Dimensionality analysis of Stock Prices / Notebook 2

In this notebook we will study the dimensionality of stock price sequences, and show that they lie between the 1D of smooth functions and 2D of rapidly varying functions.

The mathematicians Manuel Mandelbrot and Richard Hudson wrote a book titled The Misbehavior of Markets: A Fractal View of Financial Turbulence (https://www.amazon.com/gp/product/0465043577?ie=UTF8&tag=trivisonno-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=0465043577). In this book they demonstrate that financial sequences have a fractal dimension that is higher than one. In other words, the changes in stock prices are more similar to random walk, than to a smooth differentiable curve.

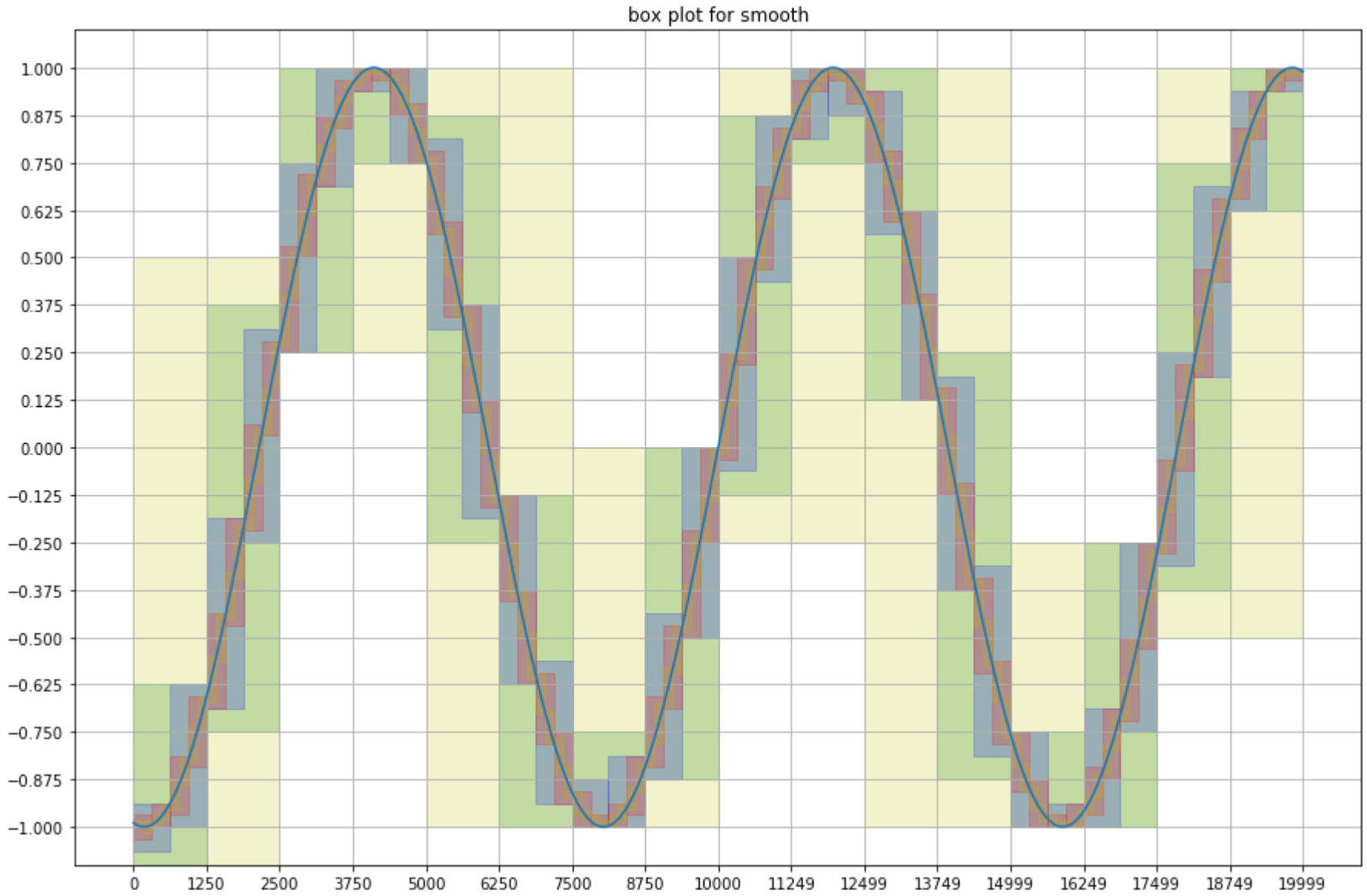
In this notebook we will estimate the fractal dimension of sequences corresponding to the log of the price of a stock. We will do the same for some other, non-random sequences.

We will use the Box Counting (https://en.wikipedia.org/wiki/Box_counting) method to estimate the dimension.

Box Counting

For the sake of simplicity, lets start with a simple smooth curve corresponding to $sin(x)$. Intuitively speaking, the dimension of this curve should be 1. Lets see how we measure that using box-counting.

The idea is simple: we split the 2D plane into smaller and smaller rectangles and count the number of rectangles that touch the curve. The gridlines in the figure below partition the figure into $16 \times 16 = 256$ rectangles. The yellow shading corresponds the partition of the figure into 8×8 rectangles. The green corresponds to the partition into 16×16 (which is the same as the grid), The blue and the red correspond to partitions into 32×32 and 64×64 respectively. You can see that as theboxes get smaller their number increases.



