In [ ]:

```
import numpy as np
import math
from scipy.stats import norm
from scipy.stats import beta
from scipy.stats import fisher_exact
import scipy.integrate as integrate
import matplotlib as mpl
import matplotlib.pyplot as plt
from sympy import symbols, solve



mpl.rcParams['pdf.fonttype'] = 42
mpl.rcParams['ps.fonttype'] = 42
fig_dpi       = 300
fig_typeface = 'Helvetica'
fig_family   = 'monospace'
fig_style    = 'normal'
```

Qustion 1

In [ ]:

```
# Haldane Prior:
pHaldane = 1 - norm.cdf(0, loc=2.20, scale= 15/14)
# Flat Prior:
pFlat = 1 - norm.cdf(0, loc=np.log(29/5), scale= 4/5)
# noninformative Prior:
pNoninformative = 1 - norm.cdf(0, loc=np.log(7), scale= 2*(1/29+2/9+1/5))
print("The probability corresponding to Haldane prior is : %.3f"%pHaldane)
print("The probability corresponding to Flat prior is : %.3f"%pFlat)
print("The probability corresponding to Noninformative prior is : %.3f"%pNoninformative)
```

In [ ]:

```
dicParam = {"Haldane":[0, 0, 0, 0],
            "Flat": [1, 1, 1, 1],
            "Noninformative": [1/2, 1/2, 1/2, 1/2]}
sampleSize = [int(1e4), int(1e5), int(1e6)]
for key in dicParam.keys():
    print(key, " prior deviates: ")
    dic = dicParam[key]
    a = dic[0]

    for _ in sampleSize:
        print(_, ": Pr = ", end = " ")
        pi_ = beta.rvs(a + 14, a + 4, size = _)
        phi_ = beta.rvs(a + 2, a + 4, size = _)
        simulations = np.log(pi_/(1-pi_)) - np.log(phi_/(1-phi_))
        print(len(simulations[simulations>0]) / _)
```

In [ ]:

```
table = np.array([[14, 4], [2, 4]])
odd_ratio, p_value = fisher_exact(table, alternative="greater")
odd_ratio, p_value
```

In [ ]:

```

```

Question 4

In [ ]:

```
def likelihood(theta, n):
    n1, n2, n3, n4 = n
    return (2+theta)**n1 * (1-theta)**n2 * (1-theta)**n3 * (theta)**n4
```

In [ ]:

```
n = (125, 18, 20, 34)
theta = np.linspace(0, 1, 1000000)
L = likelihood(theta, n)
I = integrate.quad(likelihood, 0, 1, args=[n[i] for i in range(4)])
L_norm = L/(I[0] - I[1])

theta_mle = theta[np.argmax(L)]
print('Maximum Likelihood Estimate:', theta_mle)
n_sum = np.sum(n)
# I = np.array([
#     [-np.sum(n_sum * (2+theta_mle)**2) / (2+theta_mle)**2, 0],
#     [0, -np.sum(n_sum * (1-theta_mle)**2) / (1-theta_mle)**2]
# ])
# var_theta_mle = np.abs(np.linalg.inv(I).sum())
var_theta_mle = 1/(-(-n[0]/(2+theta_mle)**2 - n[1]/(1-theta_mle)**2 - n[2]/(1-theta_mle)**2 - n[3]/theta_r
print('Variance of MLE:', var_theta_mle)

mu = theta_mle
sigma = np.sqrt(var_theta_mle)
print('Normal Approximation: N(%.4f, %.4f)' % (mu, sigma))
```

In [ ]:

```
f, ax = plt.subplots(1, 1, figsize=(3, 3), facecolor='white', dpi=300, gridspec_kw={'hspace': 0., 'wspace
ax.plot(theta, L_norm, color = "blue",ls = '-', lw=1, label='Normalized Likelihood function', zorder = 1)
plt.plot(theta, norm.pdf(theta, mu, sigma),ls = "--", lw = 1.1,color = "red",  label='Normal Approximatio
# ax.plot(x, p_beta1, color = "tab:blue",ls = '--', lw=1, label='Beta(10.2, 23.8) pdf', zorder = 1)
# ax.plot(x, p_beta2, color = "tab:red",ls = '--', lw=1, label='Uniform(20.4, 47.6) pdf', zorder = 1)

ax.tick_params(axis='both', which='both', labelsize='xx-small', right=True, top=True, direction='in', wid
ax.set_xlabel(r"$\theta$", size='x-small')
ax.set_ylabel("Density", size='x-small')
ax.set_xlim([0, 1])
ax.legend(loc = 2 ,fontsize = 4,markerscale = 3,ncol = 1,scatterpoints= 1,frameon = True,framealpha =0.).
plt.show()
# f.savefig("./HW5_4a.jpg", bbox_inches = "tight")
```

