

How linear regression analysis can be applied to moore's law?

transistor count from: https://en.wikipedia.org/wiki/Transistor_count
(https://en.wikipedia.org/wiki/Transistor_count)

Import Libraries

```
In [8]: import re
import numpy as np
import matplotlib.pyplot as plt
```

Create empty list of X and Y

```
In [9]: # Define a variable to remove non-decimal characters
X = []
Y = []

# some numbers show up as 1,170,000,000 (commas)
# some numbers have references in square brackets after them
non_decimal = re.compile(r'^\d+')
```

Create Numpy arrays for x and y

```
In [10]: # X = Years data in 3rd column
# Y = no of transistors data in 2nd column

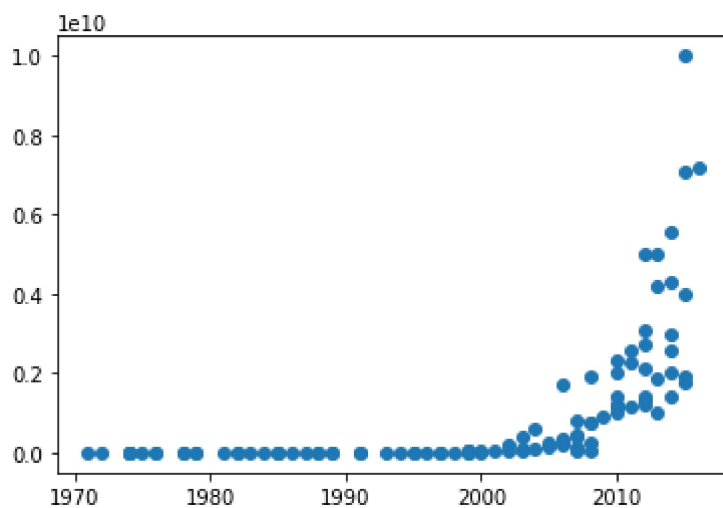
for line in open('moore.csv'):
    r = line.split('\t')

    x = int(non_decimal.sub('', r[2].split('[')[0]))
    y = int(non_decimal.sub('', r[1].split('[')[0]))
    X.append(x)
    Y.append(y)

X = np.array(X)
Y = np.array(Y)
```

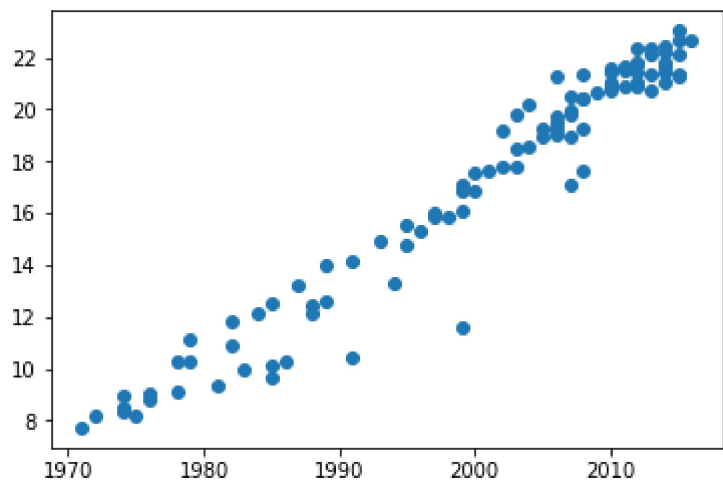
Show plot between transistor count and years data

```
In [11]: # Show exponential curve
plt.scatter(X, Y)
plt.show()
```



To check if Linear Regression can be applied

```
In [12]: Y = np.log(Y)
plt.scatter(X, Y)
plt.show()
```

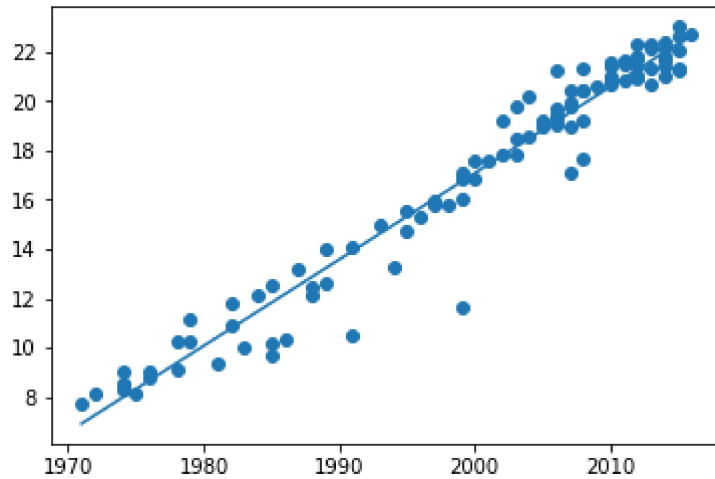


```
In [13]: # Solution for Linear Regression
denominator = X.dot(X) - X.mean() * X.sum()
a = ( X.dot(Y) - Y.mean()*X.sum() ) / denominator
b = ( Y.mean() * X.dot(X) - X.mean() * X.dot(Y) ) / denominator
```

Plot the line of best fit

```
In [14]: # Let's calculate the predicted Y
Yhat = a*X + b

plt.scatter(X, Y)
plt.plot(X, Yhat)
plt.show()
```



Determine how good the model is by computing the r-squared

```
In [15]: d1 = Y - Yhat
d2 = Y - Y.mean()
r2 = 1 - d1.dot(d1) / d2.dot(d2)
print("a:", a, "b:", b)
print("the r-squared is:", r2)

a: 0.35104357336496683 b: -685.0002843816004
the r-squared is: 0.9529442852285762
```

how long does it take to double?

```
In [16]: # log(transistorcount) = a*year + b
# transistorcount = exp(b) * exp(a*year)
# 2*transistorcount = 2 * exp(b) * exp(a*year) = exp(ln(2)) * exp(b) * exp(a *
year) = exp(b) * exp(a * year + ln(2))
# a*year2 = a*year1 + ln2
# year2 = year1 + ln2/a
print("time to double:", np.log(2)/a, "years")

time to double: 1.9745331723800172 years
```

The time that takes the transistor count to double is approximately 2 years.