The dimension is defined by the relation between the size of the cubes and the number of rectangle that touch the curve. More precisly, we say that the size of a rectangle in a $n \times n$ partition is $\epsilon = 1/n$. We denote by $N(\epsilon)$ the number of rectangles of size ϵ that touch the curve. Then if d is the dimension, the relationship between $N(\epsilon)$ and ϵ is

$$N(\epsilon) = \frac{C}{\epsilon^d}$$

For some constant C

Taking $\log$s of both side we get

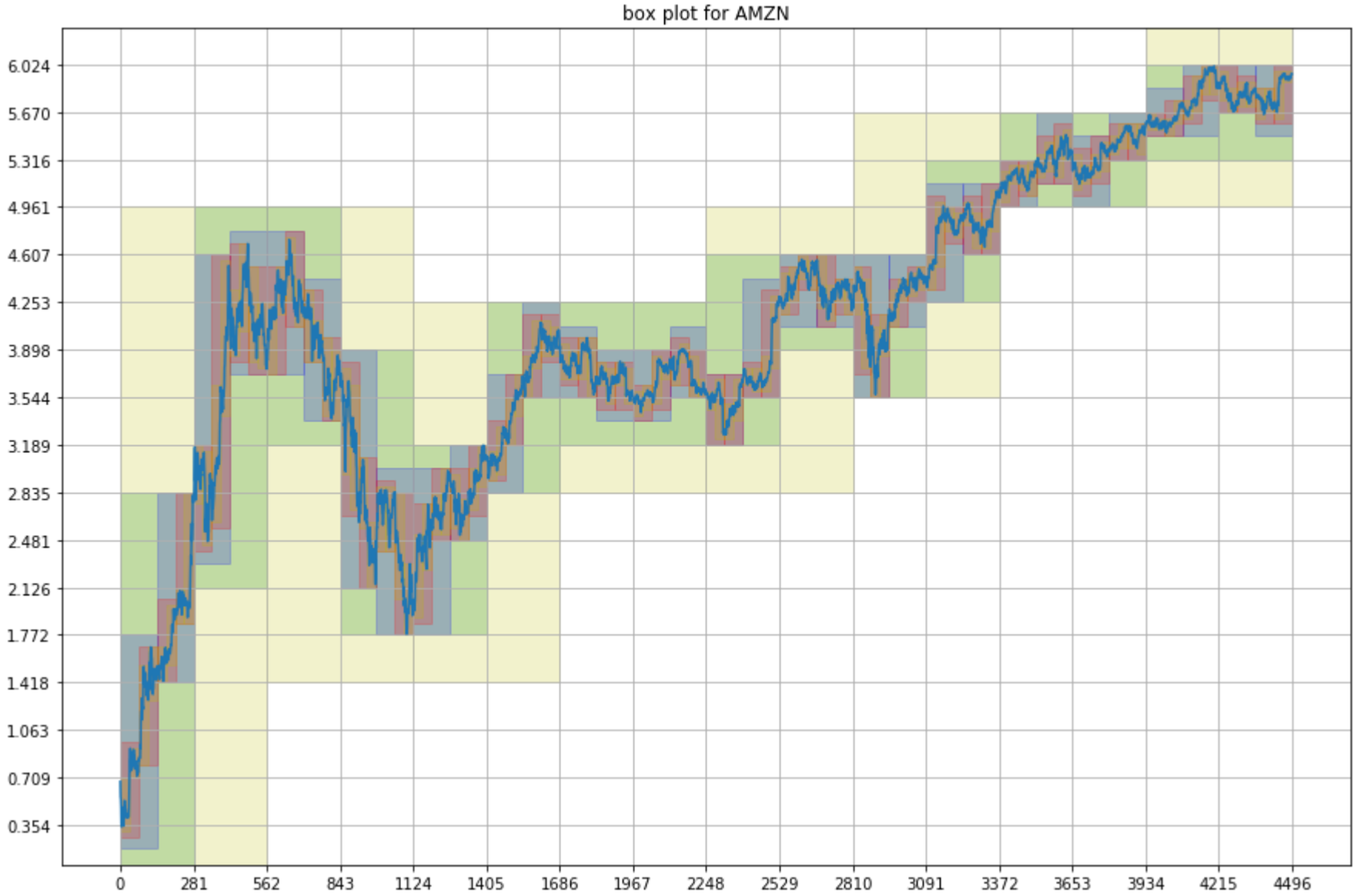
$$\log N(\epsilon) = \log C + d \log \frac{1}{\epsilon} \tag{1}$$