In [ ]:

```
n = (14, 0, 1, 5)
theta = np.linspace(0, 1, 1000000)
L = likelihood(theta, n)
I = integrate.quad(likelihood, 0, 1, args=[n[i] for i in range(4)])
L_norm = L/(I[0] - I[1])

theta_mle = theta[np.argmax(L)]
print('Maximum Likelihood Estimate:', theta_mle)
n_sum = np.sum(n)
# I = np.array([
#     [-np.sum(n_sum * (2+theta_mle)**2) / (2+theta_mle)**2, 0],
#     [0, -np.sum(n_sum * (1-theta_mle)**2) / (1-theta_mle)**2]
# ])
# var_theta_mle = np.abs(np.linalg.inv(I).sum())
var_theta_mle = 1/(-(-n[0]/(2+theta_mle)**2 - n[1]/(1-theta_mle)**2 - n[2]/(1-theta_mle)**2 - n[3]/theta_r
print('Variance of MLE:', var_theta_mle)

mu = theta_mle
sigma = np.sqrt(var_theta_mle)
print('Normal Approximation: N(%.4f, %.4f)' % (mu, sigma))
```

In [ ]:

```
f, ax = plt.subplots(1, 1, figsize=(3, 3), facecolor='white', dpi=300, gridspec_kw={'hspace': 0., 'wspace
ax.plot(theta, L_norm, color = "blue",ls = '-', lw=1, label='Normalized Likelihood function', zorder = 1)
plt.plot(theta, norm.pdf(theta, mu, sigma),ls = "--", lw = 1.1,color = "red",  label='Normal Approximatio
# ax.plot(x, p_beta1, color = "tab:blue",ls = '--', lw=1, label='Beta(10.2, 23.8) pdf', zorder = 1)
# ax.plot(x, p_beta2, color = "tab:red",ls = '--', lw=1, label='Uniform(20.4, 47.6) pdf', zorder = 1)

ax.tick_params(axis='both', which='both', labelsize='xx-small', right=True, top=True, direction='in', wid
ax.set_xlabel(r"$\theta$", size='x-small')
ax.set_ylabel("Density", size='x-small')
ax.set_xlim([0, 1])
ax.legend(loc = 2 ,fontsize = 4,markerscale = 3,ncol = 1,scatterpoints= 1,frameon = True,framealpha =0.).
plt.show()
# f.savefig("./HW5_4b.jpg", bbox_inches = "tight")
```

Question 5

In [ ]:

```
def h(theta, n):
    h = -1/(np.sum(n))*np.log(likelihood(theta, n))
    return h
def h_star(theta, n):
    h_star = h(theta, n) - np.log(theta)/np.sum(n)
    return h_star

def sigmaL(theta, n):
    n0, n1, n2, n3 = n
    sigmaL = 1/np.sqrt(1/(np.sum(n)* (n0/(2+theta)**2 + n1/(1-theta)**2 + n2/(1-theta)**2 + n3/theta**2)
    return sigmaL
```

In [ ]:

```
n1 = (125, 18, 20, 34)
n2 = (14, 0, 1, 5)


theta_hat1 = 0.6268215
theta_star1 = 0.63099204
theta_hat2 = 0.90344011
theta_star2 = 0.90295653

E_theta1 = sigmaL(theta_star1, n1)/sigmaL(theta_hat1, n1) * np.exp(-197*h_star(theta_star1, n1))/np.exp(-
E_theta1
```

In [ ]:

```
E_theta2 = sigmaL(theta_star2, n2)/sigmaL(theta_hat2, n2) * np.exp(-20*h_star(theta_star2, n2))/np.exp(-2
E_theta2
```

In [ ]: