# Distributions in Pandas

import pandas as pd

import numpy as np

np.random.binomial(1, 0.5) # 50% we get 0, run 1 time

0

np.random.binomial(1000, 0.5)/1000

0.519

Example1:

Suppose we want to simulate the probability of flipping a fair coin 20 times, and getting a number greater than or equal to 15. Use np.random.binomial(n, p, size) to do 10000 simulations of flipping a fair coin 20 times, then see what proportion of the simulations are 15 or greater:

x = np.random.binomial(20, .5, 10000)

array([11, 8, 6, ..., 11, 8, 11])

print((x>=15).mean())

0.0212

chance\_of\_tornado = 0.01/100

np.random.binomial(100000, chance\_of\_tornado)

13

chance\_of\_tornado = 0.01

tornado\_events = np.random.binomial(1, chance\_of\_tornado, 1000000)

two\_days\_in\_a\_row = 0

for j in range(1,len(tornado\_events)-1):

if tornado\_events[j]==1 and tornado\_events[j-1]==1:

two\_days\_in\_a\_row+=1

print('{} tornadoes back to back in {} years'.format(two\_days\_in\_a\_row, 1000000/365))

73 tornadoes back to back in 2739.72602739726 years

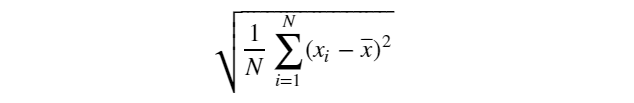
np.random.uniform(0, 1) # randomly pick a sample in uniform distribution

0.49222970126544374

np.random.normal(0.75) # randomly pick a sample in normal distribution

0.7158682537945757

Formula for standard deviation



distribution = np.random.normal(0.75,size=1000)

np.sqrt(np.sum((np.mean(distribution)-distribution)\*\*2)/len(distribution)) # std

1.0347358861974973

np.std(distribution)

1.0347358861974973

import scipy.stats as stats

stats.kurtosis(distribution) # >0 peaky distribution

0.0815385508531894

stats.skew(distribution)

0.08978336853945652

chi\_squared\_df2 = np.random.chisquare(2, size=10000) # df = 2, skew is large

stats.skew(chi\_squared\_df2)

1.9381745650011923

chi\_squared\_df5 = np.random.chisquare(5, size=10000) # df = 5, skew is decreased

stats.skew(chi\_squared\_df5)

1.2885968583617342

%matplotlib inline

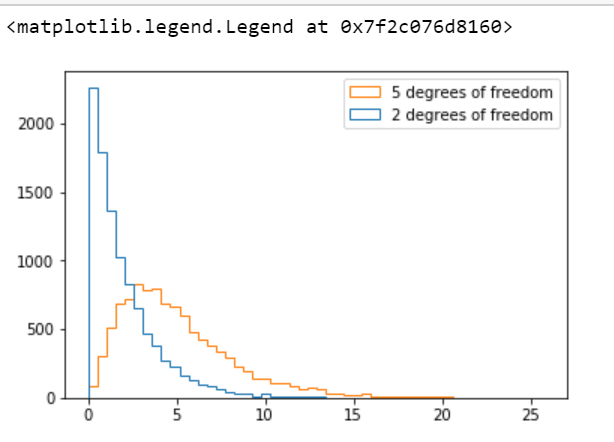
import matplotlib

import matplotlib.pyplot as plt

output = plt.hist([chi\_squared\_df2,chi\_squared\_df5], bins=50, histtype='step',

label=['2 degrees of freedom','5 degrees of freedom'])