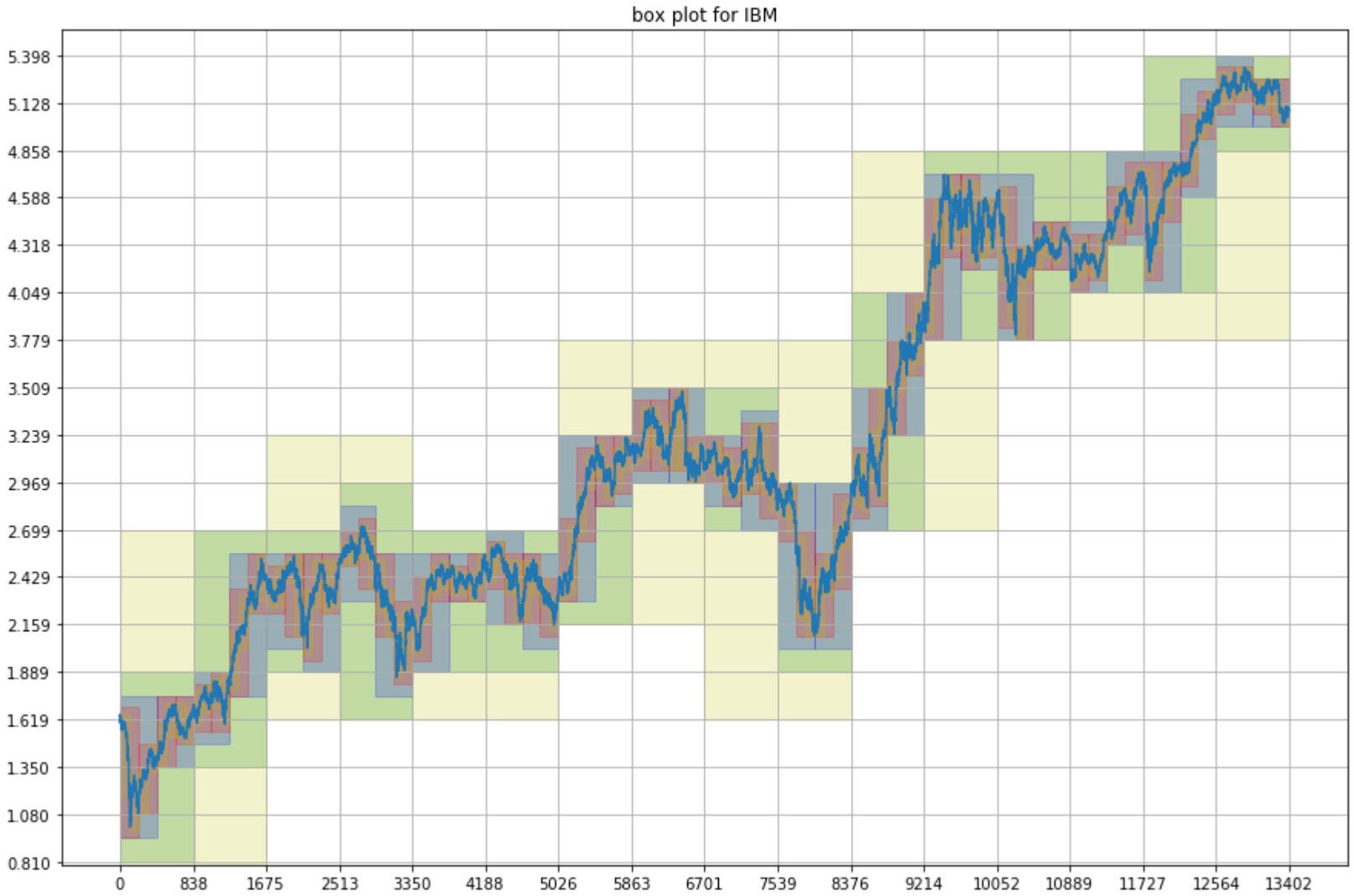
We can use this equation to estimate d as follows: let $\epsilon_2 \gg \epsilon_1$ be two sizes that are far apart (say $\epsilon_1 = 1/4$ and $\epsilon_2 = 1/1024$), and let $N(\epsilon_1)$, $N(\epsilon_2)$ be the corresponding box counts. Then by taking the difference between Equation (1) for the two sizes we get the estimate

$$d \approx \frac{\log N(\epsilon_1) - \log N(\epsilon_2)}{\log \epsilon_2 - \log \epsilon_1}$$

Note that this is an estimate, it depends on the particular values of ϵ_1 and ϵ_2. We can refer to it as the "dimension" if we get the same number for any choice of the two sizes (as well as other details sich as the extent of the function.

Here are similar figures for the seque


box plot for AMZN
6.024
5.670
5.316
4.961
4.607
4.253
3.898
3.544
3.189
2.835
2.481
2.126
1.772
1.418
1.063
0.709
0.354
0
281
562
843
1124
1405
1686
1967
2248
2529
2810
3091
3372
3653
3934
4215
4496



Download the data files

You might have made a mistake in producing the file `data/SP500.csv` . To avoid propagating this mistake, we download the correct file from S3

In [1]:
```
!wget https://mas-dse-open.s3.amazonaws.com/Stocks/data.tgz
!tar xzvf data.tgz
```

```
--2020-06-11 00:33:31--  https://mas-dse-open.s3.amazonaws.com/Stocks/data.tgz (https://mas-dse-open.s3.amazonaws.com/Stocks/data.tgz)
Resolving mas-dse-open.s3.amazonaws.com (mas-dse-open.s3.amazonaws.com)... 52.218.252.187, 52.218.252.187, 52.218.237.179
Connecting to mas-dse-open.s3.amazonaws.com (mas-dse-open.s3.amazonaws.com)|52.218.252.187|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 52006823 (50M) [application/x-tar]
Saving to: ‘data.tgz.2’

data.tgz.2          100%[===================>]  49.60M  2.11MB/s    in 24s

2020-06-11 00:33:55 (2.10 MB/s) - ‘data.tgz.2’ saved [52006823/52006823]

x data/
x data/PCA.pickle
x data/SP500.csv
x data/tickerInfo.tsv
```

