"""

Created on Thu Oct 13 16:10:01 2022

@author: JosephNavelski

"""

# Loading Packages

import random

import numpy as np

import pandas as pd

import matplotlib

import matplotlib.pyplot as plt

import scipy.stats as stats

import statsmodels.stats.power as smp

# Setting some printing options to make it easier to visualize

pd.set\_option('max\_columns', 10)

pd.set\_option('max\_rows', 20)

pd.set\_option('expand\_frame\_repr', False)

# Define the seed so that results can be reproduced

seed = 123

# Define the color maps for plots

color\_map = plt.cm.get\_cmap('RdYlBu')

color\_map\_discrete = matplotlib.colors.LinearSegmentedColormap.from\_list("", ["red","cyan","magenta","blue"])

random.seed(seed)

print(random.random())

rand = np.random.RandomState(seed)

# Example List of Distributions

dist\_list = ['normal','normal','normal','normal','normal','normal']

param\_list = ['0,1','.1,1','.2,1','.3,1','.5,1','.75,1']

colors\_list = ['green','blue','yellow','cyan','magenta','pink']

fig,ax = plt.subplots(nrows=2, ncols=3,figsize=(12,7))

plt\_ind\_list = np.arange(6)+231

# Run all at once

for dist, plt\_ind, param, colors in zip(dist\_list, plt\_ind\_list, param\_list, colors\_list):

x = eval('rand.'+dist+'('+param+',5000)')

plt.subplot(plt\_ind)

plt.hist(x,bins=50,color=colors)

plt.title(dist)

fig.subplots\_adjust(hspace=0.4,wspace=.3)

plt.suptitle('Sampling from Various Distributions',fontsize=20)

plt.show()

########################################################################

# A/B Testing Exercise Start

########################################################################

# Set Some Parameters

n = 8000

location = [0,.1,.2,.3,.5,.75]

spread = [1,1,1,1,1,1]

# A quick Power Analalysis

power\_analysis = smp.TTestIndPower()

sample\_size = power\_analysis.solve\_power(effect\_size=location[1], power=0.8, alpha=0.05)

sample\_size

effect\_sizes = np.array(location[1:6])

sample\_sizes = np.array(range(10, 1600))

plt.style.use('seaborn')

fig = plt.figure()

ax = fig.add\_subplot(1, 1, 1)

fig = power\_analysis.plot\_power(

dep\_var='nobs', nobs=sample\_sizes,

effect\_size=effect\_sizes, alpha=0.05, ax=ax,

title='Power of Independent Samples t-test\n$\\alpha = 0.05$')

plt.show(block=False)

# Generating data from a normal(mu,sigma)

dist\_x = pd.DataFrame(index=range(n))

for i,j in zip(location, spread):

print('norm\_('+str(i)+'\_'+str(j)+')')

x = pd.DataFrame(data=np.random.normal(i,j,n),columns= ['norm\_('+str(i)+','+str(j)+')'])

dist\_x = dist\_x.join(x)

# Looking at the data

bins = np.linspace(-4, 4, 50)

for i in range(1,len(location)):

plt.hist([dist\_x.iloc[:,0], dist\_x.iloc[:,i]], bins, label=['norm\_(0,1)', dist\_x.columns[i]])

plt.legend(loc='upper right')

plt.show()

# Running T-tests and building a table of p-values

r\_tbl = pd.DataFrame(index=range(15))

dist\_x.columns[1]

for i in range(1,len(location)):

x = pd.DataFrame(index=range(15),columns= [dist\_x.columns[i]])

full = pd.concat([dist\_x.iloc[:,0],dist\_x.iloc[:,i]],axis=0)

x.iloc[0,0] = stats.normaltest(full)[1]

x.iloc[1,0] = stats.normaltest(dist\_x.iloc[:,0])[1]

x.iloc[2,0] = stats.normaltest(dist\_x.iloc[:,i])[1]

# F-test - Exremely Sensitive to non-normality

F = np.var(dist\_x.iloc[:,0]) / np.var(dist\_x.iloc[:,i])

df1 = len(dist\_x.iloc[:,0]) - 1

df2 = len(dist\_x.iloc[:,i]) - 1

x.iloc[3,0] = stats.f.cdf(F, df1, df2) # F-test

x.iloc[4,0] = stats.levene(dist\_x.iloc[:,0], dist\_x.iloc[:,i])[1] # More robust equal variance test

x.iloc[5,0] = stats.bartlett(dist\_x.iloc[:,0], dist\_x.iloc[:,i])[1] # Another equal variance test

x.iloc[6,0] = stats.ttest\_ind(a=dist\_x.iloc[:,0], b=dist\_x.iloc[:,i], equal\_var=True)[1] # Standard Student t-test

x.iloc[7,0] = stats.ttest\_ind(a=dist\_x.iloc[:,0], b=dist\_x.iloc[:,i], equal\_var=False)[1] # Welch

x.iloc[8,0] = stats.wilcoxon(dist\_x.iloc[:,0],dist\_x.iloc[:,i])[1] # Non-parametric

x.iloc[9,0] = np.mean(dist\_x.iloc[:,0])

x.iloc[10,0] = np.mean(dist\_x.iloc[:,i])

x.iloc[11,0] = x.iloc[10,0] - x.iloc[9,0]

x.iloc[12,0] = power\_analysis.solve\_power(effect\_size= abs(x.iloc[11,0]), power=0.8, alpha=0.05)

if len(dist\_x.iloc[:,i]) > x.iloc[12,0]:

x.iloc[13,0] = True

else:

x.iloc[13,0] = False

x.iloc[14,0] = len(dist\_x.iloc[:,i])

r\_tbl = r\_tbl.join(x)

r\_tbl = r\_tbl.astype(float)

r\_tbl = np.round(r\_tbl,decimals=3)

# Adding in Tests as Row Names

r\_tbl.index = ['Norm Test ~ All (Fail to)', 'Norm Test ~ C (Fail to)', 'Norm Test ~ T (Fail to)',

'Equal Var F Test (Fail to)', 'Equal Var Levene Test (Fail to)', 'Equal Var Bartlett Test (Fail to)',

'Student t-test (Equal V) (Reject)', 'Welch t-test (Unequal V) (Reject)', 'Non-parametric Test (Reject)',

'Mean ~ C', 'Mean ~ T', 'Mean(T) - Mean(C)', 'Obs. Needed for .80 Power', 'Did it Pass? (1 = True)', 'n']

print(' ~~~~~~~~~~~~~~~~~~~~~~ A Table of P-values ~~~~~~~~~~~~~~~~~~~~~~')

print('Alpha = 0.05, which implies that if p-value <= .05, then reject the null ')

print(r\_tbl)