CS504- Programming Languages for Data Analysis

Assignment 2

Student name	Student Id	Student Email Id
Lohitha Yalavarthi	002289255	lyalavarthi20@ubishops.ca
Bhargav Movva	002292699	BMOVVA21@ubishops.ca
Nitish Kumar Pilla	002286814	npilla20@ubishops.ca

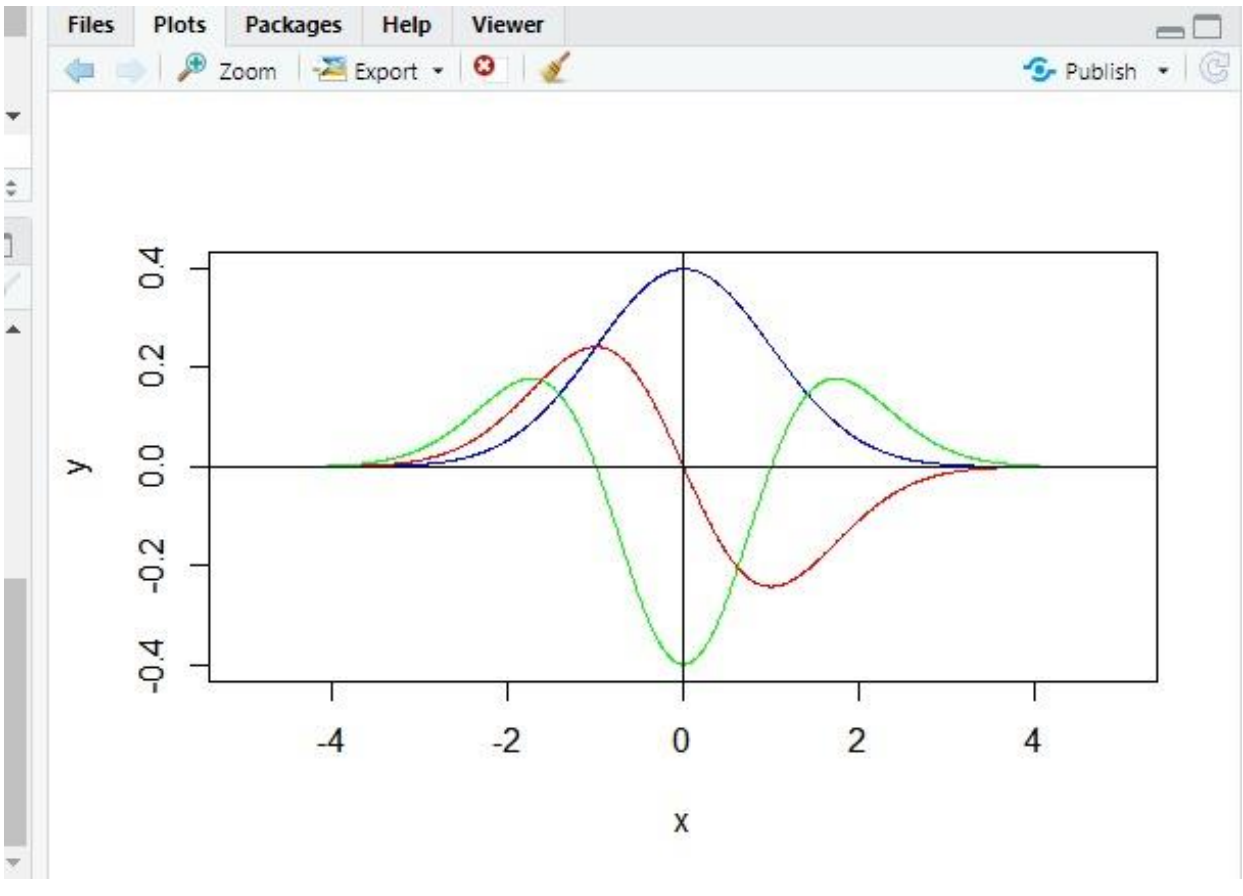
Output results by R programming language:

