Nathan Burwig
Math 87 HW 7
Due 11/12/2022

---

1 The Germinator

Suppose we are monitoring a population of bacteria in a lab. Initially there are 500 organisms. You've predicted that everyday the population grows by 6.35%. Furthermore, suppose that you introduce an additional m organisms to the population everyday.

1. **Write a recurrence relation for the population after k days, p_k, as a function of p_{k-1} and m.**

   We know that we have an initial population of 500, and that number will grow every day based on the percentage growth of 6.35% and that we also have some random number of m additional organisms joining per day. Thus we can describe the population growth in general by...

   $$p_k = 1.0635p_{k-1} + m$$

2. **Solve this recurrence relation for p_k as a function of m and k (i.e. no explicit reference to p_{k-1}).**

   Generally a good method to go about solving these kinds of problems are to just write out the first few terms of the serires and attempt to notice a pattern. Inductive proofs can be handy but are oftentimes overkill. Let's see if we can find something here...

   $$\begin{aligned} p_0 &= 500 \\ p_1 &= 1.0635p_0 + m \\ p_2 &= 1.0635p_1 + m = 1.0635^2p_0 + 1.0635m + m \\ p_3 &= 1.0635^3p_0 + 1.0635^2m + 1.0635m + m \end{aligned}$$

   It is here we notice a pattern emerge, and we can extrapolate to determine what this would look like for general k.

   $$p_k = 1.0635^k p_0 + (1.0635^{k-1} + 1.0635^{k-2} + \cdots + 1)m$$

   The parenthetical section is a geometric seires, thus we can sum it using the summation equation for a geometric series. It would then become...

   $$-\frac{m(1 - 1.0635^k)}{0.0635}$$

   We then have the general form for p_k to be...

   $$p_k = 1.0635^k \cdot 500 - \frac{m(1 - 1.0635^k)}{0.0635}$$

3. **Suppose that you are breeding the bacteria for an experiment, and need a fixed number of organisms g. Find an expression for the day, k, in which your desired number of organisms g, as a function of m.**

   What we want is to say there is some desired number of organism, and what day k will give us that desired amount. If we set our equation p_k equal to the desired number g, then we can solve it for k. A highlight of the algebra goes as follows.

$$\begin{aligned} g &= 1.0635^k \cdot 500 - \frac{m(1 - 1.0635^k)}{0.0635} \\ 0.0635g + m &= 1.0635^k(500 \cdot 0.0635 + m) \\ \ln\left[\frac{0.0635 + m}{31.75 + m}\right] &= k \ln[1.0635] \\ k &= \frac{\ln[0.0635 + m] - \ln[31.75 + m]}{\ln[1.0635]} \end{aligned}$$

   And that is our expression for day k

4. **Write a python function that takes in goal population g and the added amount of organisms m, and returns the number of days, k, to reach the goal. Suppose the goal balance is 200,000 organisms. How many days does it take to reach the goal if we introduce 100 bacteria a day? 250? How many months does it take to reach a goal of 2,000,000 introducing 100 and 250?**

   We can quickly write a ptyhon function to check these values. Taking $m = 100$ with a goal population of $200,000$, it would take 75 days. If $m = 250$ with the same goal, then it takes only 63 days.

   Evaluating how many months it would take to reach $2,000,000$ organisms is a similar process, however this time we divide the number of days by the average length of the month, or about 30 days. For $m = 100$ we get 3.7 months, and if we gained 250 per day it would only take 3.3 months.

2 How effective is regression?

In this exercise, we investigate the quality of a linear least squares regression as a function of the amount of noise in our data.

1. **Consider a linear equation $y = mx + b$. Precisely explain how one could randomly sample points satisfying this equation, under the constraints that the samples $0 \leq x \leq N$ for some N. Suppose you then performed least linear squares regression on the data. With enough samples, you should be able to recover the equation of the line. What is the minimum number of samples needed in order to do this?**

   In general, we could utilize the `random` function from `python` to randomly generate some numbers. Then, given the constraint that each point is on the line given by some m and b, we would have points on the line.

