# Programming for Cybersecurity
# Lab 06.01: regression

Lecturer: Andrew Beatty

In this lab we are going to use Python make some regression graphs,

You will need to pip install seaborn numpy, pandas if you have not done so already.

## Getting started (lab2-06.01)

1. Write a program that loads the seaborn *tips* data and prints out the top 5 rows (ie the head)

```
import seaborn as sns
import matplotlib.pyplot as plt

# load the dataset
dataset = sns.load_dataset('tips')

print(dataset.head())
```

2. Modify the script produce a regression plot for the tips against the total bill .

```
import seaborn as sns
import matplotlib.pyplot as plt

# load the dataset
dataset = sns.load_dataset('tips')

# the for debugging
#print(dataset.head())
sns.set_style('whitegrid')
sns.lmplot(x='total_bill', y='tip', order=1, data=dataset)

plt.show()
```

3. Mess around with this to produce a scatter plot,a regplot and a

## A bit more on regression (lab2-06-2)

4. Make plot using discrete values for example tips against size

```
sns.lmplot(x="size", y="tip", data=dataset)
```

5. Put in a jitter to make it easier to see

```
sns.lmplot(x="size", y="tip", data=dataset, x_jitter=.05)
```

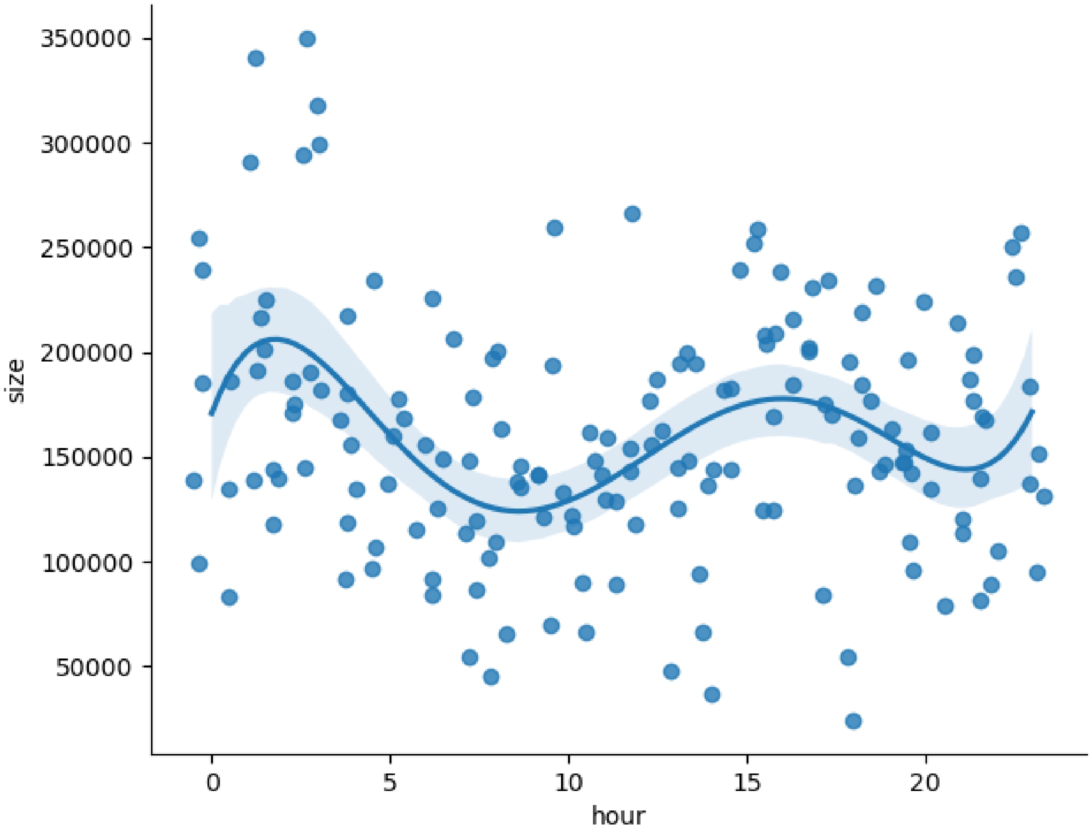
6. Instead of dots, use an estimator to estimate the mean tip for each size

```
sns.lmplot(x="size", y="tip", data=dataset, x_estimator=np.mean)
```

## Log files again

Remember the log files from semester 1 and how we put them into a data frame.

