Simulation and optimization: demo notes

Optimization

Demo: `grades.xlsx`

1. To find what grade we need to get an A, we first need to calculate a plug for the final grade. We can do this by taking the SUMPRODUCT() of the current grades and weights.

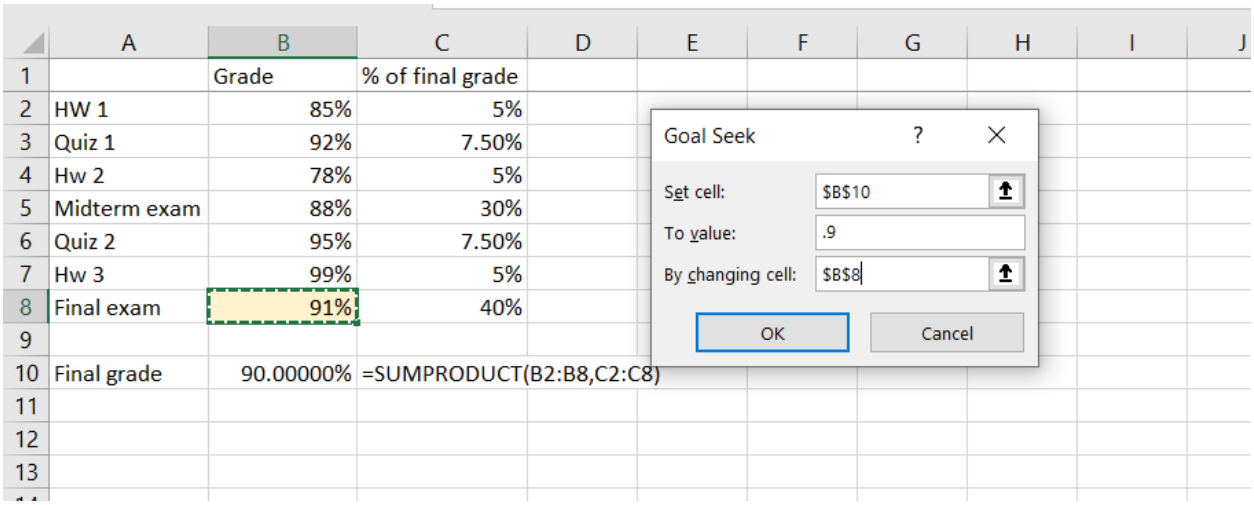
	A	B	C	D	E
1		Grade	% of final grade		
2	HW 1	85%	5%		
3	Quiz 1	92%	7.50%		
4	Hw 2	78%	5%		
5	Midterm exam	88%	30%		
6	Quiz 2	95%	7.50%		
7	Hw 3	99%	5%		
8	Final exam		40%		
9					
10	Final grade	53.52500%	=SUMPRODUCT(B2:B8,C2:C8)		
11					

2. You can now start to experiment on your own with that grade will result in an A by trial and error. But why not just have Excel do this for you?

	A	B	C	D	E
1		Grade	% of final grade		
2	HW 1	85%	5%		
3	Quiz 1	92%	7.50%		
4	Hw 2	78%	5%		
5	Midterm exam	88%	30%		
6	Quiz 2	95%	7.50%		
7	Hw 3	99%	5%		
8	Final exam	89%	40%		
9					
10	Final grade	89.12500%	=SUMPRODUCT(B2:B8,C2:C8)		
11					

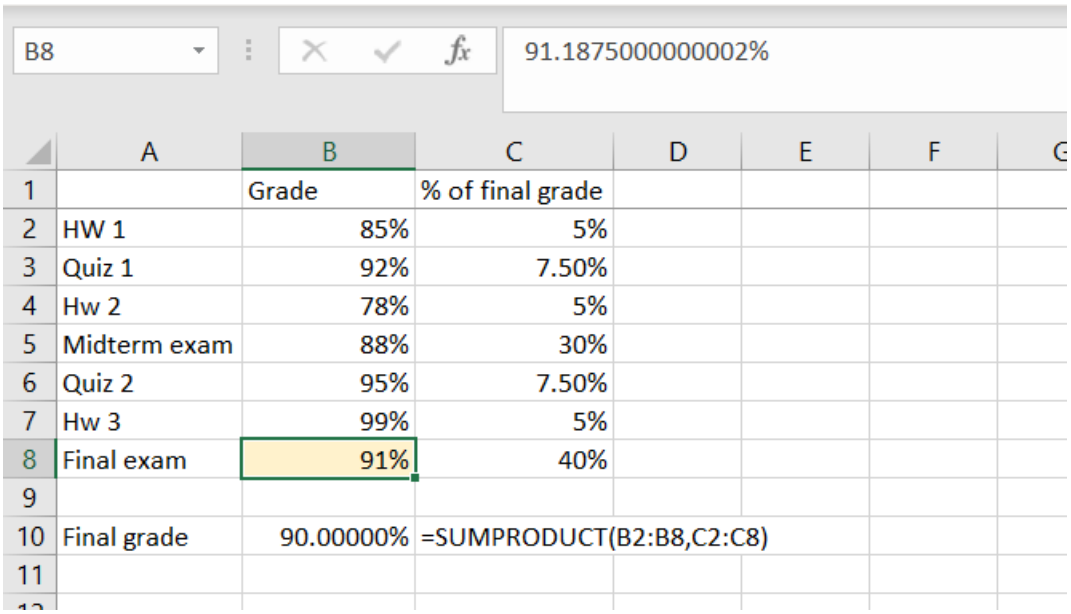
3. Select **Data > What-If Analysis > Goal Seek.**

   a. We want to set our *final grade* (B10) to 90% by changing our *final exam grade* (B8).



	A	B	C	D	E	F	G	H	I	J
1		Grade	% of final grade							
2	HW 1	85%	5%							
3	Quiz 1	92%	7.50%							
4	Hw 2	78%	5%							
5	Midterm exam	88%	30%							
6	Quiz 2	95%	7.50%							
7	Hw 3	99%	5%							
8	Final exam	91%	40%							
9										
10	Final grade	90.00000%	=SUMPRODUCT(B2:B8,C2:C8)							
11										
12										
13										

   b. It turns out that we need a ~91% on the final exam to get a 90% for the course.



B8 | 91.1875000000002%

	A	B	C	D	E	F	G
1		Grade	% of final grade				
2	HW 1	85%	5%				
3	Quiz 1	92%	7.50%				
4	Hw 2	78%	5%				
5	Midterm exam	88%	30%				
6	Quiz 2	95%	7.50%				
7	Hw 3	99%	5%				
8	Final exam	91%	40%				
9							
10	Final grade	90.00000%	=SUMPRODUCT(B2:B8,C2:C8)				
11							

Demo: `sales-price.xlsx`

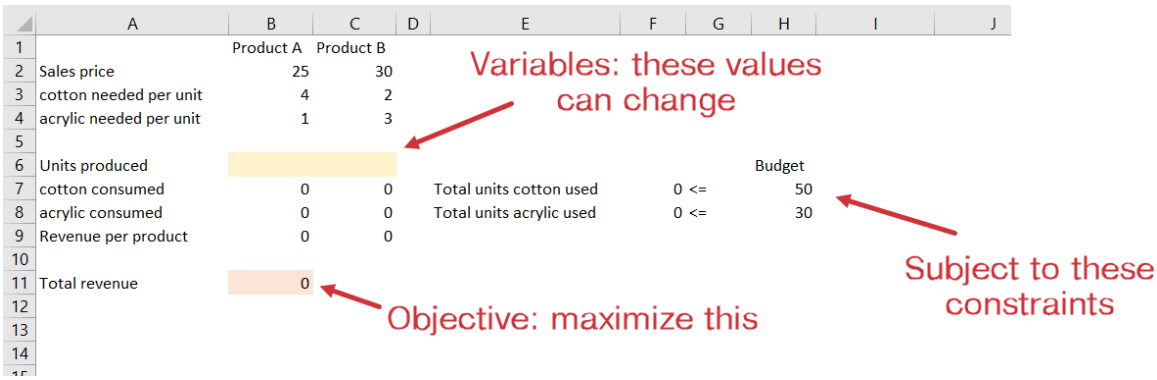
1. Notice that Goal Seek tells you to sell a fraction of a doo-hickey. That won't work! Goal Seek has some limitations, so let's get into using Solver.

Demo: `product-mix.xlsx`

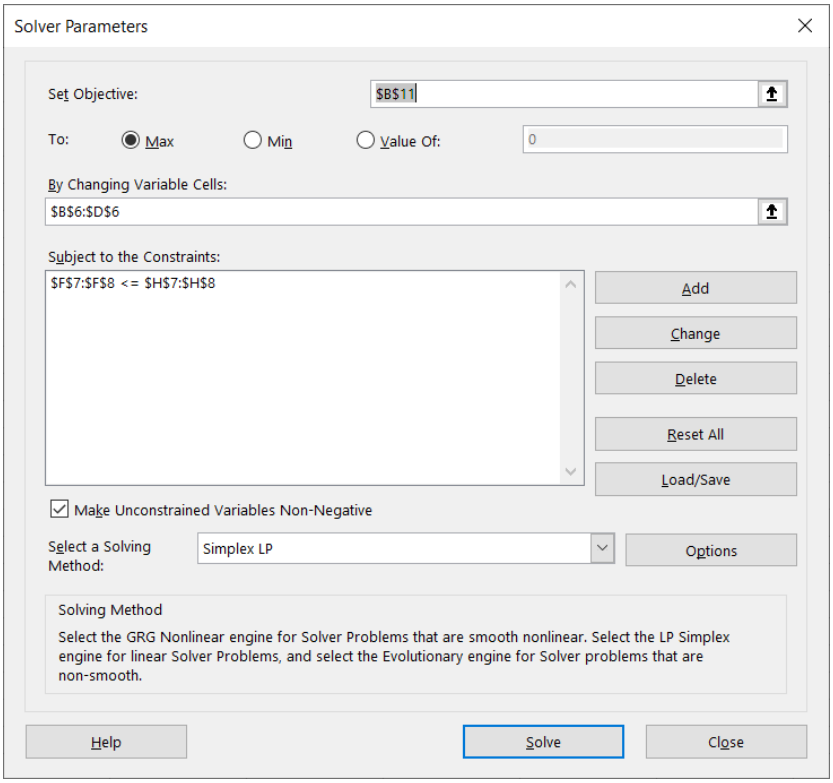
1. In this example, we use cotton and acrylic to make two different types of products. Each takes a different blend of each product, and we only have so much cotton and acrylic. How do we optimize the units produced for each to maximize our revenue potential?

	A	B	C	D	E	F	G	H	I
1		Product A	Product B						
2	Sales price	25	30						
3	cotton needed per unit	4	2						
4	acrylic needed per unit	1	3						
5									
6	Units produced							Budget	
7	cotton consumed	0	0		Total units cotton used	0	<=	50	
8	acrylic consumed	0	0		Total units acrylic used	0	<=	30	
9	Revenue per product	0	0						
10									
11	Total revenue	0							
12									
13									
14									
15									

2. We will use **Solver** for this, so make sure you have it turned on! See the deck for instructions.
3. A "Solver parameters" pop-up box will appear. We need to set the parameters of this model:



a. First we are asked to set our *objective,* which is again to maximize revenue. Set this objective to B11.
b. Next we are asked to set the *variable cells.* These are the cells that can *change*, which are the units produced of Products A and B.
c. Finally we are asked to set the constraints. This will be to make sure that the units of cotton and acrylic used do not exceed our budgets. Leave the rest as-is.

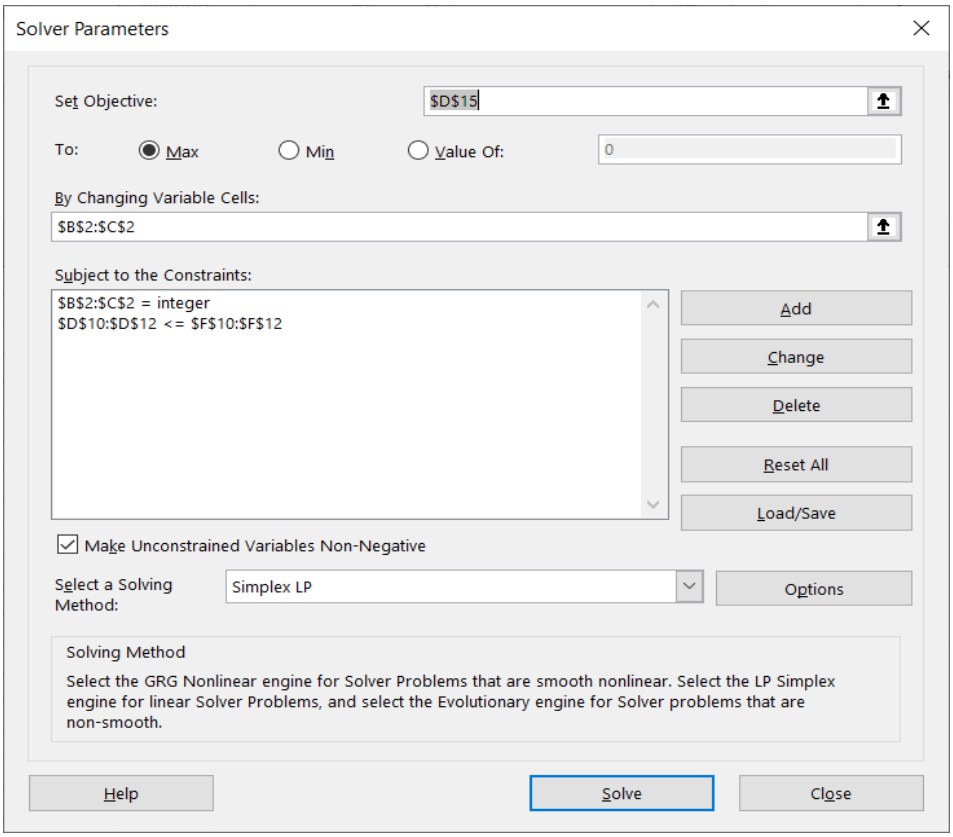


4. Solver converges to a result and it is populated to our worksheet. We use as much supply as available to maximize revenue.

	A	B	C	D	E	F	G	H	I
1		Product A	Product B						
2	Sales price	25	30						
3	cotton needed per unit	4	2						
4	acrylic needed per unit	1	3						
5									
6	Units produced	9	7	0				Budget	
7	cotton consumed	36	14		Total units cotton used	50	<=	50	
8	acrylic consumed	9	21		Total units acrylic used	30	<=	30	
9	Revenue per product	225	210						
10									
11	Total revenue	435							
12									
13									

Drill: `unit-production.xlsx`

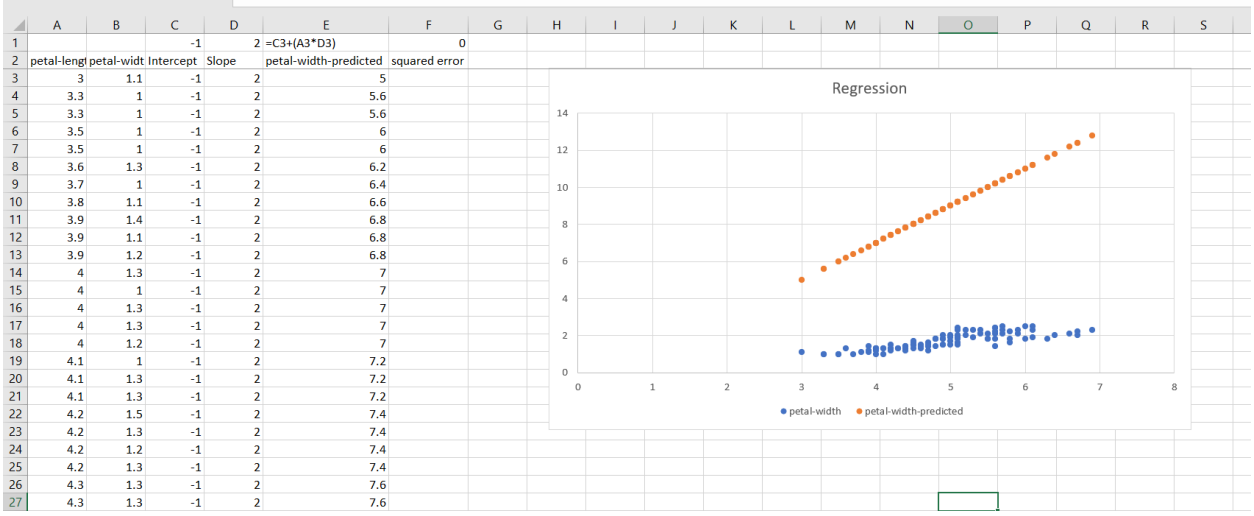
1. In this case, Solver may suggest you produce fractions of a unit. You can constrain variable to integers by setting the integer constraint.



