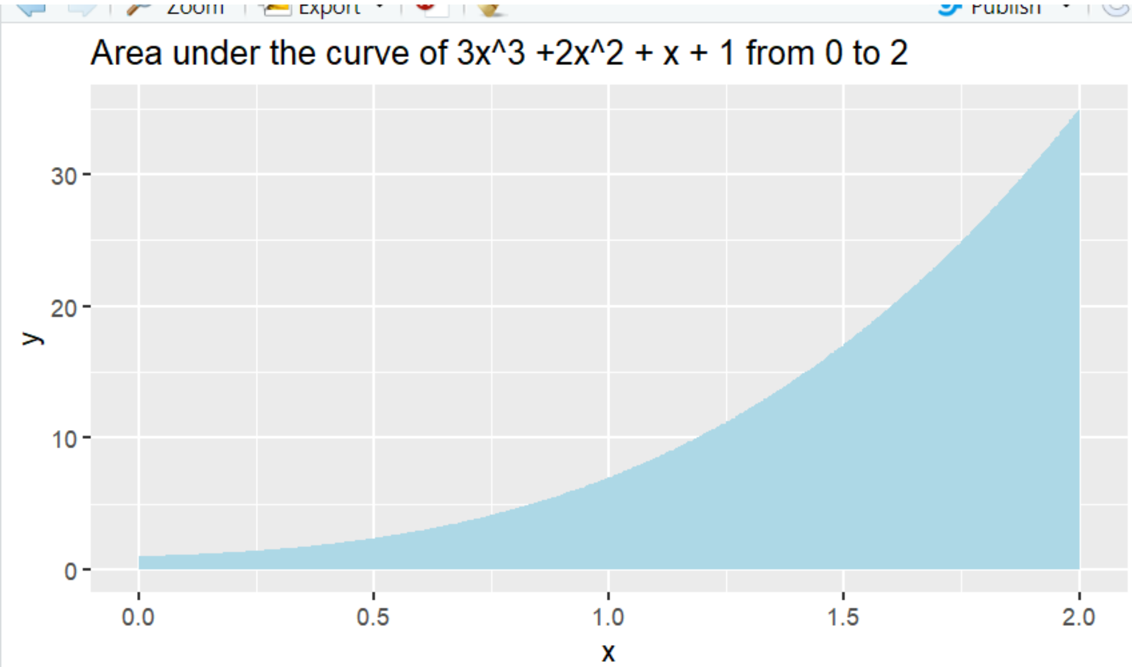
```
# Define the function
```

```
f <- function(x) { 3*x^3 + 2*x^2 + x + 1 }
# Perform integration
result <- integrate(f, lower = 0, upper = 2)
print(result$value)
# Visualization
x_values <- seq(0, 2, by = 0.01)
y_values <- f(x_values)
data <- data.frame(x = x_values, y = y_values)
ggplot(data, aes(x = x, y = y)) +
geom_area(fill = "lightblue") +
ggtitle("Area under the curve of 3x^3 + 2x^2 + x + 1 from 0 to 2")
```

Result

[1] 21.33333

Output:



2. Exercise 2: Integration of Exponential Function

o Task: Integrate the function f(x)=exf(x) = e^xf(x)=ex from 1 to 3.

o Compare: Compare the result with the analytical solution.

o Expected Output:

```
# Define the function
f <- function(x) { exp(x) }
# Perform integration
result <- integrate(f, lower = 1, upper = 3)
print(result$value)
# Analytical solution
analytical_result <- exp(3) - exp(1)
print(analytical_result)
# Comparison
cat("Numerical result:", result$value, "\n")
cat("Analytical result:", analytical_result, "\n")
```

output:

```
> f<-function(x) {exp(x)}
> result <- integrate(f,lower=1,upper=3)
> print(result$value)
[1] 17.36726
> analytical_result<-exp(3)-exp(1)
> print(analytical_result)
[1] 17.36726
> cat("Numerical result:",result$value,"\n")
Numerical result: 17.36726
> cat("Analytical result:",analytical_result,"\n")
Analytical result: 17.36726
```

3. Exercise 3: Double Integration

o Task: Perform double integration of the function f(x,y)=xy over the region [0, 1]

Expected Output:

```
# Define the function
f <- function(x, y) { x * y }
# Perform double integration
```

```
double_integral <- function(x) {
sapply(x, function(x) integrate(f, lower = 0, upper = 1, y = x)$value)
}
result <- integrate(double_integral, lower = 0, upper = 1)
print(result$value)
```

output:

```
f<-function(x,y){x*y}
> double_intergal<- function(x)
+ {
+   sapply(x,function(x) integrate(f,lower=0,upper=1,y=x)$value)
+ }
> result<-integrate(double_intergal,lower=0,upper=1)
> print(result$value)
[1] 0.25
```

4. Exercise 4: Integration with Real Data

o Task: Load a dataset containing time and velocity of a moving object. Compute the distance traveled by integrating the velocity over time. Visualize the velocity and the distance traveled.

o Expected Output:

```
# Example dataset
time <- seq(0, 10, by = 0.1)
velocity <- 3 * time^2 - 2 * time + 1 # Example velocity function
# Create a data frame
data <- data.frame(time = time, velocity = velocity)
# Compute distance traveled using integration
distance_function <- function(t) { 3 * t^2 - 2 * t + 1 }
result <- integrate(distance_function, lower = 0, upper = 10)
print(result$value)
# Cumulative distance
```

```
cumulative_distance <- cumsum(velocity) * 0.1

# Visualization

data$cumulative_distance <- cumulative_distance

ggplot(data, aes(x = time)) +

geom_line(aes(y = velocity), color = "blue") +

geom_line(aes(y = cumulative_distance), color = "red") +

ggtitle("Velocity (blue) and Cumulative Distance (red) over Time")
```

output: