JUPYTER NOTEBOOK CODE FOR "A FAIR SHAKE: HOW CLOSE CAN THE SUM OF n-SIDED DICE BE TO A UNIFORM DISTRIBUTION?"

APPENDIX A. PYTHON CODE FOR MINIMIZING D WITH THREE DICE

[1]:
```
%pylab inline
import pandas as pd
```

```
Populating the interactive namespace from numpy and matplotlib
```

[2]:
```
# Main inputs

n = 5 # number of sides on each of the three dice

p_0 = zeros(n) # initialization for the probabilities on each die
q_0 = zeros(n)
r_0 = zeros(n)
p_0[0]=.99; p_0[1]=.01 # initial guess (makes the probabilities now sum to one␣
 ↪for each die)
q_0[0]=.98; q_0[1]=.02
r_0[0]=.97; r_0[1]=.03

probs_0 = concatenate((p_0,q_0,r_0),axis=None)
```

[3]:
```
# Determine D, the sum of the squares to be minimized

def f(probs): # probs are the concatenated probabilities all three dice
    p = probs[0:n]
    q = probs[n:2*n]
    r = probs[2*n:3*n]
    c = zeros(3*n-2)  # c contains the probabilities of each of the dice sums␣
 ↪3,4,...,3n.

    for i in range(n):  # Determine c
        for j in range(n):
            for k in range(n):
                c[i+j+k] += p[i]*q[j]*r[k]

    D = 0 # Compute, D, the objective function
    for i in range(len(c)):
        D += (c[i] - 1/(3*n-2))**2
    return D
```

[4]:
```
# Create the matrix M, which will be used for the equality constraints later

one = ones(n)
zer = zeros(n)

row1 = concatenate((one,zer,zer),axis=None)
row2 = concatenate((zer,one,zer),axis=None)
row3 = concatenate((zer,zer,one),axis=None)

M = [row1,row2,row3]
```

[5]:
```
# Upload needed packages and compute the inequality and equality constraints

from scipy.optimize import minimize

from scipy.optimize import Bounds
bounds = Bounds(zeros(3*n), ones(3*n)) # The inequality constraints keeping p,␣
↪q, and r between 0 and 1

from scipy.optimize import LinearConstraint
linear_constraint = LinearConstraint(M, [1,1,1], [1,1,1]) # Equality constraints␣
↪so the sum of each die's
                                                          # probabilities equals␣
↪one.
```

[6]:
```
# Minimize D and print results

res = minimize(f, probs_0, method='trust-constr', constraints=linear_constraint,␣
↪bounds=bounds, \
               options={'gtol': 3e-11})

print("D is minimized at p =", res.x[0:n])
print("D is minimized at q =", res.x[n:2*n])
print("D is minimized at r =", res.x[2*n:3*n])
print("The minimized value of D is =", res.fun)
```

```
D is minimized at p = [4.99999945e-01 4.37695235e-08 3.73453721e-08
3.32074217e-09
 4.99999970e-01]
D is minimized at q = [4.99999963e-01 3.24953912e-08 1.18015791e-08
2.99815066e-08
 4.99999962e-01]
D is minimized at r = [0.14285715 0.23809524 0.23809524 0.23809524 0.14285715]
The minimized value of D is = 0.01236263871224755
```