E2_Q1

April 10, 2020

[1]:
```python
# importing necessary library
import numpy as np
import pandas as pd
import math

# mathematical constant
from math import e

# distributions
from scipy.stats import norm
```

[2]:
```python
# Question 1
# read in the data
X = [0.04, 0.41, 0.62, 0.72, 0.90,
     0.92, 1.05, 1.08, 1.27, 1.51,
     1.52, 1.57, 1.57, 1.59, 1.85,
     2.19, 2.25, 2.32, 2.35, 2.49,
     2.60, 2.65, 2.66, 2.69, 2.77]
```

[3]:
```python
# convert X to an ordered sample
X = pd.Series(np.sort(X))
```

[4]:
```python
# define some CDF functions to be used in Cramer Von Mises Tests

# continuous uniform CDF
def continuousUniformCDF(x, a, b):
    return (x-a)/(b-a)

# exponential CDF used in poisson process question
def exponentialCDF(x, theta):
    return (1 - e**(-x / theta))

# a wrapper for the imported normal cdf function
def normalCDF(x, mu, sigma):
    return norm.cdf(x, mu, sigma)
```

[5]:
```
# now we can define a function for calculating the Cramer-Von Mises Test␣
↪Statistic
def CVM_Test(X, n, dis, CDFparam=None, CDFparam2=None):

    # initialize a variable to store the sum of squared differences
    sumTotal = 0

    # calculate the sum of squared distances
    for i in range(n):
        ithTerm = (i+0.5)/n # i goes begins at 0, so (i+1) - 0.5 = i+0.5
        if dis == "ContinuousUniform":
            sumTotal += (continuousUniformCDF(X[i], 0, 3) - ithTerm)**2 # no␣
↪parameter
        elif dis == "Exponential":
            sumTotal += (exponentialCDF(X[i], CDFparam) - ithTerm)**2 # one␣
↪parameter
        elif dis == "Normal":
            sumTotal += (normalCDF(X[i], CDFparam, CDFparam2) - ithTerm)**2 ##␣
↪two parameter

    # calculate the CM test statistic
    observedCM = (1 / (12 * n)) + sumTotal

    # if we estimated a parameter for the argument distribution, apply␣
↪stephen's modifications
    if CDFparam != None:
        if dis == "Exponential":
            observedCM = (1 + (0.16/n)) * observedCM
        elif dis == "Normal":
            observedCM = (1 + (0.5/n)) * observedCM

    # return the test statistic
    return observedCM
```

[6]:
```
# A function to determine whether or not we reject the null
def rejectNull(observedCM, critCM, message):
    # print the question
    print(message)

    # print this test
    print(observedCM, ">", critCM, "=", observedCM>critCM)

    # print the results
    if (observedCM>critCM):
        print("Thus we reject the null.")
    else:
        print("Thus we fail to reject the null.")
```

```
    return
```

[7]:
```
# lets run a CVM test to see if X is from a continuous uniform distribution
observedCM = CVM_Test(X, 25, "ContinuousUniform")
rejectNull(observedCM, 0.347, "Question #1a:")
```

```
Question #1a:
0.13785555555555556 > 0.347 = False
Thus we fail to reject the null.
```

[8]:
```
# now lets test to see if X is a poisson process

# For a poisson process there are 3 conditions.
## c1: The difference between observations should be distributed Exp()
## c2: Events are indepenedent of one another
## c3: Observations cannot overlap

# first we need to calculate the differences between observatoins
X2 = []
for i in range (1, len(X), 1):
    X2.append(X[i] - X[i-1])

# now we want to run a CVM test to see if these differences match an␣
↪exponential distribution
# since the exponential distribution needs a parameter, we will use the MLE as␣
↪an estimate
mle = sum(X2) / len(X2) # MLE is sum(Xi)/n for exponential distribution

# now we can use CVM test to see if X2 ~ Exp(mle)
observedCM = CVM_Test(X2, 24, "Exponential", mle)
rejectNull(observedCM, 0.177, "H0: Difference in observations ~ Exponentially␣
↪for some parameter mu:")
print("")
print("Question #1b. Thus, we reject that X is from a poisson process.")
```

```
H0: Difference in observations ~ Exponentially for some parameter mu:
5.637018814662897 > 0.177 = True
Thus we reject the null.

Question #1b. Thus, we reject that X is from a poisson process.
```

[9]:
```
# finally lets test to see if X is from a normal distribution
# Again, we need to estimate the paramters of the normal CDF

## mu_mle = (1/n) * sum(xi) for normal distribution
mu_mle = 0
for x_i in X:
```

```python
    mu_mle += x_i
mu_mle = mu_mle/len(X)

## sigma_mle = sqrt((1/n) * sum((xi - mu_mle)^2)) for normal distribution
sigma_mle = 0
for x_i in X:
    sigma_mle += (x_i - mu_mle)**2
sigma_mle = sigma_mle / len(X)
sigma_mle = sigma_mle **(1/2)


observedCM = CVM_Test(X, 25, "Normal", mu_mle, sigma_mle)
rejectNull(observedCM, 0.104, "Question #1c:")
```

```
Question #1c:
0.08340833935309093 > 0.104 = False
Thus we fail to reject the null.
```