```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from scipy.interpolate import make_interp_spline

# Function definitions remain the same

def myFactorial(n):
    factorialValue = 1
    for i in range(1, n+1):
        factorialValue = i * factorialValue
    return factorialValue

def binomialCoef(f, b):
    return myFactorial(f+b) / (myFactorial(f) * myFactorial(b))

def likelihood(m, f, b):
    bfResult = binomialCoef(f, b)
    for x in range(len(likelihoodArr)):
        likelihoodArr[x] = bfResult * (m[x] ** f) * ((1 - m[x]) ** b)
    return likelihoodArr

def posterior(p, pT):
    for x in range(len(p)):
        posteriorProb[x] = p[x] / pT
    return posteriorProb

# Model and prior initialization
model = np.arange(0.0, 1.1, 0.1)  # Model values: probability of success in each jump

mound_shaped = np.array([0.05, 0.10, 0.16, 0.19, 0.18, 0.15, 0.09, 0.05, 0.2, 0.01, 0.00]) #mound-shaped
bimodal = np.array([0.8, 0.7, 0.3, 0.05, 0.05, 0.05, 0.05, 0.05, 0.3, 0.5, 0.8]) #bimodal
right_skewed = np.array([0.8, 0.6, 0.4, 0.3, 0.2, 0.1, 0.05, 0.01, 0.001, 0.001, 0.0]) #right-skewed
uniform = np.full([11], 0.5) #uniform prior

priorProb = [mound_shaped,
             bimodal,
             right_skewed,
             uniform]
linespecs = ['--', ':', '-']

```

```
41
42  # Initialize arrays for likelihood and posterior
43  likelihoodArr = np.zeros((len(model)))
44  posteriorProb = np.zeros((len(model)))
45
46  #senarious for 3/5, 15/25, and 75/125
47  floor = [3, 15, 75]
48  back = [2, 10, 50]
49
50  figure, axis = plt.subplots(2, 2)
51
52  # Mound Shaped
53  for i in range(3):
54      # Cubic interpolation for prior
55      xArr = np.array([x for x in range(len(model))])
56      X_Y_Spline = interp1d(xArr, priorProb[0], kind='cubic')
57      priorProbX = np.linspace(xArr.min(), xArr.max(), 1000)
58      priorProbY = X_Y_Spline(priorProbX)
59
60      axis[0, 0].plot(priorProbX, priorProbY, color = 'r')
61
62      likelihoodArr = likelihood(model, floor[i], back[i])
63      probTemp = priorProb[0] * likelihoodArr
64      probTempSum = sum(probTemp)
65      posteriorProb = posterior(probTemp, probTempSum)
66
67      # Spline interpolation for posterior
68      X_Y_Spline2 = make_interp_spline(xArr, posteriorProb)
69      posteriorProbY = X_Y_Spline2(priorProbX)
70      axis[0, 0].plot(priorProbX, posteriorProbY, color= 'b', ls = linespecs[i])
71
72
73  axis[0, 0].set_title("Mound Shaped prior distribution")
74
75  # Bimodal
76  for i in range(3):
77      # Cubic interpolation for prior
78      xArr = np.array([x for x in range(len(model))])
79      X_Y_Spline = interp1d(xArr, priorProb[1], kind='cubic')
80      priorProbX = np.linspace(xArr.min(), xArr.max(), 1000)
81      priorProbY = X_Y_Spline(priorProbX)
```

```
        axis[0, 1].plot(priorProbX, priorProbY, color = 'r')

