```
# 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/advanced_graphs

# In this exercises we will use four data sets which come from four unknown probability
# distributions. One of them comes from a N(0,1) distribution, another a t_5 distribution
# 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 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 distribution
# function (ecdf). This is defined as:
# Number of observations z less than or equal to z_i / Number of observations, for every z_i in
# Write a function called which takes a set of observations z and produces the empirical cdf
# If you are unfamiliar with empirical cdfs, you may want to read the following article:
# 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 single figure wi
# Save your figure and include it in your submission.
x1=np.sort(qq1)
y1=exercise1(qq1)
x2=np.sort(qq2)
y2=exercise1(qq2)
x3=np.sort(qq3)
y3=exercise1(qq3)
x4=np.sort(qq4)
y4=exercise1(qq4)
plt.figure()
plt.subplot(2,2,1)
plt.plot(x1,y1,marker='.',linestyle='None')
plt.margins(0.02)
```

```
plt.xlabel('Observations z')
plt.ylabel('ECDF')
plt.title('qq1')
plt.subplot(2,2,2)
plt.plot(x2,y2,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('Observations z')
plt.ylabel('ECDF')
plt.title('qq2')
plt.subplot(2,2,3)
plt.plot(x3,y3,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('Observations z')
plt.ylabel('ECDF')
plt.title('qq3')
plt.subplot(2,2,4)
plt.plot(x4,y4,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('Observations z')
plt.ylabel('ECDF')
plt.title('qq4')
#plt.subplots_adjust(left=0.9, bottom=0.9, right=1, top=1,
#wspace=0.4, hspace=0.4)
plt.tight_layout()
plt.savefig('Q1.pdf')

# Q2 For the next part we need to create the quantiles of a chosen probability distribution
# 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):
    a=np.random.normal(0,1,10000)
    b=np.quantile(a,ecdf)
    return b

# Create a scatter plot of the theoretical quantiles from your new function (x-axis) against qq1
# this for each dataset, creating each plot as a subplot on the same figure. Save your figure an
# submission. If the two distributions match, the points should lie on a straight line - this is
# the datasets is normally distributed variable?
x1=exercise2(y1)
x2=exercise2(y2)
x3=exercise2(y3)
x4=exercise2(y4)
y1=np.sort(qq1)
y2=np.sort(qq2)
y3=np.sort(qq3)
y4=np.sort(qq4)
plt.figure()
plt.subplot(2,2,1)
plt.plot(x1,y1,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('x')
plt.ylabel('y')
plt.title('qq1')
plt.subplot(2,2,2)
plt.plot(x2,y2,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('x')
plt.ylabel('y')
plt.title('qq2')
plt.subplot(2,2,3)
```

```
plt.plot(x3,y3,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('x')
plt.ylabel('y')
plt.title('qq3')
plt.subplot(2,2,4)
plt.plot(x4,y4,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('x')
plt.ylabel('y')
plt.title('qq4')
plt.tight_layout()
plt.savefig('Q2.pdf')

# Ans:
#datasets qq3 and qq4 are normally distributed variable.


# Q3 Create a new function that takes two arguments. The first argument should be your data Ser
# argument should be a set of simulations from some probability distribution. It should use thes
# the theoretical quantiles. This function should the computed theoretical quantiles
def exercise3(y,d):
    ecdf=exercise1(y)
    b=np.quantile(d,ecdf)
    return b
# 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 determine which
# data set matches to which probability distribution
# Complete the quiz 'W8 - Assessed exercises Q4' to submit your answer for this question.
x1=exercise3(qq1,d)
x2=exercise3(qq2,d)
x3=exercise3(qq3,d)
x4=exercise3(qq4,d)
y1=np.sort(qq1)
y2=np.sort(qq2)
y3=np.sort(qq3)
y4=np.sort(qq4)
plt.figure()
plt.subplot(2,2,1)
plt.plot(x1,y1,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('x')
plt.ylabel('y')
plt.title('qq1')
plt.subplot(2,2,2)
plt.plot(x2,y2,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('x')
plt.ylabel('y')
plt.title('qq2')
plt.subplot(2,2,3)
plt.plot(x3,y3,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('x')
plt.ylabel('y')
plt.title('qq3')
plt.subplot(2,2,4)
```

```
plt.plot(x4,y4,marker='.',linestyle='None')
plt.margins(0.02)
plt.xlabel('x')
plt.ylabel('y')
plt.title('qq4')
plt.tight_layout()
```