Demo: `solver-regression.xlsx`

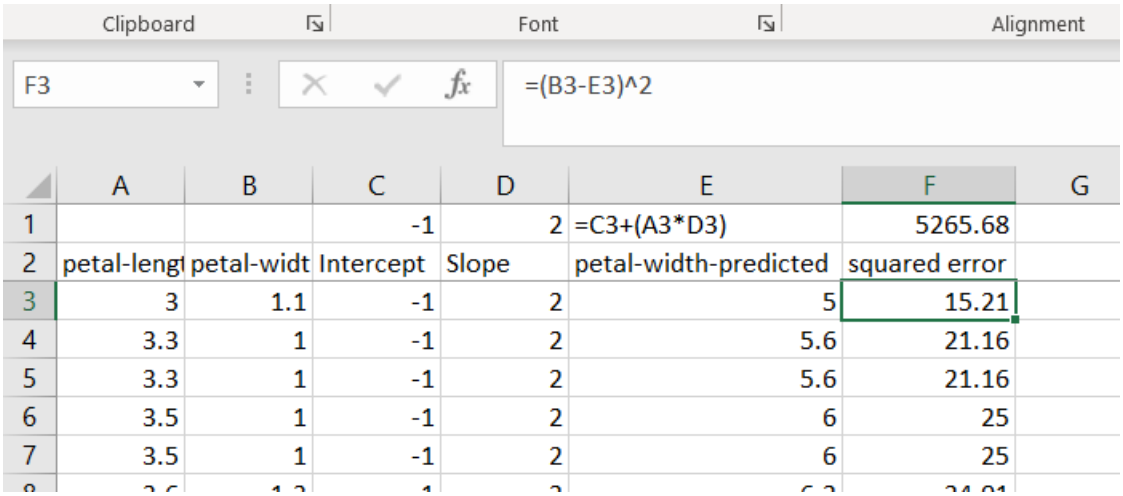
Linear regression utilizes optimization to find the best-fit line to pass through the scatterplot. Let's fit our own regression line using Solver.

1. Remember that our basic regression is Y = a + b*x. We will want to find the optimal intercept and slope of the line to predict each datapoint.



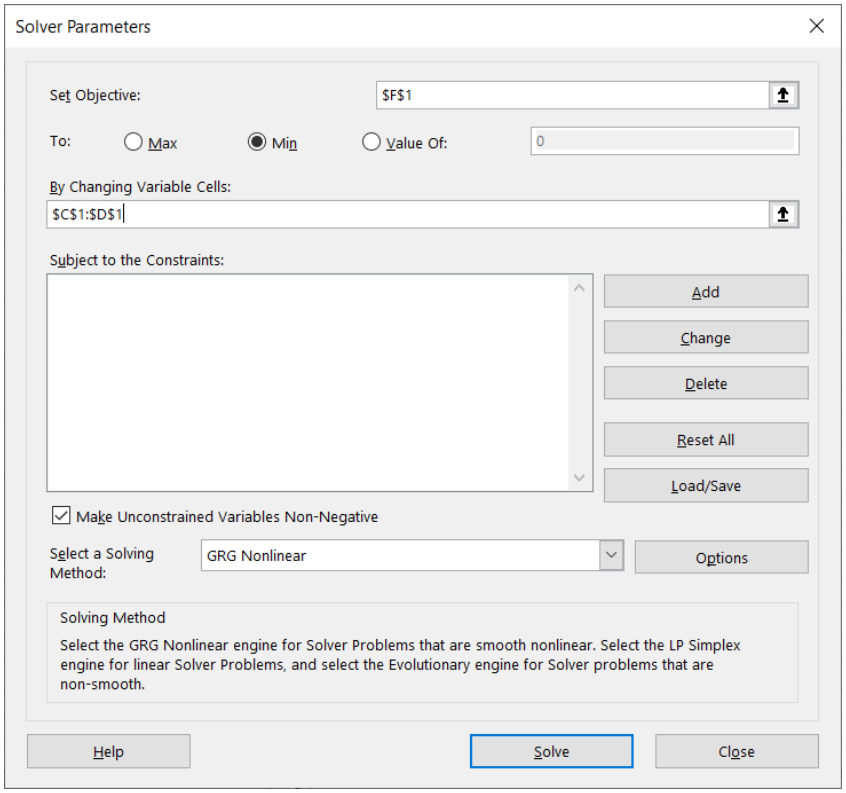
	A	B	C	D	E	F
1			-1	2	=C3+(A3*D3)	0
2	petal-leng	petal-widt	Intercept	Slope	petal-width-predicted	squared error
3	3	1.1	-1	2	5	
4	3.3	1	-1	2	5.6	
5	3.3	1	-1	2	5.6	
6	3.5	1	-1	2	6	
7	3.5	1	-1	2	6	
8	3.6	1.3	-1	2	6.2	
9	3.7	1	-1	2	6.4	
10	3.8	1.1	-1	2	6.6	
11	3.9	1.4	-1	2	6.8	
12	3.9	1.1	-1	2	6.8	
13	3.9	1.2	-1	2	6.8	
14	4	1.3	-1	2	7	
15	4	1	-1	2	7	
16	4	1.3	-1	2	7	
17	4	1.3	-1	2	7	
18	4	1.2	-1	2	7	
19	4.1	1	-1	2	7.2	
20	4.1	1.3	-1	2	7.2	
21	4.1	1.3	-1	2	7.2	
22	4.2	1.5	-1	2	7.4	
23	4.2	1.3	-1	2	7.4	
24	4.2	1.2	-1	2	7.4	
25	4.2	1.3	-1	2	7.4	
26	4.3	1.3	-1	2	7.6	
27	4.3	1.3	-1	2	7.6	