Code:

```
probability_DensityFn <- function (){
# Choose the mean as 0 and standard deviation as 1.
x <- seq(-5, 5,0.01)
y <- c(-0.4, 0.4)
#y <- 1/sqrt(2*pi)*exp(-x^2/2)

first_derivative <- function(x) {}
second_derivative <- function(x) {}
normalDensity <- function(x)  dnorm(x,0,1)
body(first_derivative) <- D(body(normalDensity), 'x')
body(second_derivative) <- D(body(first_derivative), 'x')
firstderivative= D(dnorm(x),'x')


plot(x,normalDensity(x), type="l",col="blue",ylim=c(-0.4,0.4),xlab="x",ylab="y")
lines(x,first_derivative(x),type="l",col="red")
lines(x,second_derivative(x),type="l",col="green")
abline(v=0, h=0)
}

probability_DensityFn()
```

```
> source('~/R/R_ProbablityDensityFn.r')
>
```

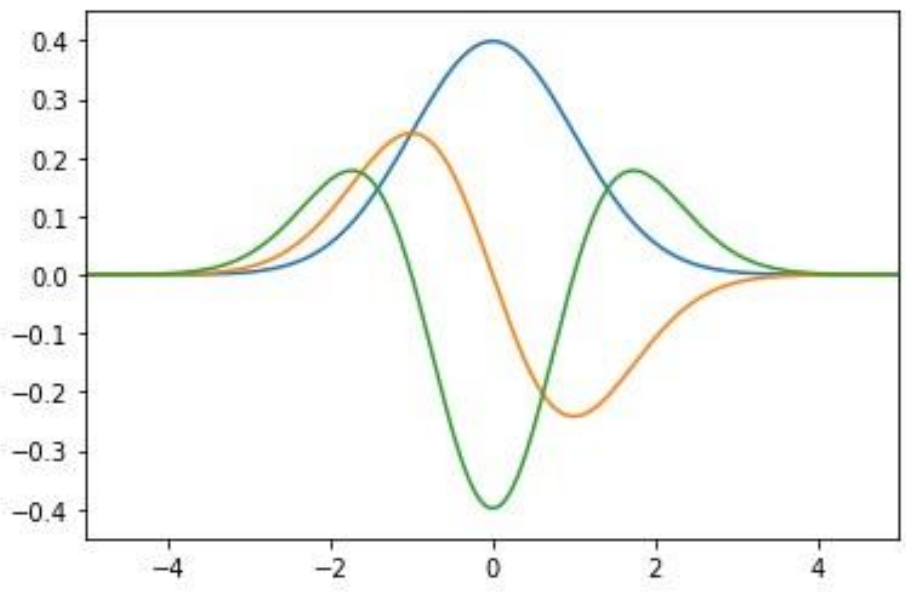
Graph:



Output results by python programming language:

Code and output graph:

In [2]:
```
import sympy as sp
import matplotlib.pyplot as plt
import numpy as np
from scipy.misc import derivative
from scipy.stats import norm

def probability_Densityfn():
    x = sp.Symbol('x')
    xGrid=np.arange(-5,5,0.01)
    def normal(x):
        return 1/np.sqrt(2*np.pi)*np.exp(-x**2/2)
    def first(x):
        return derivative(normal,x,dx=0.01)
    def second(x):
        return derivative(first,x,dx=0.01)

    values =   np.arange(-5,5,0.01)
    density =  [normal(i) for i in values]
    first1   =  [first(i) for i in values]
    second1  =  [second(i) for i in values]

    plt.ylim([-0.45,0.45])
    plt.xlim([-5,5])
    plt.plot(xGrid,density)
    plt.plot(xGrid,first1)
    plt.plot(xGrid,second1)

probability_Densityfn()
```



