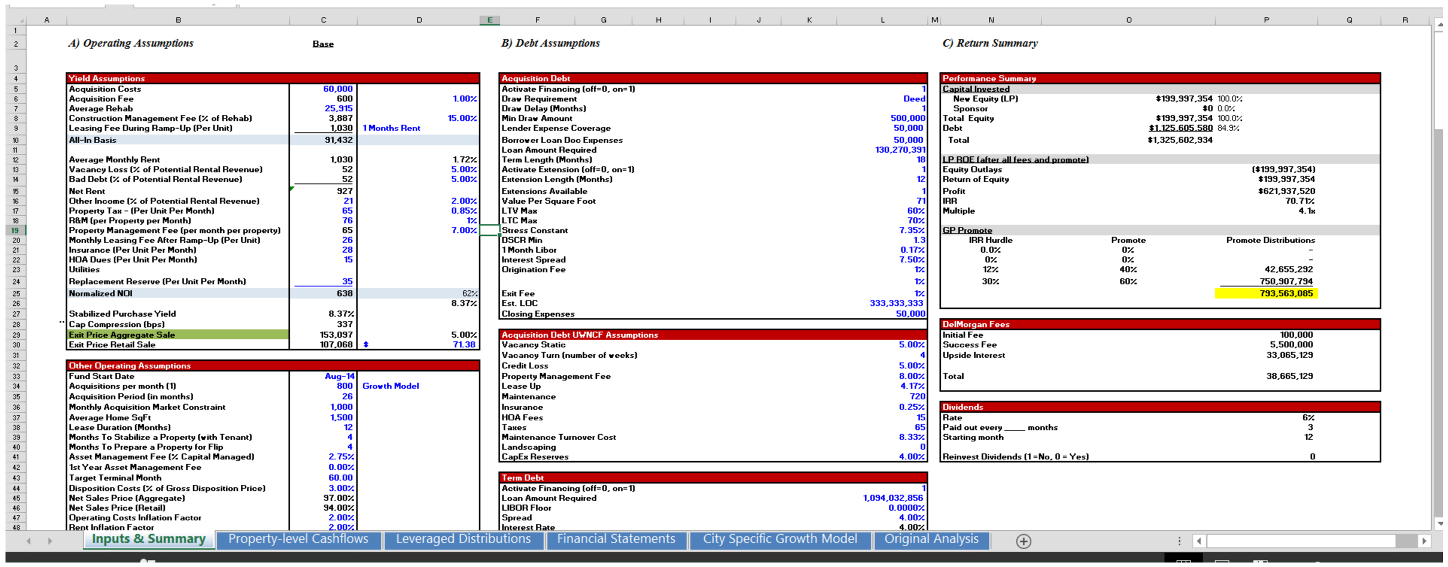
Implementation Documentation:

Jarett Brock

- **This section explains exactly what business problem had to be solved.**

To fully understand what my macros accomplish, it is necessary for me to explain the model around which it is based. I originally sat down with the Senior Financial Analyst for 3 hours while he explained this model to me, so I will do my best to make my explanation succinct. This model is a real-estate investment model that took the finance team more than 6 months to perfect. *It takes into account several hundred factors when determining the investment's ultimate IRR (Internal Rate of Return- a profitability measure) and the amount of money Key Property Investing will make in an investment* —i.e. inflation, average rent, financing through highly levered transactions, tiered system of debt structures, lagging repayments of mortgages, acquisitions through non-performing loans. Here is a snapshot of some of the inputs on the first sheet that are vital in the model.



A) Operating Assumptions — **Base**

Yield Assumptions		
Acquisition Costs	60,000	
Acquisition Fee	600	1.00%
Average Rehab	25,915	
Construction Management Fee (% of Rehab)	3,887	15.00%
Leasing Fee During Ramp-Up (Per Unit)	1,030	1 Months Rent
All-In Basis	91,432	
Average Monthly Rent	1,030	1.72%
Vacancy Loss (% of Potential Rental Revenue)	52	5.00%
Bad Debt (% of Potential Rental Revenue)	52	5.00%
Net Rent	927	
Other Income (% of Potential Rental Revenue)	21	2.00%
Property Tax - (Per Unit Per Month)	65	0.85%
R&M (per Property per Month)	76	1%
Property Management Fee (per month per property)	65	7.00%
Monthly Leasing Fee After Ramp-Up (Per Unit)	26	
Insurance (Per Unit Per Month)	28	
HOA Dues (Per Unit Per Month)	15	
Utilities		
Replacement Reserve (Per Unit Per Month)	35	
Normalized NOI	638	62%
		8.37%
Stabilized Purchase Yield	8.37%	
Cap Compression (bps)	337	
Exit Price Aggregate Sale	153,097	5.00%
Exit Price Retail Sale	107,068	$ 71.38

Other Operating Assumptions		
Fund Start Date	Aug-14	
Acquisitions per month (1)	800	Growth Model
Acquisition Period (in months)	26	
Monthly Acquisition Market Constraint	1,000	
Average Home SqFt	1,500	
Lease Duration (Months)	12	
Months To Stabilize a Property (with Tenant)	4	
Months To Prepare a Property for Flip	4	
Asset Management Fee (% Capital Managed)	2.75%	
1st Year Asset Management Fee	0.00%	
Target Terminal Month	60.00	
Disposition Costs (% of Gross Disposition Price)	3.00%	
Net Sales Price (Aggregate)	97.00%	
Net Sales Price (Retail)	94.00%	
Operating Costs Inflation Factor	2.00%	
Rent Inflation Factor	2.00%	

