```
# Assessed exercises for week 8 -qq plots

# It is often the case that we wish to decide which distribution is the best fit
# to a single variable. For example, we might want to see whether the residuals
# of a linear regression are approximately normally distributed. QQ-plots are
# one of the best ways to do this. They are often superior to drawing histograms
# as it's easier to assess whether the tails of the distribution fit.

# In this assessed exercise we're going to create some QQ-plots. The steps to create
# a qqplot to compare a chosen probability distribution with a single variable x are:
# 1. Calculate the empirical cdf (ecdf) of x
# 2. Simulate a large number of observations from the chosen probability distribution
# 3. Find the quantiles of the distribution at the probabilities defined by the ecdf
# If the two data sets match, a plot of the quantiles and the original data should
# fall on a straight line. For more detail, see e.g. http://onlinestatbook.com/2/adva

# In this exercises we will use four data sets which come from four unknown probabili
# distributions. One of them comes from a N(0,1) distribution, another a t_5 distribu
# another a Exp(1) distribution, and finally a Chi-squared(1) distribution. The files
# are labelled qq1 to qq4.txt and are all of different lengths. We're going to use
# QQ-plots to find which data set matches to which probability distribution

#

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
import statsmodels.api as sm
import numpy.random as npr

# First you will need to load in the data sets
path = './'
qq1 = pd.read_csv(path+'qq1.txt',header=None)
qq2 = pd.read_csv(path+'qq2.txt',header=None)
qq3 = pd.read_csv(path+'qq3.txt',header=None)
qq4 = pd.read_csv(path+'qq4.txt',header=None)

# Q1 For the first part of the task, we need to create the empirical cumulative distr
# function (ecdf). This is defined as:
# Number of observations z less than or equal to z_i / Number of observations, for ev
# Write a function called which takes a set of observations z and produces the empiri
# If you are unfamiliar with empirical cdfs, you may want to read the following artic
# https://towardsdatascience.com/what-why-and-how-to-read-empirical-cdf-123e2b922480
def exercise1(z):
    n = z.size
    y = np.arange(1, n+1) / n
    return y

# Plot each of the variables qq1, qq2, etc. versus their ecdf, as subplots in a singl
# Save your figure and include it in your submission.
plt.clf()
y = exercise1(qq1)
plt.subplot(2,2,1)
plt.plot(qq1,y,'blue')
ax = plt.gca()
ax.annotate('qq1', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('ecdf', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', t

y = exercise1(qq2)
plt.subplot(2,2,2)
plt.plot(qq2,y,'red')
ax = plt.gca()
ax.annotate('qq2', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('ecdf', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', t
```

```
y = exercise1(qq3)
plt.subplot(2,2,3)
plt.plot(qq3,y,'green')
ax = plt.gca()
ax.annotate('qq3', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('ecdf', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', t

y = exercise1(qq4)
plt.subplot(2,2,4)
plt.plot(qq4,y,'yellow')
ax = plt.gca()
ax.annotate('qq4', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('ecdf', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', t

plt.savefig("./Q1.png")
# Q2 For the next part we need to create the quantiles of a chosen probability distri
# Write a function which takes an ecdf created by your function in Q2
# and simulates 10,000 observations from a normal(0,1) distribution. Then calculate
# the quantiles of these simulations at the probabilities defined by the ecdf
def exercise2(ecdf):
    mu, sigma = 0, 1
    simulation = np.random.normal(mu, sigma, 10000)
    quant = list()
    for i in ecdf:
        temp = np.quantile(simulation,i)
        quant.append(temp)
    return quant

# Create a scatter plot of the theoretical quantiles from your new function (x-axis)
# this for each dataset, creating each plot as a subplot on the same figure. Save you
# submission. If the two distributions match, the points should lie on a straight lin
# the datasets is normally distributed variable?
plt.clf()
quant_x = exercise2(exercise1(qq1))
plt.subplot(2,2,1)
plt.scatter(quant_x, qq1, c = "orange")
ax = plt.gca()
ax.annotate('quantiles', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('qq1', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', te


quant_x = exercise2(exercise1(qq2))
plt.subplot(2,2,2)
plt.scatter(quant_x, qq2, c = "blue")
ax = plt.gca()
ax.annotate('quantiles', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('qq2', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', te

quant_x = exercise2(exercise1(qq3))
plt.subplot(2,2,3)
plt.scatter(quant_x, qq3, c = "red")
ax = plt.gca()
ax.annotate('quantiles', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('qq3', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', te

quant_x = exercise2(exercise1(qq4))
plt.subplot(2,2,4)
plt.scatter(quant_x, qq4, c = "yellow")
ax = plt.gca()
ax.annotate('quantiles', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('qq4', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', te
plt.savefig('./Q2.png')
```

```
# Ans:
# The qq4 is normally distributed variable, because almost points lie on a straight l

# Q3 Create a new function that takes two arguments. The first argument should be you
# argument should be a set of simulations from some probability distribution. It shou
# the theoretical quantiles. This function should return the computed theoretical qua
def exercise3(y,d):
    ecdf_y = exercise1(y)
    quant = list()
    for i in ecdf_y:
        temp = np.quantile(d,i)
        quant.append(temp)
    return quant

# Q4 Run your function for each of the datasets, with
# - d = Series(npr.randn(10000)) (normal distribution)
# - d = Series(npr.exponential(1,10000)) (exponential distribution)
# - d = Series(npr.standard_t(5,10000)) (t_5 distribution)
# - d = Series(npr.chisquare(1,10000)) (chi-squared distribution)
# Plot empirical data versus the theoretical quantiles returned by exercise3 to deter
# data set matches to which probability distribution
# Complete the quiz 'W8 - Assessed exercises Q4' to submit your answer for this quest
plt.clf()
d = Series(npr.exponential(1,10000))
d = Series(npr.randn(10000))
d = Series(npr.chisquare(1,10000))
d = Series(npr.standard_t(5,10000))

quant_x = exercise3(qq1,d)
plt.subplot(2,2,1)
plt.scatter(quant_x, qq1, c = "red")
ax = plt.gca()
ax.annotate('quantiles', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('qq1', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', te

quant_x = exercise3(qq2,d)
plt.subplot(2,2,2)
plt.scatter(quant_x, qq2, c = "blue")
ax = plt.gca()
ax.annotate('quantiles', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('qq2', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', te

quant_x = exercise3(qq3,d)
plt.subplot(2,2,3)
plt.scatter(quant_x, qq3, c = "green")
ax = plt.gca()
ax.annotate('quantiles', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('qq3', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', te


quant_x = exercise3(qq4,d)
plt.subplot(2,2,4)
plt.scatter(quant_x, qq4, c = "orange")
ax = plt.gca()
ax.annotate('quantiles', xy=(0.98, 0), ha='left', va='top', xycoords='axes fraction')
ax.annotate('qq4', xy=(0, 1), xytext=(-15,2), ha='left', xycoords='axes fraction', te

# Through 4 tests, by observing the figures
# qq1 seems to be chi-squared distribution,
# qq2 seems to be exponential distribution,
# qq3 seems to be t_5 distribution,
# qq4 seems to be normal distribution
```