2. But how do we now it's optimized? Linear regression aims to minimize the *sum of squared errors*. We will take the difference between the actual and forecasted value and square it, then sum all these results.



	A	B	C	D	E	F	G
1			-1	2	=C3+(A3*D3)	5265.68	
2	petal-leng	petal-widt	Intercept	Slope	petal-width-predicted	squared error	
3	3	1.1	-1	2	5	15.21	
4	3.3	1	-1	2	5.6	21.16	
5	3.3	1	-1	2	5.6	21.16	
6	3.5	1	-1	2	6	25	
7	3.5	1	-1	2	6	25	

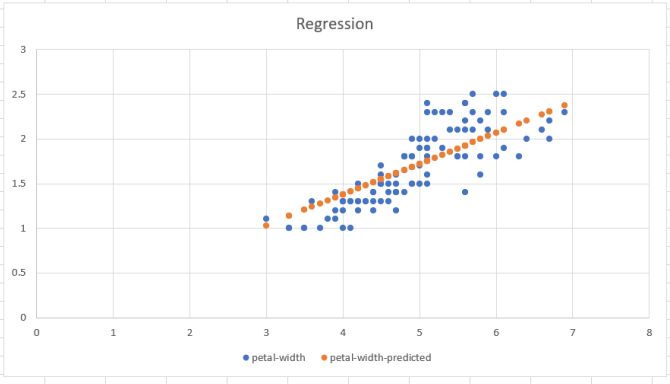
3. Now that we know our objective and variable cells, we can set up a Solver optimization to find the best values for the slope and intercept.



4. Solver has converged to a solution with an intercept of 0 and slope of .35. The fitted line is on our chart.

   a. Your results would be slightly different using the ToolPak or other regression methods. Solver has found a *sub-optimal solution* – a non-trivial problem in optimization methods!

	A	B	C	D	E	F
1			0	0.343859	=C3+(A3*D3)	6.194775498
2	petal-lengt	petal-widt	Intercept	Slope	petal-width-predicted	squared error
3	3	1.1	0	0.343859	1.031575842	0.004681865
4	3.3	1	0	0.343859	1.134733426	0.018153096
5	3.3	1	0	0.343859	1.134733426	0.018153096
6	3.5	1	0	0.343859	1.203505149	0.041414346
7	3.5	1	0	0.343859	1.203505149	0.041414346
8	3.6	1.3	0	0.343859	1.23789101	0.003857527
9	3.7	1	0	0.343859	1.272276872	0.074134695
10	3.8	1.1	0	0.343859	1.306662733	0.042709485
11	3.9	1.4	0	0.343859	1.341048594	0.003475268
12	3.9	1.1	0	0.343859	1.341048594	0.058104425
13	3.9	1.2	0	0.343859	1.341048594	0.019894706
14	4	1.3	0	0.343859	1.375434456	0.005690357
15	4	1	0	0.343859	1.375434456	0.140951031
16	4	1.3	0	0.343859	1.375434456	0.005690357
17	4	1.3	0	0.343859	1.375434456	0.005690357
18	4	1.2	0	0.343859	1.375434456	0.030777248
19	4.1	1	0	0.343859	1.409820317	0.167952692
20	4.1	1.3	0	0.343859	1.409820317	0.012060502
21	4.1	1.3	0	0.343859	1.409820317	0.012060502
22	4.2	1.5	0	0.343859	1.444206179	0.003112951
23	4.2	1.3	0	0.343859	1.444206179	0.020795422
24	4.2	1.2	0	0.343859	1.444206179	0.059636658
25	4.2	1.3	0	0.343859	1.444206179	0.020795422



Simulation

Demo: `inverse-distribution.xlsx`

1. Take a variable following a normal distribution with a mean of 50 and standard deviation of 10. We want to know what percentage of values in the distribution take on each of the numbers from 1 to 100, which we have listed in Column A.
    a. We can do this with the `NORM.DIST()` function. We will pass in the X value (from Column A), the mean and standard deviation, and whether this is a cumulative or probability mass function. We will choose `FALSE` for the latter.