Notebook Setup

In [2]:
```
#import findspark
#findspark.init()
from pyspark import SparkContext

#sc.stop()
sc = SparkContext(master="local[3]")

from pyspark.sql import *
sqlContext = SQLContext(sc)
```

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:
```
df=sqlContext.read.csv('data/SP500.csv',header='true',inferSchema='true')
df.count()
```

Out[4]:
```
13422
```

In [5]:
```
def get_seq(ticker):
    if ticker[0].isdigit():
        key='test/%s_P'%ticker
    else:
        key='train/%s_P'%ticker
    L=df.select(key).collect()

    L=[x[key] for x in L if (not x[key] is None) and x[key] >0 ]
#     print(L[0])
    return L
```

In [6]:
```
def calc_splitpoints(R,count):
    return np.arange(R[0],R[1],(R[1]-R[0])/(count*1.00001))
```

Exercise

Your task in this exercise is to compute the dimensionality of a stock price sequence. The sequence contains the log prices of a single stock over the course of 30+ years. We shall use the box counting approach described above to deduce the dimenionality. Specifically, you are required to do the following:

- Take the 2D region that contains the stock price sequence. The X-axis represents the ***day*** and the Y-axis represents the ***log of the stock price*** on that day. Divide this 2D space into rectangles of equal sizes such that the 2D space is divided into a grid of $N \times N$ rectangles where $N \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$
  - For each value of N, keep track of the number of boxes that the curve passes through. You can do this by using the intuition that the y-axis (log of prices) is a function of the x-axis (days). Assume, that the X-axis needs to be divided into N parts. Then we can find $N + 1$ points on the X-axis that divides it into N parts of equal length l. Each consecutive pair of points $[x_k, x_{k+1}]$ defines a range on the X-axis that will contain N rectangles of height h stacked on top of each other and spanning over the range $[y_{min}, y_{max}]$ where y_{min} is the minumum stock price and y_{max} is the maximum stock price over the entire span of time.
  - Within the set of N rectangles in the range $R = [x_k, xk + 1]$, the y values (log prices) will lie in a continuous interval y^R_{min}, y^R_{max} where y^R_{min} and y^R_{max} are the lower and upper limits of the stock price sequence in the range of R. By knowing this interval and the height of each rectangle, you can compute the number of rectangles that the sequence passes through in the range R. Repeating the process for each $R = [x_k, xk + 1]$ where $k \in 1, 2, 3...N$ gives us the total number of rectangles that the curve passes through.
- To find the dimensionality of the sequence, use the formula given above and the $\epsilon_1 = \frac{1}{16}, \epsilon_2 = \frac{1}{2048}$

Output Requirements: You are expected to return the following values in the order given below

1. ***box_counts***: A 2D numpy array that contains the number of boxes the sequence passed through for each value $N \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$. The first column in the array is the value of N and the second is the count of number of rectangles that the sequence passed through for that value of N. An example is shown below:

```
[[1.0 2.0000e+00]
 [2.0 5.0000e+00]
 [4.0 1.7000e+01]
 [8.0 4.4000e+01]
 [16.0 9.2000e+01]
 [32.0 1.8700e+02]
 [64.0 3.7800e+02]
 [128.0 7.5800e+02]
 [256.0 1.5120e+03]
 [512.0 2.9990e+03]
 [1024.0 5.8950e+03]
 [2048.0 1.1252e+04]]
```

2. ***dims***: A variable that contains the dimensionality of the sequence using the calculation given above
3. ***covers***: A dictionary where the keys are is $N \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$ and the value is an array containing tuples of the form ($x_k, x_{k+1}, floo Rec_{x_k,x_{k+1}}(y^R_{min}), ceil Rec_{x_k,x_{k+1}}(y^R_{max})$). Here, $floor Rec$ and $ceil Rec$ are defined as:

$$floor Rec_{x_k,x_{k+1}}(y) = \text{Floor of the rectangle that contains the point y, in the X-axis range } [x_k, x_{k+1}]$$
$$ceil Rec_{x_k,x_{k+1}}(y) = \text{Ceiling of the rectangle that contains the point y, in the X-axis range } [x_k, x_{k+1}]$$

For example if the the range of x values is $[0, 8]$ and $N = 4$, then we would have 4 intervals on the x-axis: $[0, 2], [2, 4], [4, 6], [6, 8]$. Assuming the y values lie in the range $[0, 6]$, the corresponding 4 intervals on the y-axis would be $[0, 1.5], [1.5, 3.0], [3.0, 4.5], [4.5, 6.0]$. The y^R_{min}, y^R_{max} represent the minimum and maximum values taken by the stock price sequence in each of the 4 intervals. Assuming these min and max values are $[3.5, 4.6], [2.5, 3.7], [1.2, 5.5], [5.2, 5.6]$ respectively, the key value pair in ***covers*** for $N = 4$ would be:

```
{4: [(0, 2, 3.0, 6.0), (2, 4, 1.5, 4.5), (4, 6, 0.0, 6.0), (6, 8, 4.5, 6.0)]} # Notice there are N entries in the value for key N
```