        likelihoodArr = likelihood(model, floor[i], back[i])
        probTemp = priorProb[1] * likelihoodArr
        probTempSum = sum(probTemp)
        posteriorProb = posterior(probTemp, probTempSum)

        # Spline interpolation for posterior
        X_Y_Spline2 = make_interp_spline(xArr, posteriorProb)
        posteriorProbY = X_Y_Spline2(priorProbX)
        axis[0, 1].plot(priorProbX, posteriorProbY, color= 'b', ls = linespecs[i])

    axis[0, 1].set_title("Bimodal prior distribution")

    # Right-skewed
    for i in range(3):
        # Cubic interpolation for prior
        xArr = np.array([x for x in range(len(model))])
        X_Y_Spline = interp1d(xArr, priorProb[2], kind='cubic')
        priorProbX = np.linspace(xArr.min(), xArr.max(), 1000)
        priorProbY = X_Y_Spline(priorProbX)

        axis[1, 0].plot(priorProbX, priorProbY, color = 'r')

        likelihoodArr = likelihood(model, floor[i], back[i])
        probTemp = priorProb[2] * likelihoodArr
        probTempSum = sum(probTemp)
        posteriorProb = posterior(probTemp, probTempSum)

        # Spline interpolation for posterior
        X_Y_Spline2 = make_interp_spline(xArr, posteriorProb)
        posteriorProbY = X_Y_Spline2(priorProbX)
        axis[1, 0].plot(priorProbX, posteriorProbY, color= 'b', ls = linespecs[i])

    axis[1, 0].set_title("Right-skewed prior distribution")
```

```
    # Uniform
    for i in range(3):
        # Cubic interpolation for prior
        xArr = np.array([x for x in range(len(model))])
        X_Y_Spline = interp1d(xArr, priorProb[3], kind='cubic')
        priorProbX = np.linspace(xArr.min(), xArr.max(), 1000)
        priorProbY = X_Y_Spline(priorProbX)

        # to avoid excess labels in legend
        if i == 0:
            axis[1, 1].plot(priorProbX, priorProbY, color = 'r', label='Prior')
        else:
            axis[1, 1].plot(priorProbX, priorProbY, color = 'r')
        # Likelihood and posterior
        likelihoodArr = likelihood(model, floor[i], back[i])
        probTemp = priorProb[3] * likelihoodArr
        probTempSum = sum(probTemp)
        posteriorProb = posterior(probTemp, probTempSum)

        # Spline interpolation for posterior
        X_Y_Spline2 = make_interp_spline(xArr, posteriorProb)
        posteriorProbY = X_Y_Spline2(priorProbX)
        axis[1, 1].plot(priorProbX, posteriorProbY, color= 'b', ls = linespecs[i], label=f'Posterior, {floor[i]}/{floor[i]+back[i]} data')

    axis[1, 1].set_title("Uniform prior distribution")

    plt.legend()
    plt.show()
```

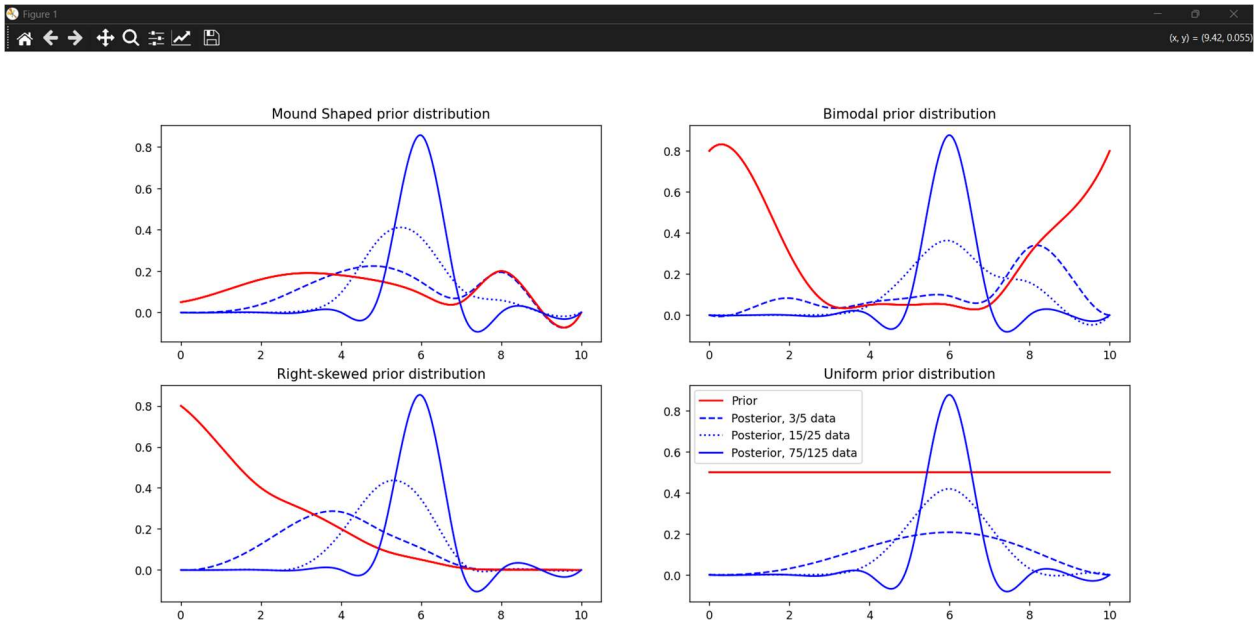
Figure 1
(x, y) = (9.42, 0.055)
Mound Shaped prior distribution
Bimodal prior distribution
Right-skewed prior distribution
Uniform prior distribution
Prior
Posterior, 3/5 data
Posterior, 15/25 data
Posterior, 75/125 data