	A	B	C	D	E
1	Mean	50			
2	Std. dev	10			
3					
4		=NORM.DIST(A6,$B$1,$B$2,FALSE)			
5	X	P|X			
6	1	2.43896E-07			
7	2	3.9613E-07			
8	3	6.36983E-07			
9	4	1.01409E-06			

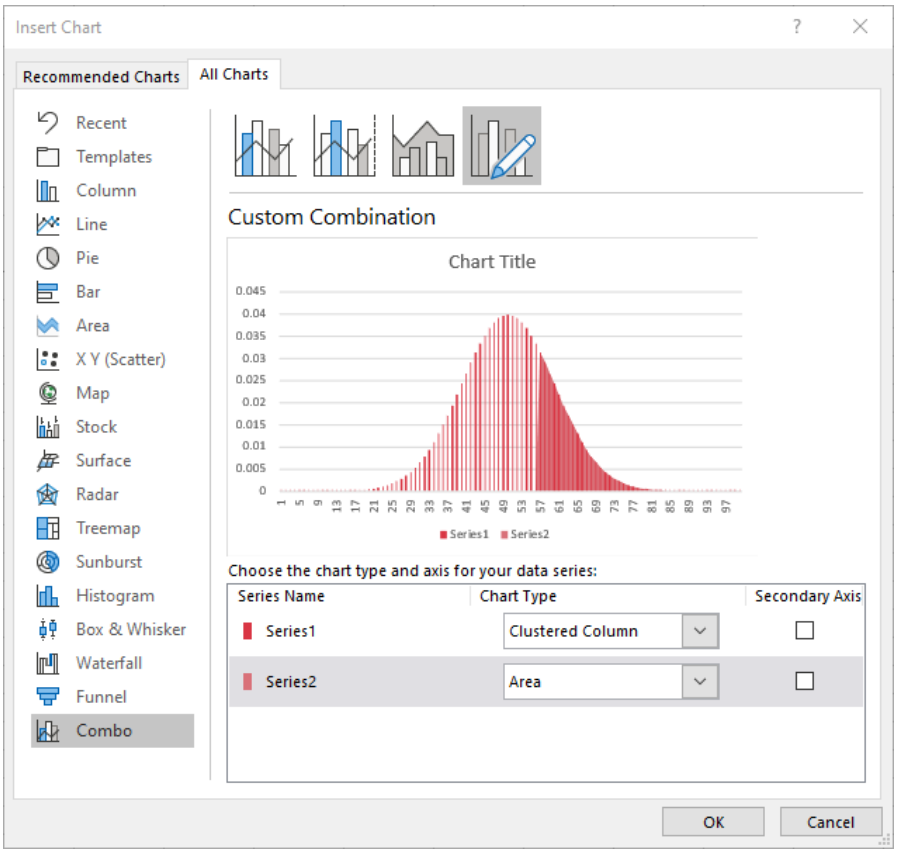
2. This will tell us what percentage likelihood a datapoint from the distribution would take on a value of 1, 2, and so on. Were you to scroll down on the table, you would see that there is approximately a 3.3% chance of a datapoint being equal to 44.
3. Let's plot the result
4. We will now pick an X value at random from the distribution, using the *inverse* of the normal distribution with `NORM.INV()`. This will go in cell C5.
    i. We will pick a random part of the normal distribution by passing `RAND()` to the first argument, followed by the mean and standard deviation.

	A	B	C	D
1	Mean	50		
2	Std. dev	10		
3				
4		=NORM.DIST(A6,$B$1,$B$2,FALSE)	=NORM.INV(RAND(),B1,B2)	
5	X	P|X	44.39302085	
6	1	2.43896E-07		
7	2	3.9613E-07		
8	3	6.36983E-07		
9	4	1.01409E-06		
10	5	1.59837E-06		

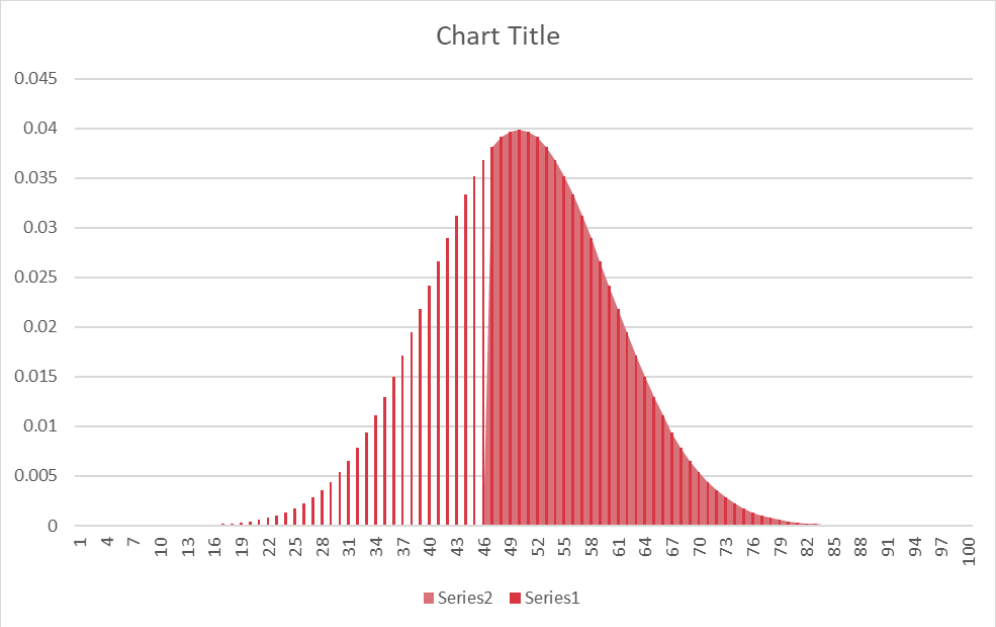
b. We will now write an `IF()` statement in column C. We will set the values in Column C equal to Column B if C5 is greater than `Column A`; otherwise 0.

	A	B	C	D	E
1	Mean	50			
2	Std. dev	10			
3					
4		=NORM.DIST(A6,$B$1,$B$2,FALSE)	=NORM.INV(RAND(),B1,B2)		
5	X	P|X	56.9884319		
6	1	2.43896E-07	0	=IF(A6>$C$5,B6,0)	
58	53	0.038138782	0		
59	54	0.036827014	0		
60	55	0.035206533	0		
61	56	0.03332246	0		
62	57	0.031225393	0.031225393		
63	58	0.028969155	0.028969155		
64	59	0.026608525	0.026608525		

5. We can now create a visualization using Columns B and C to understand how the inverse normal datapoint relates to our distribution.
   a. Insert a chart with the data from `B6:C105`.
   b. Go to Combo charts in the Insert Chart menu and set Series1 to a Clustered Column and Series2 to an Area.



6. We can now see that with the inverse normal, we are essentially picking a spot at random from our normal distribution. Select F9 to recalculate and pick a new place in the distribution.



Demo: `widget-sales.xlsx`

1. First, we want to simulate how much is sold on Day 1, Day 2, etc. We can use NORM.INV() plus RAND() to find the value at a random part of the distribution curve generated.

	A	B	C	D	E	F	G
1	Goal	$10,000					
2	Average sales/day	$1,000					
3	Standard deviation	$300					
4	Days to sell 100						
5							
6		=NORM.INV(RAND(),$B$2,$B$3)					
7	Day	Demand	Cumulative demand			Round	Days to sell
8	1	$640				1	
9	2					2	
10	3					3	
11	4					4	
12	5					5	

2. We can now find the running total of widgets sold by mixing our references for the SUM() function.

	A	B	C	D	E	F	G
1	Goal	$10,000					
2	Average sales/day	$1,000					
3	Standard deviation	$300					
4	Days to sell 100						
5							
6		=NORM.INV(RAND(),$B$2,$B$3)					
7	Day	Demand	Cumulative demand			Round	Days to sell
8	1	$490	$490			1	
9	2	$1,046	$1,536	=SUM($B$8:B9)		2	
10	3	$270	$1,805			3	
11	4	$752	$2,557			4	
12	5	$979	$3,537			5	
13	6	$756	$4,293			6	
14	7	$627	$4,920			7	
15	8	$871	$5,791			8	
16	9	$876	$6,667			9	

3. Now I want to find on what day we got to 100 widgets. I will use the XMATCH() function for this. This is a new Office 365 function. The way that our values are sorted, it's easier to use XMATCH() than MATCH().

	A	B	C	D	E	F	G	H
1	Goal	$10,000						
2	Average sales/day	$1,000						
3	Standard deviation	$300						
4	Days to sell 100	11	=XMATCH(B1,C8:C34,1,1)					
5								
6		=NORM.INV(RAND(),$B$2,$B$3)						
7	Day	Demand	Cumulative demand			Round	Days to sell	
8	1	$1,427	$1,427			1		
9	2	$1,121	$2,548	=SUM($B$8:B9)		2		
10	3	$957	$3,505			3		
11	4	$1,228	$4,733			4		
12	5	$1,362	$6,095			5		
13	6	$787	$6,881			6		

4. We've conducted this simulation once and saw it takes 11 days to sell $10,000 widgets. But what about doing this many, many times? We can use Data Tables to make this easier.
   a. Plug the result of B4 into G8.

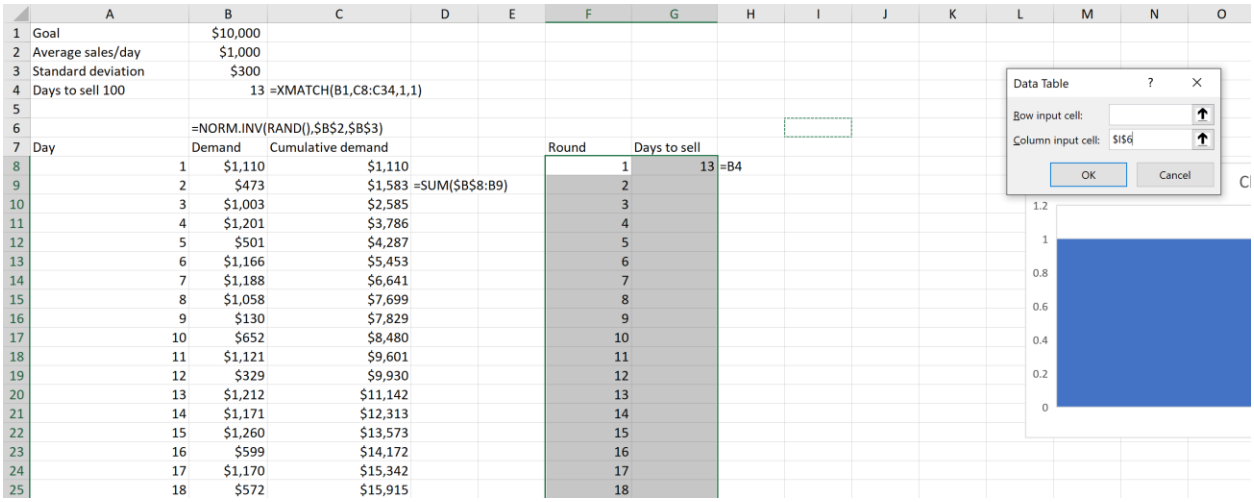
	A	B	C	D	E	F	G	H
1	Goal	$10,000						
2	Average sales/day	$1,000						
3	Standard deviation	$300						
4	Days to sell 100	13	=XMATCH(B1,C8:C34,1,1)					
5								
6		=NORM.INV(RAND(),$B$2,$B$3)						
7	Day	Demand	Cumulative demand			Round	Days to sell	
8	1	$1,110	$1,110			1	13	=B4
9	2	$473	$1,583	=SUM($B$8:B9)		2		
10	3	$1,003	$2,585			3		
11	4	$1,201	$3,786			4		
12	5	$501	$4,287			5		
13	6	$1,166	$5,453			6		
14	7	$1,188	$6,641			7		