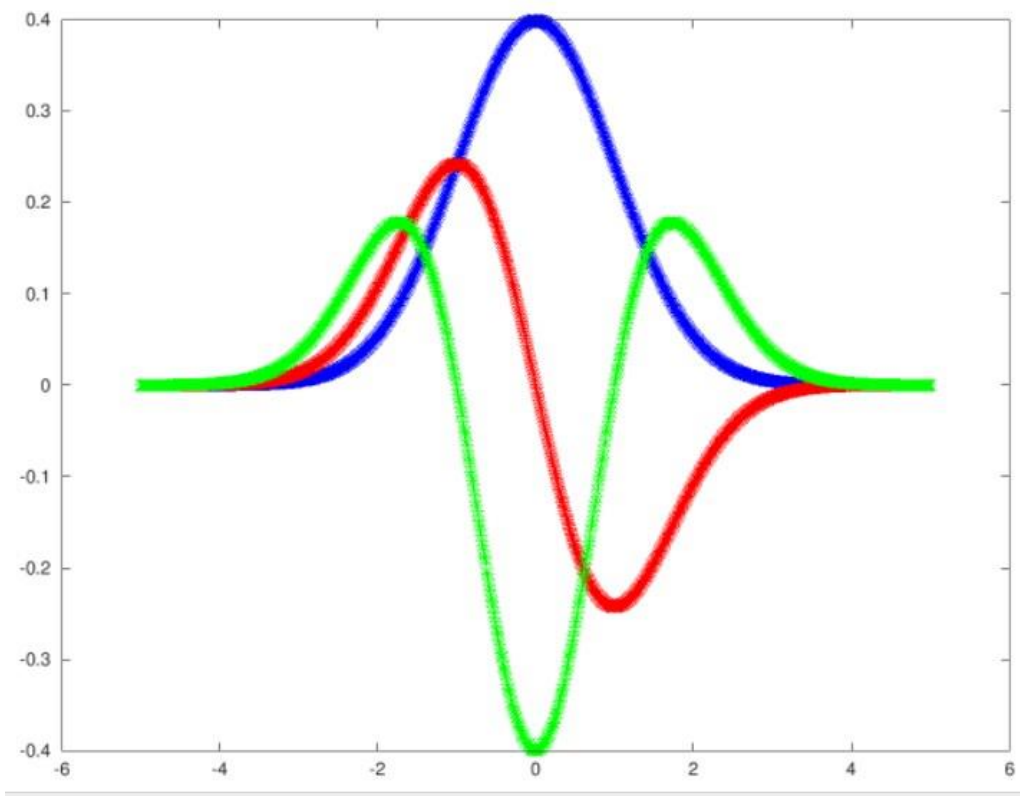
Output results by octave programming language:

Code:

Execute | Share | demo.m | STDIN

```
pkg load statistics
pkg load optim
function demo
    X = -5:0.01:5;
    Y = normpdf(X);
    dY = @(num) normpdf(num);
    d0 = deriv(dY,X);
    d1 = @(X) deriv(dY,X);
    d2 = deriv(d1,X);
    plot(X, Y, "b*" , X, d0, "r*",  X, d2, "g*");
endfunction
demo()
print -dpng figure.png
```

Graph:



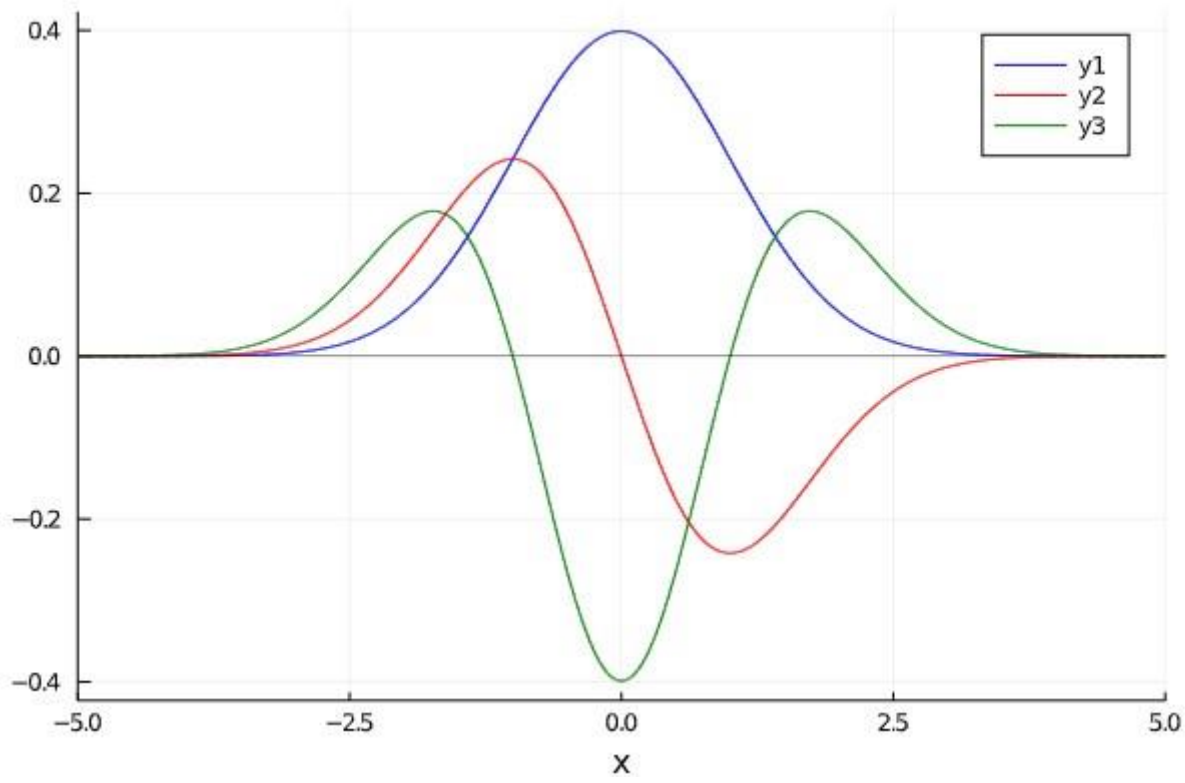
Output results by Julia programming language:

Code:

```
using Distributions, Calculus, SpecialFunctions, Plots;


function julia_pdf()
    xGrid = -5:0.01:5

    PhiA(x) = 0.5*(1+erf(x/sqrt(2)))
    PhiB(x) = cdf(Normal(),x)

    println("Maximum difference between two CDF implementations: ",
    maximum(PhiA.(xGrid) - PhiB.(xGrid)))

    normalDensity(z) = pdf(Normal(),z)

    d0 = normalDensity.(xGrid)
    d1 = derivative.(normalDensity,xGrid)
    d2 = second_derivative.(normalDensity, xGrid)

    plot(xGrid, [d0 d1 d2], c=[:blue :red :green])
    plot!([-5,5],[0,0], color=:black, lw=0.5, xlabel="x", xlims=(-5,5), label="")
end

julia_pdf()
```

Graph:

```
Maximum difference between two CDF implementations: 1.1102230246251565e-16
```



Which programming language that provided relevant solution for this assignment?

Ans: python, R, Julia and octave gave the relevant solution and we felt it easy to implement in octave as it took less lines of code**.**

Which difficulties will you face if you want to solve this problem using Scala programming language?

Ans:

1) Scala lacks behind python, R, Julia, and octave in terms of flexible plotting libraries. for example, R has built-in plots like ggplot2, and python had flexible libraries like matplotlib but for Scala, we have Vegas which is an Html based plotting, which is not flexible to use, and has less functionality when compared to other language plotting libraries.

2) Scala doesn't have statistical inbuilt functions to perform tasks like probability distribution function, derivation of the function...etc.

3) Scala has complex syntax structure when compared to R, python, Julia, octave because Scala is an object-oriented, statically typed programming language, we must specify object types and variables which is an extra process when compared to dynamic programming languages like python and R.