Hint: You might find the calc_splitpoints useful to split the X-axis into N segments

In [7]:
```
def Box_count(LL,ticker):
    ### Your code here

    # import libraries
    from collections import defaultdict
    import math

    # initialize dictionary for covers
    covers = defaultdict(list)

    # define list for NxN dimensions
    N_list = [1,2,4,8,16,32,64,128,256,512,1024,2048]

    # define count list for number of rectangles
    N = []

    # iterate over every grid size
    for n in N_list:

        # find rectangle height
        h = (max(LL)-min(LL))/(n * 1.00001)

        # use calc_splitpoints to find range points, round x for indexing
        x = [math.floor(i) for i in calc_splitpoints([0,len(LL)], n)]
        y = calc_splitpoints([min(LL),max(LL)*1.0001], n)

        # generate intervals
        x_inter = [(i,j) for i,j in zip(x[:-1], x[1:])]
        y_inter = [(i,j) for i,j in zip(y[:-1], y[1:])]

        # find stock intervals for x
        stock_price_x = [(min(LL[i:j]), max(LL[i:j])) for i,j in x_inter]

        # find floor and ceiling stock values for y
        floorRec = [j[0] for i in stock_price_x for j in y_inter if (i[0]>=j[0]) and (i[0]<=j[1])]
        ceilRec = [j[1] for i in stock_price_x for j in y_inter if (i[1]>=j[0]) and (i[1]<=j[1])]

        # add tuples covers to dictionary
        covers[n] = [(i[0],i[1],j,k) for i,j,k in zip(x_inter, floorRec, ceilRec)]

        # add box counts
        N.append([n, sum([int(round((i - j)/h)) for i,j in zip(ceilRec, floorRec)])])

    # calculate the dimension with 16 and 2048
```

```
    num = math.log([i[1] for i in N if i[0] == 16][0])-math.log([i[1] for i in N if i[0] == 2048][0])
    denom = math.log(1/2048)-math.log(1/16)
    dim = num/denom

    # convert N to array
    N = np.array(N)

    print('dim=',dim)
    return N,dim,covers
```

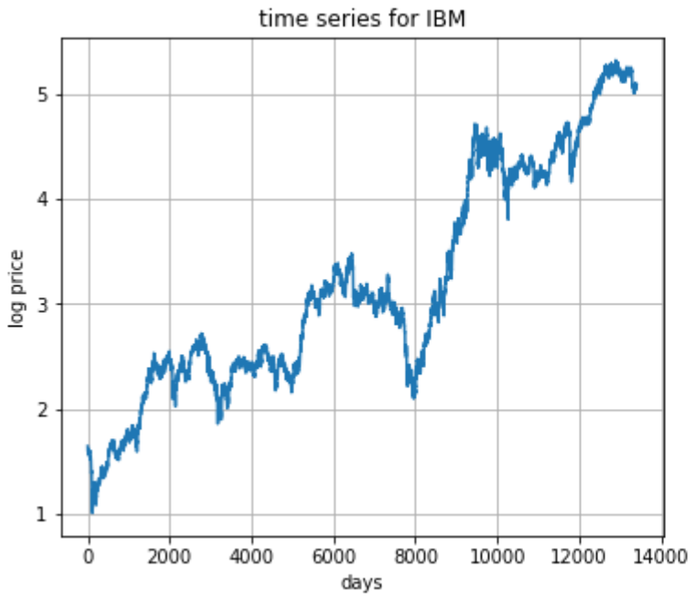
In [8]:
```
def plot_box_count(N,ticker):
    plt.plot(np.log(N[:,0]),np.log(N[:,1]));
    plt.title('Box Count Graph for '+ticker)
    plt.xlabel('log 1/epsilon')
    plt.ylabel('log N');
    plt.grid()
```

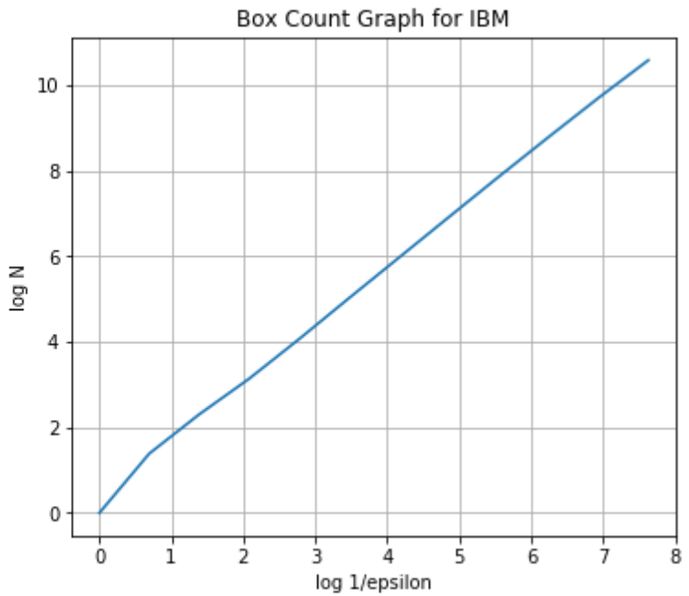
In [9]:
```
def analyze_sequence(L,ticker):
    plt.figure(figsize=(13,5))
    plt.subplot(121)
    LL=np.log(L)
#     print(LL)
    plt.plot(LL)
    plt.title('time series for '+ticker)
    plt.xlabel('days')
    plt.ylabel('log price')
    plt.grid()
    plt.subplot(122)
    N,dim,covers=Box_count(LL,ticker)
    plot_box_count(N,ticker)
    return N,dim,covers
```