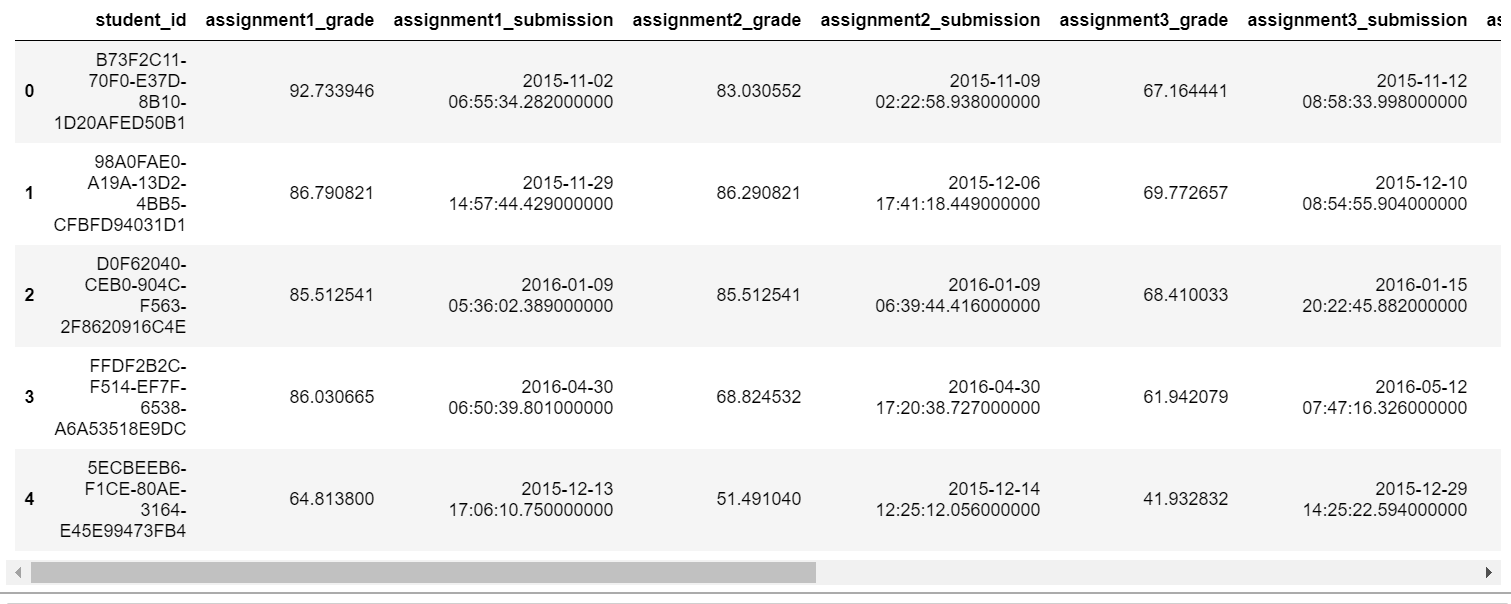
plt.legend(loc='upper right')



# Hypothesis Testing

df = pd.read\_csv('grades.csv')

df.head()



len(df)

2315

early = df[df['assignment1\_submission'] <= '2015-12-31']

late = df[df['assignment1\_submission'] > '2015-12-31']

early.mean()

assignment1\_grade 74.972741

assignment2\_grade 67.252190

assignment3\_grade 61.129050

assignment4\_grade 54.157620

assignment5\_grade 48.634643

assignment6\_grade 43.838980

dtype: float64

late.mean()

assignment1\_grade 74.017429

assignment2\_grade 66.370822

assignment3\_grade 60.023244

assignment4\_grade 54.058138

assignment5\_grade 48.599402

assignment6\_grade 43.844384

dtype: float64

# compare the means of two different populations

from scipy import stats

stats.ttest\_ind?

stats.ttest\_ind(early['assignment1\_grade'], late['assignment1\_grade'])

# > 0.05 cannot reject the null hypothesis, 2 popu are the same

Ttest\_indResult(statistic=1.400549944897566, pvalue=0.16148283016060577)

stats.ttest\_ind(early['assignment2\_grade'], late['assignment2\_grade'])

# > 0.05 cannot reject the null hypothesis, 2 popu are the same

Ttest\_indResult(statistic=1.3239868220912567, pvalue=0.18563824610067967)

stats.ttest\_ind(early['assignment3\_grade'], late['assignment3\_grade'])

# > 0.05 cannot reject the null hypothesis, 2 popu are the same

Ttest\_indResult(statistic=1.7116160037010733, pvalue=0.087101516341556676)