**Functions**

1. **Statistics intermediate in Python**
2. Max value in column and its row

most\_bars\_country = flags["name"][flags["bars"].idxmax()]

highest\_population\_country = flags["name"][flags["population"].idxmax()]

1. Probability

total\_countries = flags.shape[0]

country\_col\_orange = flags["name"][flags["orange"] == 1]

orange\_probability = len(country\_col\_orange)/total\_countries

1. Dependent probabilities

def pick(colr, numrow):

r = flags[colr][flags[colr] == 1]

p = len(flags["colors"])

red = 1

for i in range(0,numrow):

prob = (len(r) - i)/(p- i)

red = red \* prob

return red

three\_red = pick("red", 3)

1. Dependent OR possibilities

stripes\_or\_bars = None

red\_or\_orange = None

red = flags[flags["red"] == 1].shape[0] / flags.shape[0]

orange = flags[flags["orange"] == 1].shape[0] / flags.shape[0]

red\_and\_orange = flags[(flags["red"] == 1) & (flags["orange"] == 1)].shape[0] / flags.shape[0]

red\_or\_orange = red + orange - red\_and\_orange

1. Factorial function

import math

def find\_outcome\_combinations(N, k):

# Calculate the numerator of our formula.

numerator = math.factorial(N)

# Calculate the denominator.

denominator = math.factorial(k) \* math.factorial(N - k)

# Divide them to get the final value.

return numerator / denominator

1. Function To Calculate The Probability Of A Single Combination

p = .6

q = .4

def find\_combination\_probability(N, k, p, q):

# Take p to the power k, and get the first term.

term\_1 = p \*\* k

# Take q to the power N-k, and get the second term.

term\_2 = q \*\* (N-k)

# Multiply the terms out.

return term\_1 \* term\_2

prob\_8 = find\_outcome\_combinations(10, 8) \* find\_combination\_probability(10, 8, p, q)

1. Binomial distribution (pmf) function

from scipy import linspace

from scipy.stats import binom

# Create a range of numbers from 0 to 30, with 31 elements (each number has one entry).

outcome\_counts = linspace(0,30,31)

# Create the binomial probabilities, one for each entry in outcome\_counts.

dist = binom.pmf(outcome\_counts,30,0.39)

# Create the binomial probabilities, one for each entry in outcome\_counts.

dist = binom.pmf(outcome\_counts,30,0.39)

plt.bar(outcome\_counts, dist)

#plt.xlabel("Number of days with more than 5000 riders")

#plt.ylabel("Probability score")

plt.show()

1. Binomial distribution (cdf) function

from scipy import linspace

from scipy.stats import binom

# Create a range of numbers from 0 to 30, with 31 elements (each number has one entry).

outcome\_counts = linspace(0,30,31)

# Create the cumulative binomial probabilities, one for each entry in outcome\_counts.

dist = binom.cdf(outcome\_counts,30,0.39)

left\_16 = None

right\_16 = None

# The sum of all the probabilities to the left of k, including k.

left\_16 = binom.cdf(16,30,0.39)

# The sum of all probabilities to the right of k.

right\_16 = 1 - left\_16

1. Dictionary with get() function

sampling\_distribution = {}

for r in mean\_differences:

if sampling\_distribution.get(r, False):

# If in the dictionary, grab the value, increment by 1, reassign.

val = sampling\_distribution.get(r)

inc = val + 1

sampling\_distribution[r] = inc

else:

# If not in the dictionary, assign `1` as the value to that key.

sampling\_distribution[r] = 1

1. Counting the frequencies

frequencies = []

for i in sampling\_distribution.keys():

if i >= 2.52:

frequencies.append(sampling\_distribution[i])

sum\_freq = np.sum(frequencies)

p\_value = sum\_freq/1000

1. Chi-squared distribution

chi\_squared\_values = []

from numpy.random import random

import matplotlib.pyplot as plt

for i in range(1000):

sequence = random((32561,))

sequence[sequence < .5] = 0

sequence[sequence >= .5] = 1

a = len(sequence[sequence == 0])

b = len(sequence[sequence == 1])

female\_diff = (a - 16280.5) \*\* 2 / 16280.5

male\_diff = (b - 16280.5) \*\* 2 / 16280.5

gender\_chisq = female\_diff + male\_diff

chi\_squared\_values.append(gender\_chisq)

plt.hist(chi\_squared\_values)

plt.show()

1. Chi-squared distribution function in scipy

from scipy.stats import chisquare

import numpy as np

observed = [27816, 3124, 1039, 311, 271]

expected = [26146.5, 3939.9, 944.3, 260.5, 1269.8]

chisquare\_value, pvalue = chisquare(observed, expected)

race\_pvalue = pvalue