The next series of tasks is to create a regression plot for the amount of data that was transferred for each hour of the day. Using the data from a log file (you can use access.log file that we used in semester 1) . The plot will look something like this



Note that this is a 5th order polynomial regression.

The access.log file looks like this, the fields are separated by space and the strings have quotes and the datetime has [] so we will need to write functions to read in those fields

```
199.15.234.66 - - [15/Feb/2021:22:24:31] "GET
/cart.do?action=view&itemId=EST-6&productId=SC-MG-
G10&JSESSIONID=SD5SL9FF2ADFF4958 HTTP 1.1" 200 3033
"http://www.google.com" "Mozilla/5.0 (Windows; U; Windows NT 5.1;
en-US; rv:1.9.2.28) Gecko/20120306 YFF3 Firefox/3.6.28 ( .NET CLR
3.5.30729; .NET4.0C)" 177
```

1. Write a function to read in the strings, it should strip off the firsts and last characters

```
def parse_str(x):
    return x[1:-1]
```

2. Write a function to read in the date in the format it is in the log file, it will also need to strip the first and last characters (ie the [])

```
def parse_datetime(x):

    dt = datetime.strptime(x[1:-1], '%d/%b/%Y:%H:%M:%S')
    return dt
```

3. Read in the log file into a data frame, remember the colums in the log file are.

```
['ip', 'time', 'request', 'status', 'size', 'referer', 'user_agent']
```

We will not need columns 1 or 2 (they are blank), and remember to assign the parsing functions for each column. The delimitator in this code looks like a scary regular expression.

```
df = pd.read_csv(
    'data/access.log',
    sep=r'\s(?=(?:[^"]*"[^"]*")*[^"]*$)(?![^\[]*\])',
    engine='python',
    na_values='-',
    header=None,
    usecols=[0, 3, 4, 5, 6, 7, 8],
    names=['ip', 'time', 'request', 'status', 'size', 'referer',
'user_agent'],
    converters={'time': parse_datetime,
                'request': parse_str,
                'status': int,
                'size': int,
                'referer': parse_str,
                'user_agent': parse_str})
```

4. You can check that it was all read in correctly by either printing out the head() of the data frame or by storing the data frame into an excel and looking at it.

```
excelFilename = './data/log.xlsx'
df.to_excel(excelFilename, index=False, sheet_name='data')
```

5. NOTE you can manipulate the columns to get extract data from them eg you could get more information about the URI and URL (this is not needed for this excercise

```
request = df.pop('request').str.split()
df['resource'] = request.str[1]
df['method'] = request.str[0]
#yes I could have used regex for this
# from the request get the string before the ?
df['url'] = request.str[1].str.split('?').str[0]
```

6. OK. Now lets make another dataframe that stores the sum of all the data transferred for each hour (H)

```
dfbyhour=df.resample('H',on='time').sum()
```

7. You should inspect this dataframe, you will see the index has the time periods
8. We can then make another column in the dataframe for the hour of the day (I make one for the date as well while I am at it, though this is not required.

```
dfbyhour['hour'] = dfbyhour.index.hour
dfbyhour['date'] = dfbyhour.index.date
```

9. Now lets make a regression plot for the size of the transfer over the hour of the day, you can mess about with the order,

```
sns.lmplot(x="hour", y="size", order=1 ,data=dfbyhour, x_jitter=0.5)
```

10. we really should make a plot of the residual to see which order is most appropriate.

```
sns.residplot(x="hour", y="size", data=dfbyhour, order=1)
```

You will see that there is a bit of a pattern with this residual plot so increase the order until you can not see a pattern