B) Debt Assumptions

Acquisition Debt	
Activate Financing (off=0, on=1)	1
Draw Requirement	Deed
Draw Delay (Months)	1
Min Draw Amount	500,000
Lender Expense Coverage	50,000
Borrower Loan Doc Expenses	50,000
Loan Amount Required	130,270,391
Term Length (Months)	18
Activate Extension (off=0, on=1)	1
Extension Length (Months)	12
Extensions Available	1
Value Per Square Foot	71
LTV Max	60%
LTC Max	70%
Stress Constant	7.35%
DSCR Min	1.3
1 Month Libor	0.17%
Interest Spread	7.50%
Origination Fee	1%
	1%
Exit Fee	1%
Est. LOC	333,333,333
Closing Expenses	50,000

Acquisition Debt UWNCF Assumptions	
Vacancy Static	5.00%
Vacancy Turn (number of weeks)	4
Credit Loss	5.00%
Property Management Fee	8.00%
Lease Up	4.17%
Maintenance	720
Insurance	0.25%
HOA Fees	15
Taxes	65
Maintenance Turnover Cost	8.33%
Landscaping	0
CapEx Reserves	4.00%

Term Debt	
Activate Financing (off=0, on=1)	1
Loan Amount Required	1,094,032,856
LIBOR Floor	0.0000%
Spread	4.00%
Interest Rate	4.00%

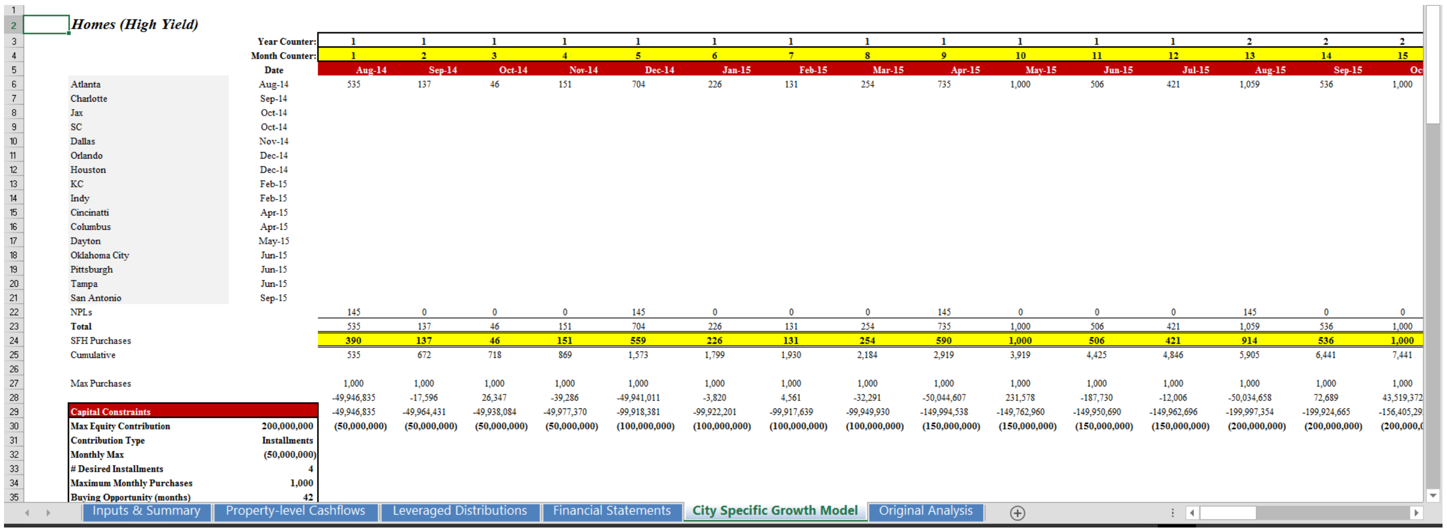
C) Return Summary

Performance Summary			
Capital Invested			
New Equity (LP)		$199,997,354	100.0%
Sponsor		$0	0.0%
Total Equity		$199,997,354	100.0%
Debt		$1,125,605,580	84.9%
Total		$1,325,602,934	
LP ROE (after all fees and promote)			
Equity Outlays			($199,997,354)
Return of Equity			$199,997,354
Profit			$621,937,520
IRR			70.71%
Multiple			4.1x
GP Promote			
IRR Hurdle	Promote		Promote Distributions
0.0%	0%		-
0%	0%		-
12%	40%		42,655,292
30%	60%		750,907,794
			793,563,085

DelMorgan Fees	
Initial Fee	100,000
Success Fee	5,500,000
Upside Interest	33,065,129
Total	38,665,129

Dividends	
Rate	6%
Paid out every _____ months	3
Starting month	12
Reinvest Dividends (1=No, 0=Yes)	0

These inputs are then used countlessly throughout the other worksheets to, with tons of iterative calculations, ultimately calculate the total IRR. All of these numbers and equations, however, are all predicted off of the number of homes bought each month. In the following screenshot, you can see the worksheet where the property acquisitions are registered (they are not determined by formulas within the sheet, they are manually generated or created through macros).



Homes (High Yield)

	Year Counter:	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2
	Month Counter:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Date	Aug-14	Sep-14	Oct-14	Nov-14	Dec-14	Jan-15	Feb-15	Mar-15	Apr-15	May-15	Jun-15	Jul-15	Aug-15	Sep-15	Oc
Atlanta	Aug-14	535	137	46	151	704	226	131	254	735	1,000	506	421	1,059	536	1,000
Charlotte	Sep-14															
Jax	Oct-14															
SC	Oct-14															
Dallas	Nov-14															
Orlando	Dec-14															
Houston	Dec-14															
KC	Feb-15															
Indy	Feb-15															
Cincinatti	Apr-15															
Columbus	Apr-15															
Dayton	May-15															
Oklahoma City	Jun-15															
Pittsburgh	Jun-15															
Tampa	Jun-15															
San Antonio	Sep-15															
NPLs		145	0	0	0	145	0	0	0	145	0	0	0	145	0	0
Total		535	137	46	151	704	226	131	254	735	1,000	506	421	1,059	536	1,000
SFH Purchases		**390**	**137**	**46**	**151**	**559**	**226**	**131**	**254**	**590**	**1,000**	**506**	**421**	**914**	**536**	**1,000**
Cumulative		535	672	718	869	1,573	1,799	1,930	2,184	2,919	3,919	4,425	4,846	5,905	6,441	7,441
Max Purchases		1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
		-49,946,835	-17,596	26,347	-39,286	-49,941,011	-3,820	4,561	-32,291	-50,044,607	231,578	-187,730	-12,006	-50,034,658	72,689	43,519,372
		-49,946,835	-49,964,431	-49,938,084	-49,977,370	-99,918,381	-99,922,201	-99,917,639	-99,949,930	-149,994,538	-149,762,960	-149,950,690	-149,962,696	-199,997,354	-199,924,665	-156,405,29
		(50,000,000)	**(50,000,000)**	**(50,000,000)**	**(50,000,000)**	**(100,000,000)**	**(100,000,000)**	**(100,000,000)**	**(100,000,000)**	**(150,000,000)**	**(150,000,000)**	**(150,000,000)**	**(150,000,000)**	**(200,000,000)**	**(200,000,000)**	**(200,000,0**

Capital Constraints	
Max Equity Contribution	**200,000,000**
Contribution Type	**Installments**
Monthly Max	**(50,000,000)**
# Desired Installments	**4**
Maximum Monthly Purchases	**1,000**
Buying Opportunity (months)	**42**

They previously had a simple macro that started at the first month, and from zero added one house at a time until they exceeded their capital restraints or available cash for purchasing homes that month, and then they would increment the loop by one cell to the right (which is the next month) and do it again. The problem with this was that every time a house was added, it would take nearly 1.5 seconds for the sheet to update fully. As you can see in the worksheet above, there were several hundred homes purchased every month. Therefore, it would take their macro an average of 2 hours to do 120 months of property optimization—adding a house one by one by one.

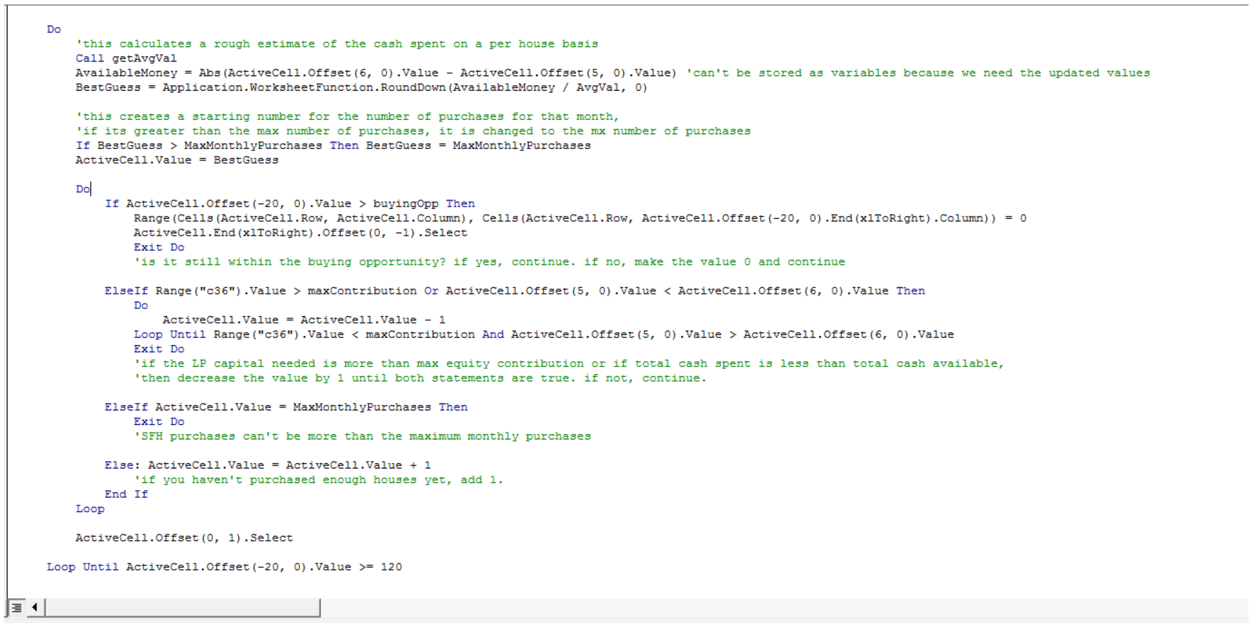
- How did I fix the problem? (All of the following is found in the "Optimization" module)

The first thing I did was erase their macro and start from scratch. After a bit of trial and error, I came up with a method that I thought would be successful at predicting the total number of homes purchased for each month. I created a sub procedure called getAvgVal that does a very simple task. First it makes sure the number of acquisitions starts at 0 and records the value of available cash for that month. Then, it adds one house to the spreadsheet and records the new value of available cash—when you purchase a home with layered financing, it doesn't simply lower it by the average value of a home. *The difference between the two recorded variables is a simple estimate of each property's impact on the capital constraints and available cash* for spending for that particular month—that is the ultimate purpose of that sub procedure.

Then, by taking the remaining available cash for that month and dividing it by the average property's impact on available cash (as calculated through getAvgVal), I generated a number that was within 1 or 2 of the true amount. If my estimate of homes for that particular month created a scenario that exceeded our capital restraints, the macro subtracts one house at a time until it passes the restraints—this was achieved by looping IF statements. If my estimate of homes for that particular month was less than what our restraints allowed, than it would add one house until it exceeds the limits—when it exceeds the limit, it uses the previous value and exits the loop. There is also a maximum number of homes that we believe we could purchase in a month, so we never pass that value, even if money permits (that value is stored as a variable at the beginning).

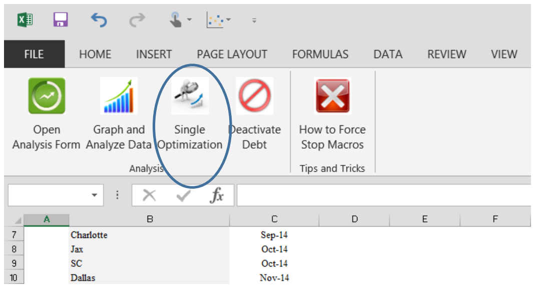
After a month is optimized, the loop is incremented and it repeats the process for the next month until the number of months is equal to 120, because 10 years is our maximum investment horizon. This whole process occurs in the sub procedure call "Optimize".

Honestly, I think the code is a little easier to follow than a written explanation:



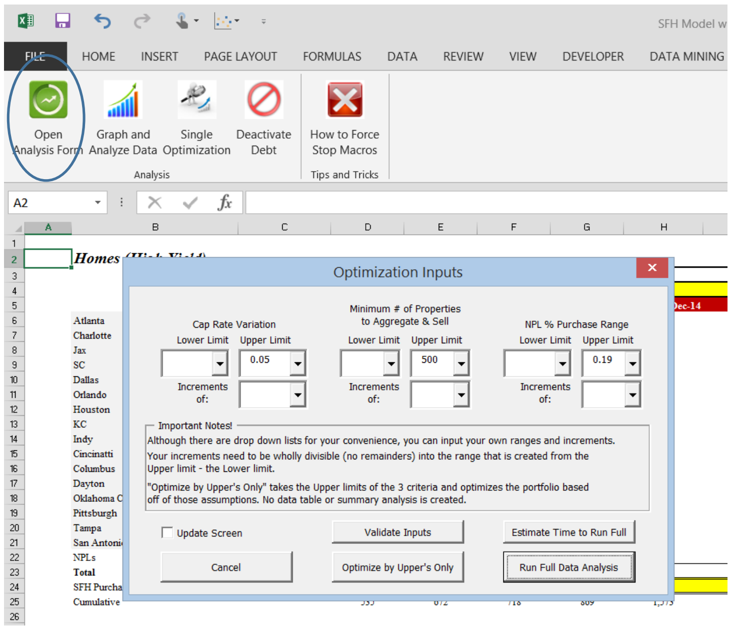
```
Do
    'this calculates a rough estimate of the cash spent on a per house basis
    Call getAvgVal
    AvailableMoney = Abs(ActiveCell.Offset(6, 0).Value - ActiveCell.Offset(5, 0).Value) 'can't be stored as variables because we need the updated values
    BestGuess = Application.WorksheetFunction.RoundDown(AvailableMoney / AvgVal, 0)

    'this creates a starting number for the number of purchases for that month,
    'if its greater than the max number of purchases, it is changed to the mx number of purchases
    If BestGuess > MaxMonthlyPurchases Then BestGuess = MaxMonthlyPurchases
    ActiveCell.Value = BestGuess

    Do
        If ActiveCell.Offset(-20, 0).Value > buyingOpp Then
            Range(Cells(ActiveCell.Row, ActiveCell.Column), Cells(ActiveCell.Row, ActiveCell.Offset(-20, 0).End(xlToRight).Column)) = 0
            ActiveCell.End(xlToRight).Offset(0, -1).Select
            Exit Do
            'is it still within the buying opportunity? if yes, continue. if no, make the value 0 and continue

        ElseIf Range("c36").Value > maxContribution Or ActiveCell.Offset(5, 0).Value < ActiveCell.Offset(6, 0).Value Then
            Do
                ActiveCell.Value = ActiveCell.Value - 1
            Loop Until Range("c36").Value < maxContribution And ActiveCell.Offset(5, 0).Value > ActiveCell.Offset(6, 0).Value
            Exit Do
            'if the LP capital needed is more than max equity contribution or if total cash spent is less than total cash available,
            'then decrease the value by 1 until both statements are true. if not, continue.

        ElseIf ActiveCell.Value = MaxMonthlyPurchases Then
            Exit Do
            'SFH purchases can't be more than the maximum monthly purchases

        Else: ActiveCell.Value = ActiveCell.Value + 1
            'if you haven't purchased enough houses yet, add 1.
        End If
    Loop

    ActiveCell.Offset(0, 1).Select

Loop Until ActiveCell.Offset(-20, 0).Value >= 120
```

By using the timer function, I record how long it takes to run each optimization so that I could quantify to my previous bosses how much more efficient my macro was than theirs. **When previously it took 2 hours to run a single optimization, it now takes 54 seconds on average to optimize a portfolio.** This optimization can also be called through clicking on the “Single Optimization” button in the Analysis ribbon that I added.



- **How did I further use my new optimization macro?**

One of the main reasons Key Property Investing wanted to decrease the amount of time it took to optimize a scenario was because investors want to see many variations of a portfolio. They want to know best case scenario, worst case scenario, and everything in between. The picture below shows what the user form looks like; I will then explain what it does.



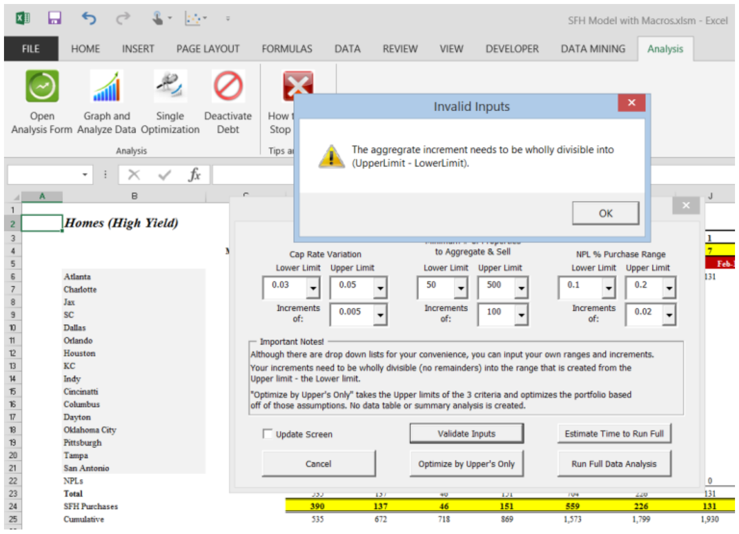
Important Notes!
Although there are drop down lists for your convenience, you can input your own ranges and increments.
Your increments need to be wholly divisible (no remainders) into the range that is created from the Upper limit - the Lower limit.
"Optimize by Upper's Only" takes the Upper limits of the 3 criteria and optimizes the portfolio based off of those assumptions. No data table or summary analysis is created.

To facilitate those needs, I created a user form that performs 5 different functions.

FULL ANALYSIS

1) Run Full Data Analysis – this is where I spent the majority of my time for this project. The three variables shown in the User Form—Cap Values, Minimum Numbers to Aggregate and Sell, and NPL % Purchase Range—are the 3 most common inputs that private equity funds like to tinker with when determining the possible variations of the portfolio performance. When the button is clicked and you confirm that you do want to proceed, the first step is to validate that the ranges meet the criteria. This is done by calling the validateFullInputs macro making sure that it passes the criteria. These criteria are mentioned in the explanation of the “Validate Inputs” button. If the inputs are validated, the

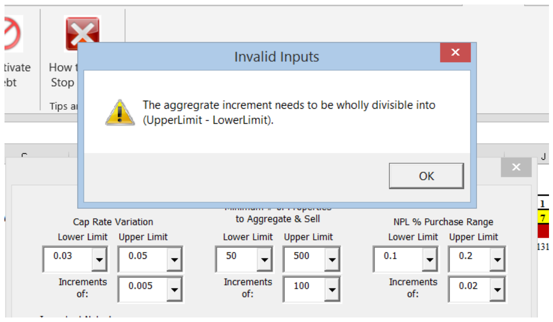
validateFullInputs macro sets the global variable “Validator”. The called procedure ends, sending us back into the main macro where the following statement is run: If Validator = 1 Then Exit Sub. This allows for the sub to end without having to use Unload Me or End, thus leaving the User Form open—which was my intention. Once all necessary validation is passed, the Full Analysis begins.



Invalid Inputs

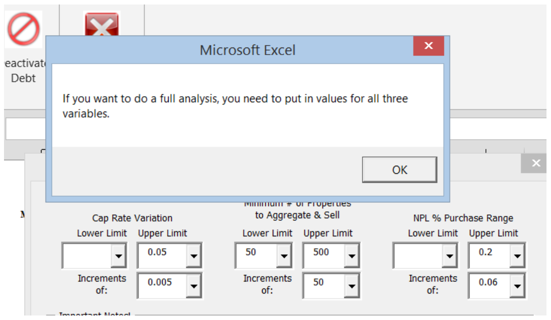
The aggregrate increment needs to be wholly divisible into (UpperLimit - LowerLimit).

Important Notes!
Although there are drop down lists for your convenience, you can input your own ranges and increments. Your increments need to be wholly divisible (no remainders) into the range that is created from the Upper limit - the Lower limit.
"Optimize by Upper's Only" takes the Upper limits of the 3 criteria and optimizes the portfolio based off of those assumptions. No data table or summary analysis is created.



Invalid Inputs

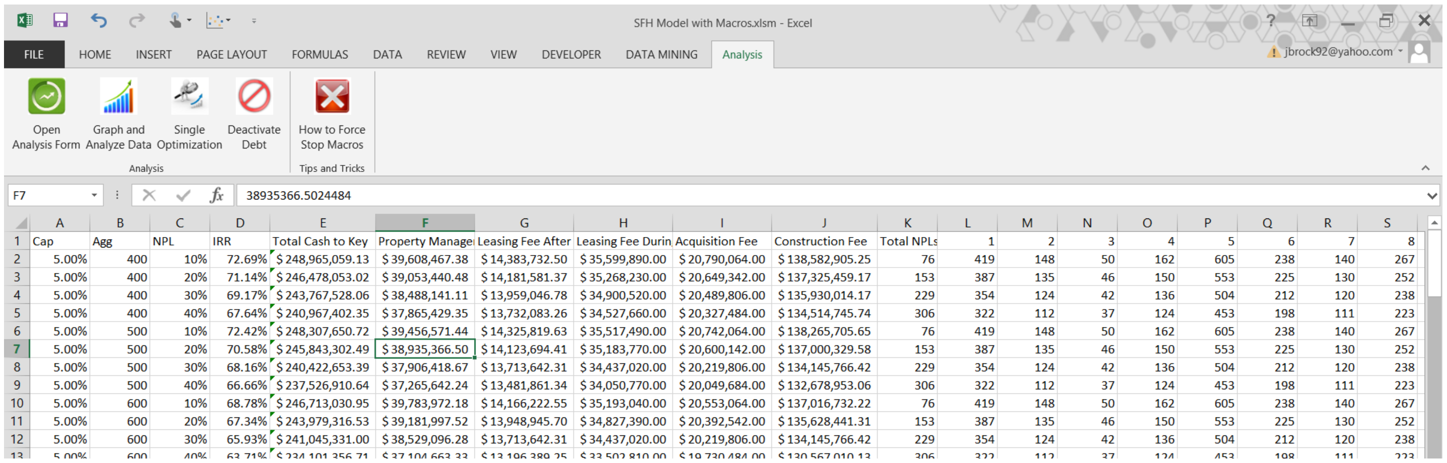
The aggregrate increment needs to be wholly divisible into (UpperLimit - LowerLimit).



Microsoft Excel

If you want to do a full analysis, you need to put in values for all three variables.

The full analysis opens a new tab and names the headers for each column. The code then goes through every possible permutation of the 3 variables within their ranges, optimizes the portfolio at each unique scenario, and then copies down the important information (what the company decided was important) into the newly created sheet. After every new scenario, the new data is added to the bottom of the sheet, creating a well-arranged data set. The criteria of each scenario is listed in the first 3 columns, and then the other information follows.



F7 | 38935366.5024484

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Cap	Agg	NPL	IRR	Total Cash to Key	Property Manage	Leasing Fee After	Leasing Fee Durin	Acquisition Fee	Construction Fee	Total NPLs	1	2	3	4	5	6	7	8
2	5.00%	400	10%	72.69%	$ 248,965,059.13	$ 39,608,467.38	$ 14,383,732.50	$ 35,599,890.00	$ 20,790,064.00	$ 138,582,905.25	76	419	148	50	162	605	238	140	267
3	5.00%	400	20%	71.14%	$ 246,478,053.02	$ 39,053,440.48	$ 14,181,581.37	$ 35,268,230.00	$ 20,649,342.00	$ 137,325,459.17	153	387	135	46	150	553	225	130	252
4	5.00%	400	30%	69.17%	$ 243,767,528.06	$ 38,488,141.11	$ 13,959,046.78	$ 34,900,520.00	$ 20,489,806.00	$ 135,930,014.17	229	354	124	42	136	504	212	120	238
5	5.00%	400	40%	67.64%	$ 240,967,402.35	$ 37,865,429.35	$ 13,732,083.26	$ 34,527,660.00	$ 20,327,484.00	$ 134,514,745.74	306	322	112	37	124	453	198	111	223
6	5.00%	500	10%	72.42%	$ 248,307,650.72	$ 39,456,571.44	$ 14,325,819.63	$ 35,517,490.00	$ 20,742,064.00	$ 138,265,705.65	76	419	148	50	162	605	238	140	267
7	5.00%	500	20%	70.58%	$ 245,843,302.49	$ 38,935,366.50	$ 14,123,694.41	$ 35,183,770.00	$ 20,600,142.00	$ 137,000,329.58	153	387	135	46	150	553	225	130	252
8	5.00%	500	30%	68.16%	$ 240,422,653.39	$ 37,906,418.67	$ 13,713,642.31	$ 34,437,020.00	$ 20,219,806.00	$ 134,145,766.42	229	354	124	42	136	504	212	120	238
9	5.00%	500	40%	66.66%	$ 237,526,910.64	$ 37,265,642.24	$ 13,481,861.34	$ 34,050,770.00	$ 20,049,684.00	$ 132,678,953.06	306	322	112	37	124	453	198	111	223
10	5.00%	600	10%	68.78%	$ 246,713,030.95	$ 39,783,972.18	$ 14,166,222.55	$ 35,193,040.00	$ 20,553,064.00	$ 137,016,732.22	76	419	148	50	162	605	238	140	267
11	5.00%	600	20%	67.34%	$ 243,979,316.53	$ 39,181,997.52	$ 13,948,945.70	$ 34,827,390.00	$ 20,392,542.00	$ 135,628,441.31	153	387	135	46	150	553	225	130	252
12	5.00%	600	30%	65.93%	$ 241,045,331.00	$ 38,529,096.28	$ 13,713,642.31	$ 34,437,020.00	$ 20,219,806.00	$ 134,145,766.42	229	354	124	42	136	504	212	120	238
13	5.00%	600	40%	63.71%	$ 234,101,356.71	$ 37,104,663.33	$ 13,196,389.25	$ 33,502,810.00	$ 19,730,484.00	$ 130,567,010.13	306	322	112	37	124	453	198	111	223

2) Optimize by Upper’s Only – this performs almost the exact same task as the ribbon button “Single Optimization”. The only difference is that upon clicking the button, it validates the upper limit values of each variable, saves them into the model and runs the optimization. A warning box appears to make sure that you want to run the optimization.

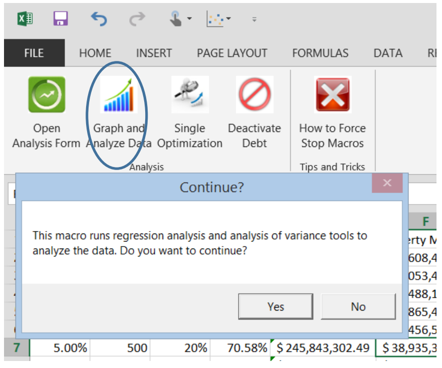
3) Update Screen – This is a checkbox that has is implemented within "Optimize by Uppers Only" and "Run Full Data Analysis". If checked, screen updating will be activated during the macros. If left unchecked, screen updating will be turned off.

4) Validate Inputs- this runs the macro to confirm whether or not the inputs are capable of being analyzed with the "Run Full Data Analysis" criteria. These criteria include not being above or below arbitrary limits that the company decided—the limits are in the code if you're interested in seeing what they are. But, the important part is the following: the chosen/or manually entered incrementation has to be wholly divisible into the difference of the Upper Limit minus the Lower limit. This is because I used For loops in the "Run Full Data Analysis" button that need the increments to fit into the difference of the Upper and Lower limits a perfect number of times. Also, another criteria for the full analysis is that all combo boxes must have a lower and upper limit. If any of these criteria are not fulfilled, a message box explaining the exact reason why you can't proceed appears and explains how to fix it.