b. We now want to fill in the rest of this simulation table and can "trick" Data Tables into doing it for us. With the range F8:G307 highlighted, head to Data > What-If Analysis > Data Table.

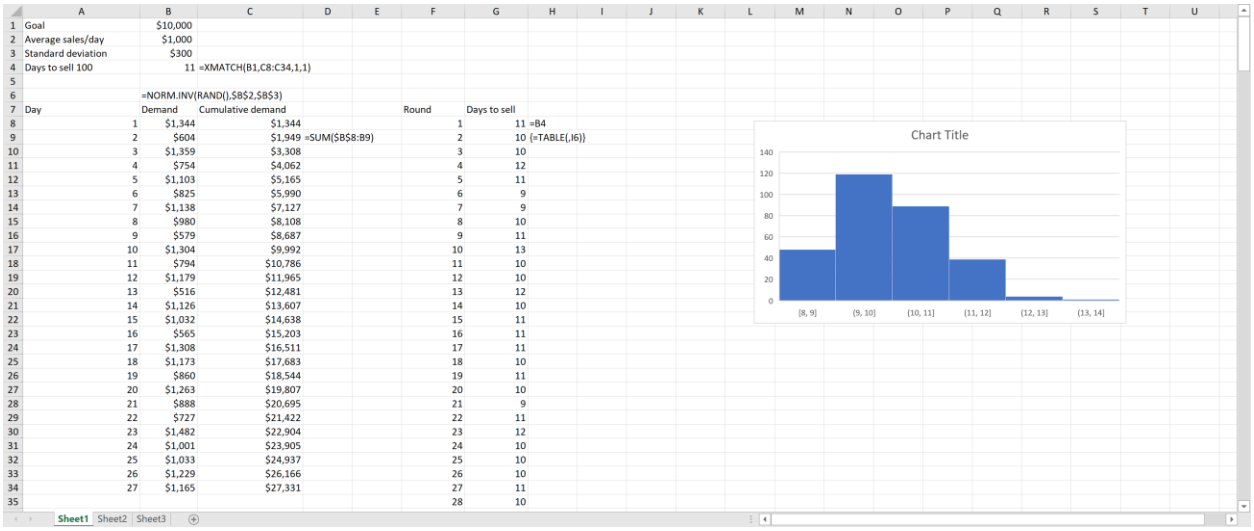
c. **This part is quite confusing!**

   i. You will leave the Row input cell *blank.*

   ii. You will set the Column input cell to a *random blank cell.* This will trick Excel into populating the table with the results of G8.



d. Our results of 300 simulations is completed. Now you can get a more balanced understanding of how many days it will take to sell 100 widgets.



	A	B	C	D	E	F	G	H
1	Goal	$10,000						
2	Average sales/day	$1,000						
3	Standard deviation	$300						
4	Days to sell 100	11	=XMATCH(B1,C8:C34,1,1)					
5								
6		=NORM.INV(RAND(),$B$2,$B$3)						
7	Day	Demand	Cumulative demand			Round	Days to sell	
8	1	$1,344	$1,344			1	11	=B4
9	2	$604	$1,949	=SUM($B$8:B9)		2	10	{=TABLE(,I6)}
10	3	$1,359	$3,308			3	10	
11	4	$754	$4,062			4	12	
12	5	$1,103	$5,165			5	11	
13	6	$825	$5,990			6	9	
14	7	$1,138	$7,127			7	9	
15	8	$980	$8,108			8	10	
16	9	$579	$8,687			9	11	
17	10	$1,304	$9,992			10	13	
18	11	$794	$10,786			11	10	
19	12	$1,179	$11,965			12	10	
20	13	$516	$12,481			13	12	
21	14	$1,126	$13,607			14	10	
22	15	$1,032	$14,638			15	11	
23	16	$565	$15,203			16	11	
24	17	$1,308	$16,511			17	11	
25	18	$1,173	$17,683			18	10	
26	19	$860	$18,544			19	11	
27	20	$1,263	$19,807			20	10	
28	21	$888	$20,695			21	9	
29	22	$727	$21,422			22	11	
30	23	$1,482	$22,904			23	12	
31	24	$1,001	$23,905			24	10	
32	25	$1,033	$24,937			25	10	
33	26	$1,229	$26,166			26	10	
34	27	$1,165	$27,331			27	11	
35						28	10	

Drill: `monte-carlo.xlsx`

1. This file is so-called because these are called *Monte Carlo simulations.*
   a. Again, make sure to leave the *column input cell blank* in the data table. That and getting the NORM.INV() functions down make this one take practice!