In [10]:
```
def analyze_stock(ticker):
    L=get_seq(ticker)
#     print(L[0])
    N,dim,covers=analyze_sequence(L,ticker)
#     print(covers)
    return N,dim,covers,L
```

In [11]:
```
#(3 points)
ticker='IBM'
N,dim,covers,L=analyze_stock(ticker)
assert 1.1390537589837473 <= dim <= 1.541072732742717
```

```
dim= 1.3436178412057505
```





Plot graph with boxes

This portion of the code plots the stock price sequence and overlays it with the boxes that the sequence passes over. The ***covers*** dictionary returned by box_count is used to overlay the boxes. You can use the graphs generated to verify if you have identified the correct set of rectangles that pass through the sequence.

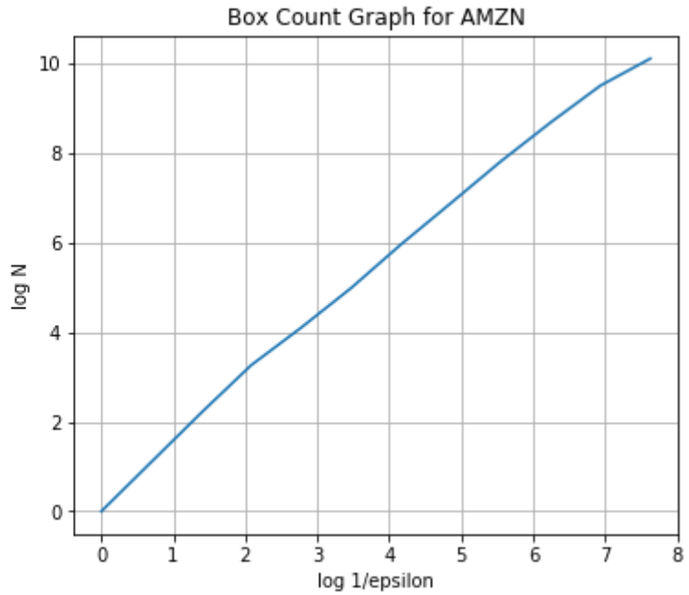
In [12]:
```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
```

In [13]:

```
import matplotlib.pyplot as plt
import matplotlib.ticker as plticker
G=16
def plot_boxes(LL,covers):
    plt.figure(figsize=[15,10])
    plt.plot(LL)
    axes=plt.gca()
    xmin=0.0; xmax=len(LL)-1.0
    ymin=min(LL)
    ymax=max(LL)
    yloc = plticker.MultipleLocator((ymax-ymin)/G)
    xloc = plticker.MultipleLocator((xmax-xmin)/G)
    axes.xaxis.set_major_locator(xloc)
    axes.yaxis.set_major_locator(yloc)

    # Add the grid
    axes.grid(which='major', axis='both', linestyle='-')

    colors='gbrygbrygbrygbry'
    for level in range(3,8):
        N = 2**level
        blocks=covers[N]

        for xmin,xmax,ymin,ymax in blocks:
            width=xmax-xmin
            height=ymax-ymin
            axes.add_patch(
                patches.Rectangle(
                    (xmin, ymin),   # (x,y)
                    width,          # width
                    height,          # height
                    alpha=0.2,color=colors[level]
                ))
```