5) Estimate to Time to Run Full – After running 300 unique optimization scenarios, a calculated an average amount of time it takes to optimize one scenario. Then, by taking that two times the average (it takes twice as long with everything else that is going on in the macro) and multiplying it by the total number of permutations found within the ranges of the 3 variables. It reports that estimated time in a message box.

- **Graph and Analyze Data Button in the Ribbon**

When the workbook is opened, the code makes sure the Analysis Tool and Analsysis Tool – VBA are both installed, if they aren't, it will install them. Once a full analysis has been generated, the Graph and Analyze Data button will take the data from the last sheet and run two separate multivariate analyses on it with the Analysis Tool in excel. It puts them into two different tabs, naming them according to the nature of each regression. One analyzes the three independent variables (Capp, Agg, and NPL) to predict IRR, and one uses the same independent variable to predict the total cash flow to Key.

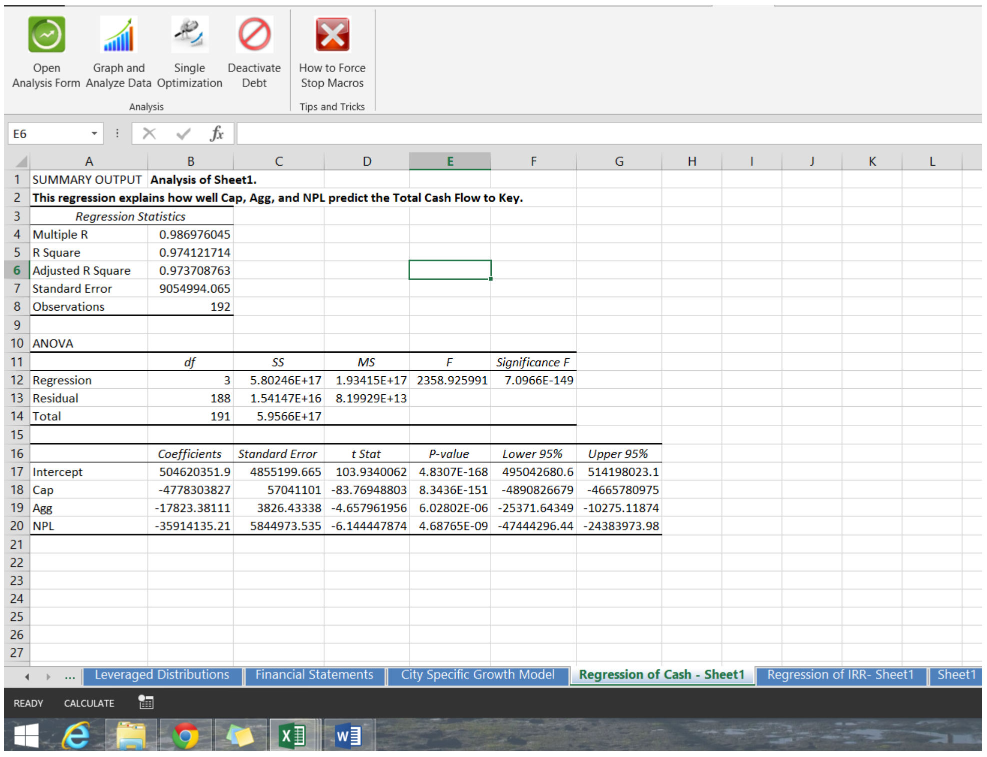


Continue?

This macro runs regression analysis and analysis of variance tools to analyze the data. Do you want to continue?

Yes No

7	5.00%	500	20%	70.58%	$ 245,843,302.49	$ 38,935,3

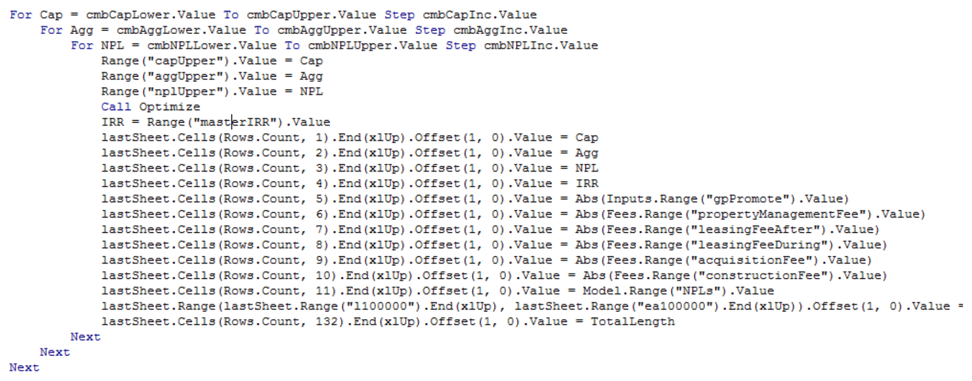


	A	B	C	D	E	F	G
1	SUMMARY OUTPUT	**Analysis of Sheet1.**					
2	**This regression explains how well Cap, Agg, and NPL predict the Total Cash Flow to Key.**						
3	*Regression Statistics*						
4	Multiple R	0.986976045					
5	R Square	0.974121714					
6	Adjusted R Square	0.973708763					
7	Standard Error	9054994.065					
8	Observations	192					
9							
10	ANOVA						
11		*df*	*SS*	*MS*	*F*	*Significance F*	
12	Regression	3	5.80246E+17	1.93415E+17	2358.925991	7.0966E-149	
13	Residual	188	1.54147E+16	8.19929E+13			
14	Total	191	5.9566E+17				
15							
16		*Coefficients*	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*
17	Intercept	504620351.9	4855199.665	103.9340062	4.8307E-168	495042680.6	514198023.1
18	Cap	-4778303827	57041101	-83.76948803	8.3436E-151	-4890826679	-4665780975
19	Agg	-17823.38111	3826.43338	-4.657961956	6.02802E-06	-25371.64349	-10275.11874
20	NPL	-35914135.21	5844973.535	-6.144447874	4.68765E-09	-47444296.44	-24383973.98

The next sheet, Regression of IRR – Sheet1, provides the same format. The only different in the explanation in A2 and the dependent variable that was used to run the regression analysis.