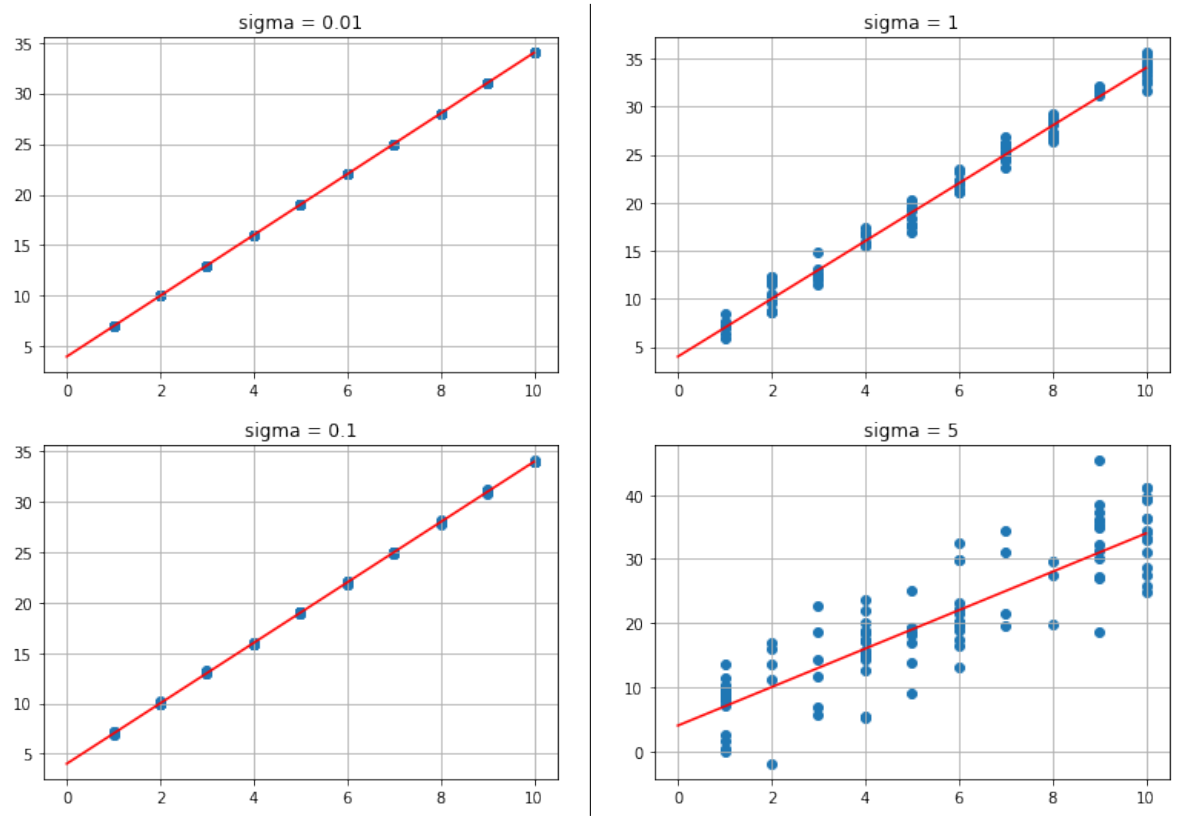
   We would generate those points by saying something along the lines of `x = random.randint(0, N)`. This would produce numbers randomly, and, if we got two of these points, that would be enough to reproduce the equation of the line.

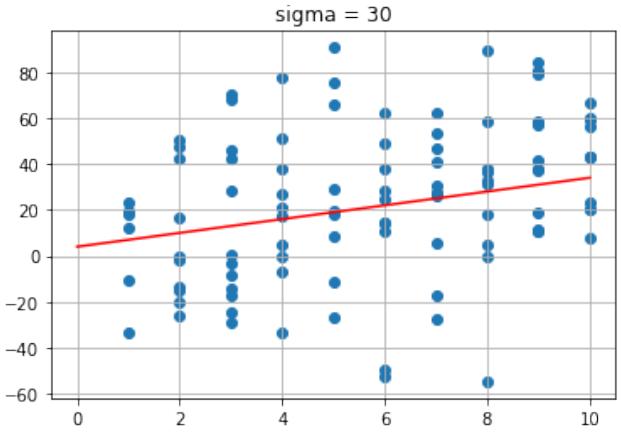
2. **Suppose that we have introduced some noise to our equation, i.e. our samples are drawn from $y = mx + b + \eta$, where η is drawn from a normal distribution with mean 0 and variance σ. One would hope that, with enough samples, one could recover the equation of the original line. Predict how σ affects the number of samples required to approximate the line.**

   We know that our variance will affect the width of the distribution of points we generate. If we have a set of points with a wider distribution, then we will need to sample more points the average out to the actual equation of the line. If our σ is smaller, then we will need fewer points in order to reproduce the equation of the line.

3. **For concreteness, let $m = 3, b = 4, M = 10$. Sample and plot (on separate plots) 100 points sampled from $y = mx + b + \eta$ for $\sigma = 0.01, 0.1, 1, 5, 30$ and plot the original (i.e. no noise) line over the data.**

   We can utilize `matplotlib` to help us plot these lines and label them based on the variance we use.