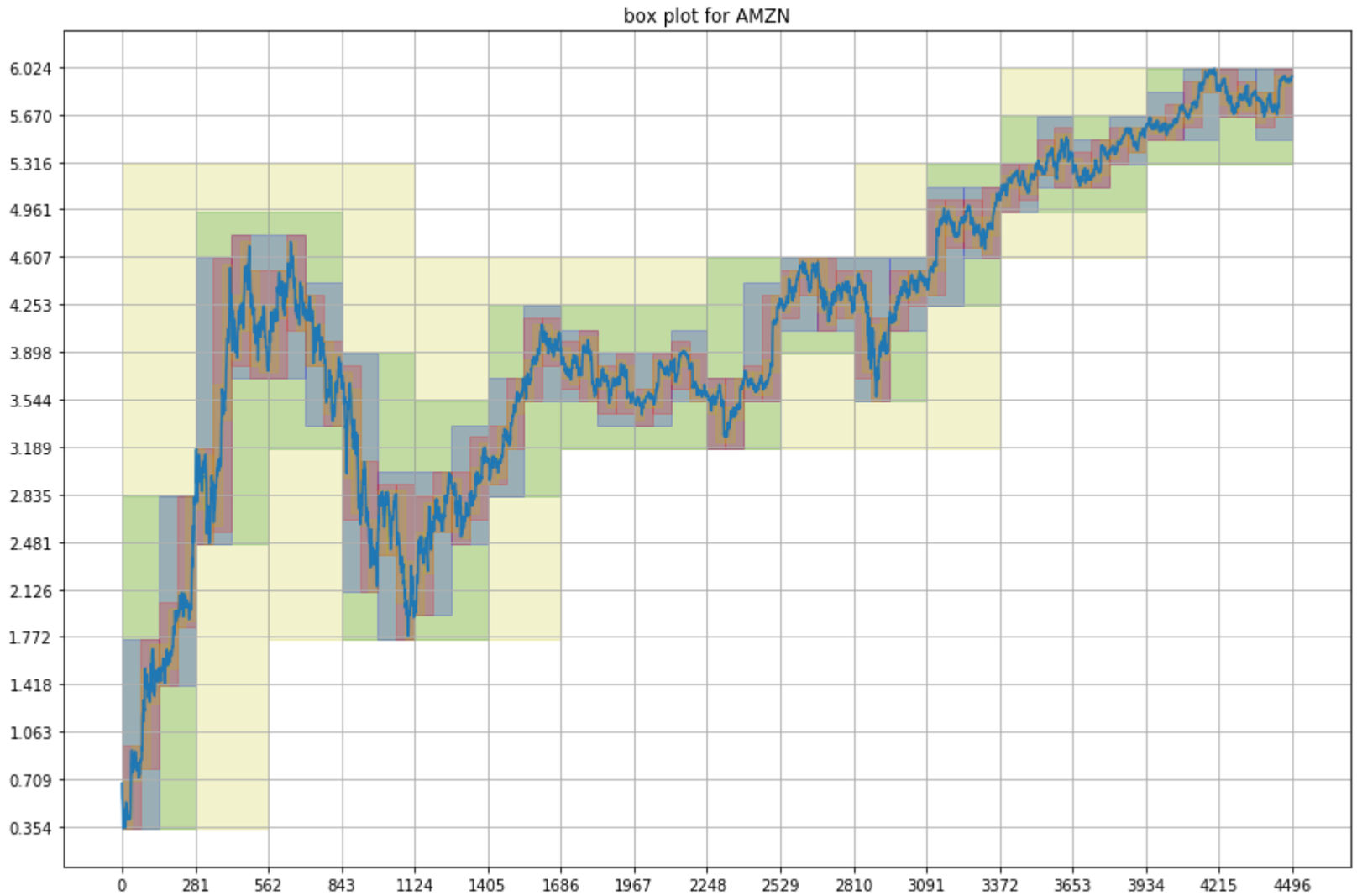
In [14]:
```
#(4 points)
ticker='AMZN'
N,dim,covers,L=analyze_stock(ticker)
assert 1.049035790021836 <= dim <= 1.4192837159118956
```

```
dim= 1.2376850389375027
```



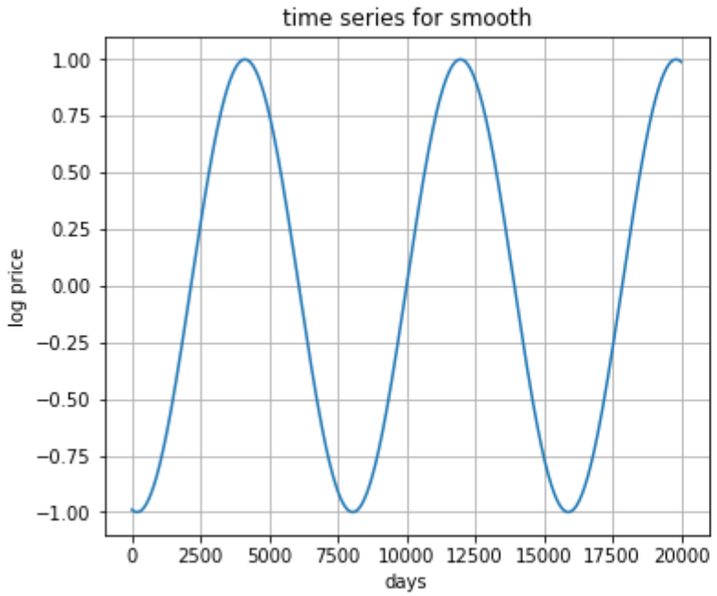
In [15]:
```
LL=np.log(L)
plot_boxes(LL,covers)
plt.title('box plot for '+ticker)
```

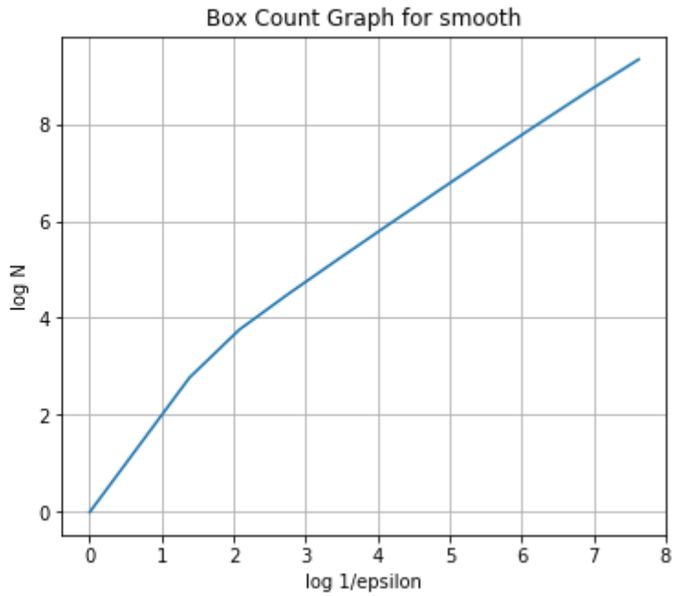
Out[15]:
```
Text(0.5, 1.0, 'box plot for AMZN')
```