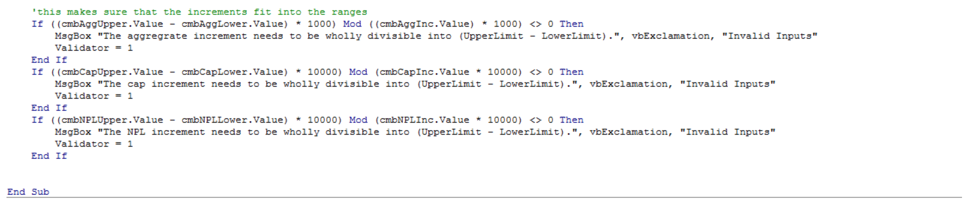
What I learned.

I learned more during this project than I have through the whole semester. I ran into nurmerous problems that really required me to think how I could solve the problem most efficiently. My first real problem was trying to find out how I could write code that would go through all the possible permutations of the ranges defined in the user form. I thought a lot about using 3 and 4 dimensional arrays, but I could never see the end product coming out the way I wanted when doing that. I ended up finding the layered For lops did exactly what I wanted to do. It probably took me 4 to 5 hours of just looking at my computer screen, trying something, erasing it, and trying again before I settled on the For loop method—but, I am proud of where it arrived. The code looks like this:



```
For Cap = cmbCapLower.Value To cmbCapUpper.Value Step cmbCapInc.Value
    For Agg = cmbAggLower.Value To cmbAggUpper.Value Step cmbAggInc.Value
        For NPL = cmbNPLLower.Value To cmbNPLUpper.Value Step cmbNPLInc.Value
            Range("capUpper").Value = Cap
            Range("aggUpper").Value = Agg
            Range("nplUpper").Value = NPL
            Call Optimize
            IRR = Range("masterIRR").Value
            lastSheet.Cells(Rows.Count, 1).End(xlUp).Offset(1, 0).Value = Cap
            lastSheet.Cells(Rows.Count, 2).End(xlUp).Offset(1, 0).Value = Agg
            lastSheet.Cells(Rows.Count, 3).End(xlUp).Offset(1, 0).Value = NPL
            lastSheet.Cells(Rows.Count, 4).End(xlUp).Offset(1, 0).Value = IRR
            lastSheet.Cells(Rows.Count, 5).End(xlUp).Offset(1, 0).Value = Abs(Inputs.Range("gpPromote").Value)
            lastSheet.Cells(Rows.Count, 6).End(xlUp).Offset(1, 0).Value = Abs(Fees.Range("propertyManagementFee").Value)
            lastSheet.Cells(Rows.Count, 7).End(xlUp).Offset(1, 0).Value = Abs(Fees.Range("leasingFeeAfter").Value)
            lastSheet.Cells(Rows.Count, 8).End(xlUp).Offset(1, 0).Value = Abs(Fees.Range("leasingFeeDuring").Value)
            lastSheet.Cells(Rows.Count, 9).End(xlUp).Offset(1, 0).Value = Abs(Fees.Range("acquisitionFee").Value)
            lastSheet.Cells(Rows.Count, 10).End(xlUp).Offset(1, 0).Value = Abs(Fees.Range("constructionFee").Value)
            lastSheet.Cells(Rows.Count, 11).End(xlUp).Offset(1, 0).Value = Model.Range("NPLs").Value
            lastSheet.Range(lastSheet.Range("l100000").End(xlUp), lastSheet.Range("ea100000").End(xlUp)).Offset(1, 0).Value =
            lastSheet.Cells(Rows.Count, 132).End(xlUp).Offset(1, 0).Value = TotalLength
        Next
    Next
Next
```

Another piece that gave me loads of trouble was validating that the Increments of the percentage variables where wholly divisible into their ranges. That proved difficult because I used the Mod function to warn me when the remainded wasn't equal to zero. Where is the problem you say? The Mod function, for whatever reason, doesn't like decimals and I had to figure out through research that I had to multiply all those values by 10,000 to assure they were all integers.



```
    'this makes sure that the increments fit into the ranges
    If ((cmbAggUpper.Value - cmbAggLower.Value) * 1000) Mod ((cmbAggInc.Value) * 1000) <> 0 Then
        MsgBox "The aggregrate increment needs to be wholly divisible into (UpperLimit - LowerLimit).", vbExclamation, "Invalid Inputs"
        Validator = 1
    End If
    If ((cmbCapUpper.Value - cmbCapLower.Value) * 10000) Mod (cmbCapInc.Value * 10000) <> 0 Then
        MsgBox "The cap increment needs to be wholly divisible into (UpperLimit - LowerLimit).", vbExclamation, "Invalid Inputs"
        Validator = 1
    End If
    If ((cmbNPLUpper.Value - cmbNPLLower.Value) * 10000) Mod (cmbNPLInc.Value * 10000) <> 0 Then
        MsgBox "The NPL increment needs to be wholly divisible into (UpperLimit - LowerLimit).", vbExclamation, "Invalid Inputs"
        Validator = 1
    End If


End Sub
```

Assistance:

The only help I received from others during this project was from various online websites that explained how to do simple things like Downloading an Add-In when a workbook opens or finding out why the Mod function had trouble with decimal numbers. I spent many, many hours figuring out some of the more complex ideas in this macro.