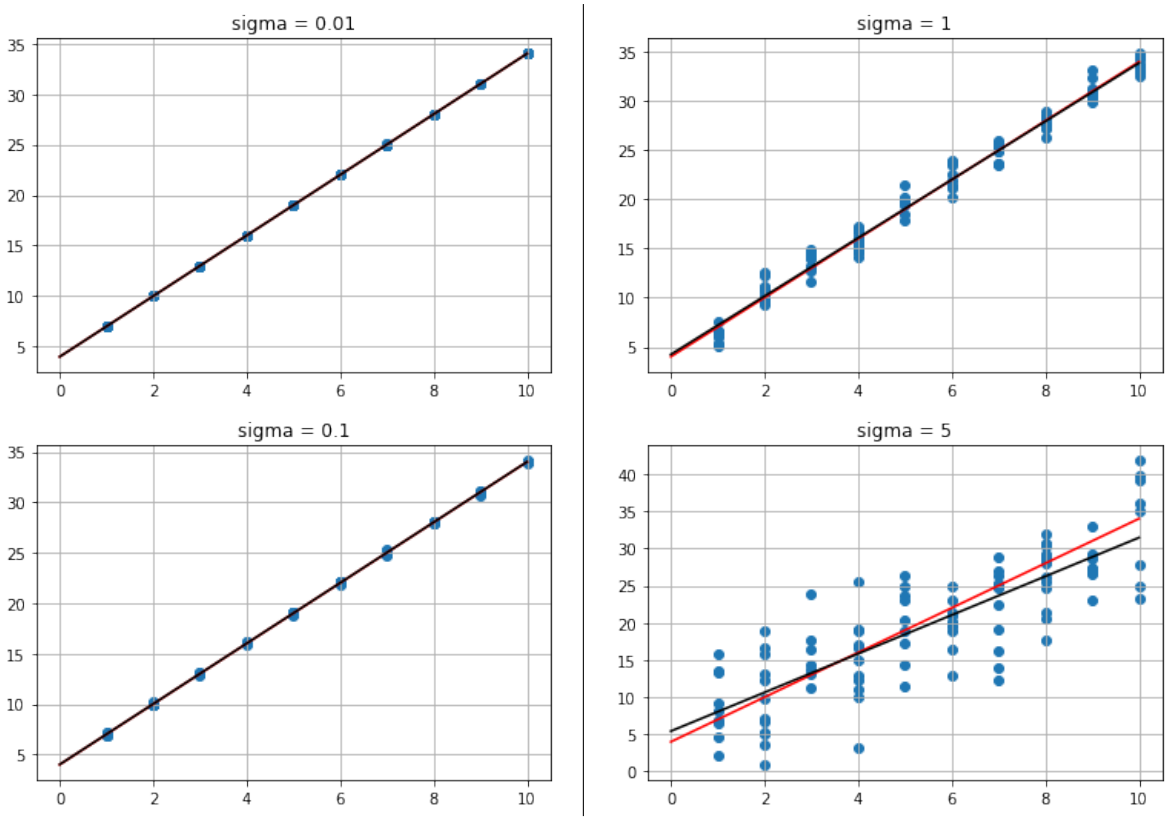


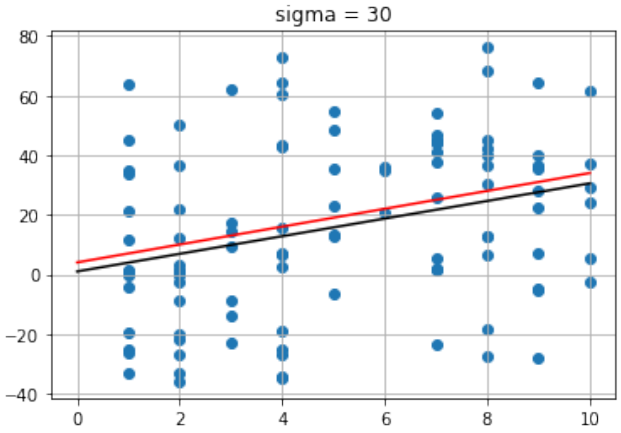


This is a fairly expected outcome.

4. **Perform least linear squares on the data obtained in the preceding part. The solution will give you the coefficients of a line. On each plot, plot this line over the data.**

   We can perform least squares regression quite easily using the `polyfit` module of `matplotlib.pyplot`. We can also code this manually, and for completness, I went ahead and did a couple out on paper to make sure I was getting numbers that were accurate to the plots given by polyfit. The results are given below.





We can see that it is just as we expected. The accuracy to the actual line is determined by the variance in our data, and as we increase that variance, we get farther away from represting the original line.

5. **Run an experiment to determine how many samples are needed in order for the least squares solution to be within 0.01 of the coefficients of the original line for each value of σ (you do not need to find the minimal such value, just some number of samples that works). For each value of σ, plot the error in slope (i.e. the difference between the slope of the original line and the one computed by least squares regression) for the following number of samples:** $20, 200, 2000, 20000, 200000$**.**

We can determine the error and plot them in a table based on the given sigma and sample size. I found the error by just using the standard error formula based on the actual slow ($m = 3$) and the slope obtained during the regression. The table plots as follows...

Error					
–	0.01	0.1	1	5	30
20	0.00051	0.00493	0.06094	0.09329	4.49227
200	9.270e-05	0.00041	0.05218	0.21477	0.75302
2000	1.897e-05	0.00032	0.00352	0.03483	0.16247
20000	8.058e-07	0.00016	0.00428	0.00875	0.06850
200000	5.410e-06	1.620e-05	0.00100	0.00433	0.007773

I've gone ahead and marked the cells which pass the 0.01 criteria in green. Across the top row is the variance, and the left most column is the sample size.

This confirms our previous thoughts, showing that the wider the variance, the more samples we need in order to get a smaller error between the two lines.