In [16]:
```
#(3 points)
x=np.arange(-1,1,0.0001)
y=np.exp(np.sin(8*x))
N,dim,covers=analyze_sequence(y,'smooth')

assert 0.842026014152719 <= dim <= 1.1392116662066198
```

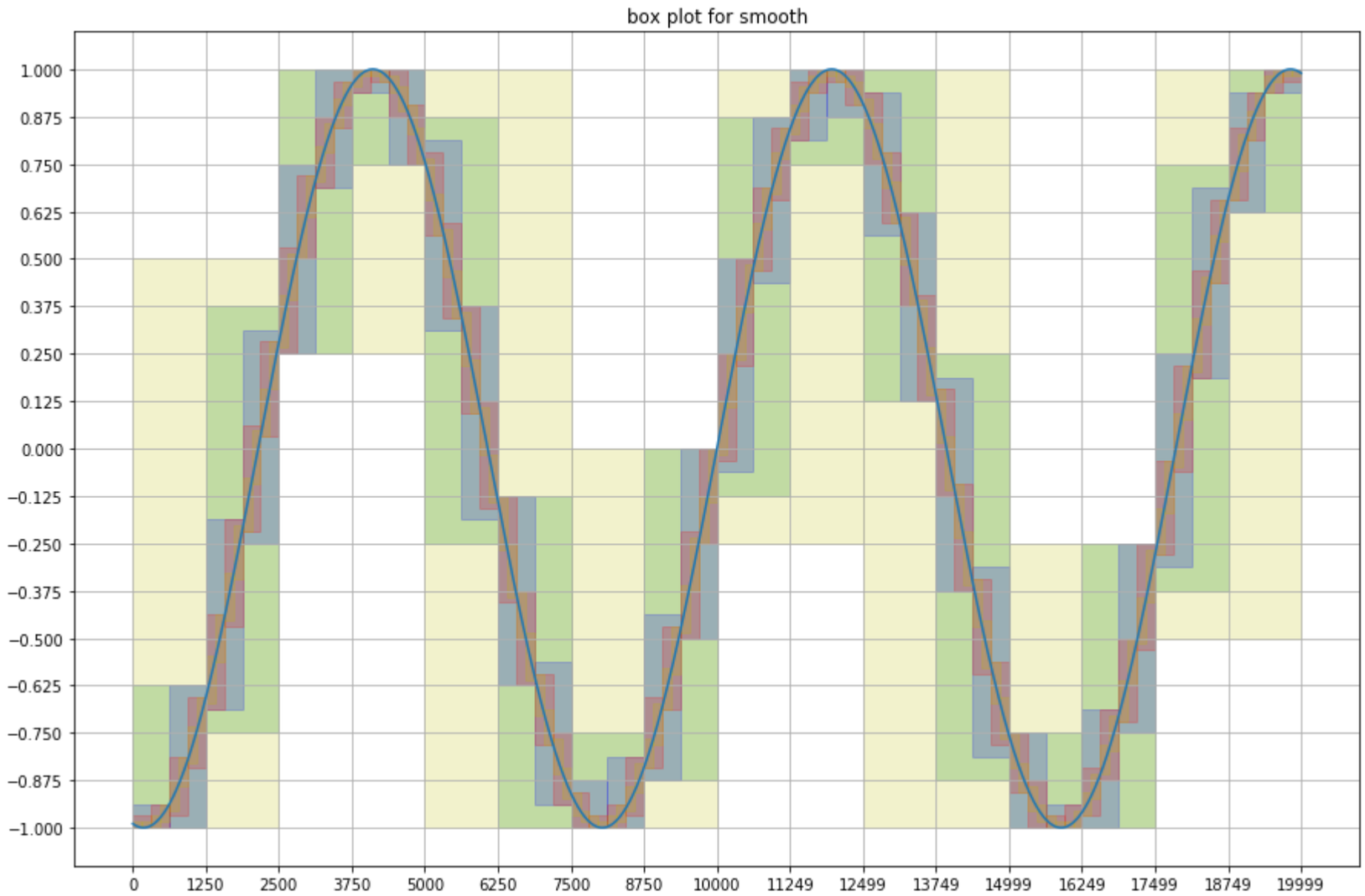
```
dim= 0.992761384477444
```





In [17]:
```
LL=np.log(y)
plot_boxes(LL,covers)
plt.title('box plot for '+'smooth')
```

Out[17]:
```
Text(0.5, 1.0, 'box plot for smooth')
```



In [18]:
```
#(5 points)
# Hidden tests
```

In [19]:
```
#(5 points)
# Hidden tests
```

In [20]:
```
#(5 points)
# Hidden tests
```

In [21]:
```
#(5 points)
# Hidden tests
```

In [ ]:

In [ ]: