

KSL Cars Classifieds Search and Regression

By Aaron Rickett

Executive Summary

My spreadsheet program is designed to help car buyers search the many advertisements on the KSL website to find the best deal. The program will first allow the user to input their search criteria. Then the program will interact with internet explorer to pull the listings from the KSL website and sort them into their own sheets by make. After the listings are pulled, the program will perform an ordinary least squares statistical regression using year, mileage and seller type as factors or predictors of price.

I designed this project to help with the task of buying a car for my own personal use. Since I am a college student I need to know that I am getting a good deal. However in the future I would like to use this project to buy cars of high value which I could then resell for a profit after fixing minor mechanical problems on the cars.

Of key importance to this potential business is finding undervalued cars to buy. Many of the advertisements are far overpriced to be worth buying. This is further complicated because there are many factors to consider to each car and it is difficult to determine exactly how much each one should be weighted when buying a car. For example is it a better deal to buy a 2010 Corolla with 25,000 miles for \$13, 000, or a 2004 Corolla with 150,000 miles. Since there is more to compare than simply price this sort of analysis requires regression to best understand each factor influence in the price of the car. This tool allows the user to see which cars are undervalued and by how much thus providing higher likelihood of profit at resale.

In order to find a good deal without this tool it could potentially require many hours to pour through the many listings. This program will automate this process and pull the key factors of each car so that the user can get a good overview. The program also pulls the URLs of each listing so that the user can easily view the listings of highest value. These features of the program make finding the car easier and less time consuming.

Implementation

The following is a list of the major components of the program with a short description of each one.

Ribbon Customization.

In order to make this program easy to access and run, the Excel's ribbon has been modified. There is a new tab called Regression which has a button to start the program by calling the user form.

User Form.

The user form allows the user to input data similar to the data required to interact with the KSL website. However unlike the KSL website which consists only of check boxes, the user form consists of list boxes, text boxes and a couple check boxes. Input data is checked for validity and modified if needed before being passed to the web query.

Distance Query

The distance query allows the user to get additional functionality that the KSL form does support. The distance query was a tool similar to the web control that used distanccheck.com to find the distance between all counties and all other counties in all six states in order to estimate the distance between a given zip and all other zip codes in that area. For speed of processing this information is held in the hidden sheet "DistMatrix". This information is used in the user form to create list of zip codes that are within a certain distance of a central zip code. This list is then passed to the web control.

Web Control.

The web control accepts the data from the user form and then accesses the internet and fills out the form on KSL and pulls up the matching listings. This is done using a special agent class. Once the listings are pulled then web control finds the name of the link and the time it was posted, for each individual advertisement and passes them to the web query.

Data Extraction.

The data from the listings is extracted by using the web query wizard tool in excel in automated VBA fashion. It receives the name of each link and uses it to pull down the whole page and copy to a hidden sheet called "dropsheet".

Data Collection.

This component searches the dropsheet after each listing is dropped to find the key pieces of information including, ad number, date, price, year, seller type, mileage, sold or not sold, city, state, zip, cell number, phone number, contact name, URL, and description. This information is sorted by into a sheet whose name is the model of the car of which the information is held.

Data Checking.

After all of the data is pulled. The data is checked in three parts. The first two parts check whether the data is suitable for a regression. Then third check is used to check if the car is truly for an investment opportunity.

Matrix Loading.

The data is read into the variant data type to be prepared for regression.

Regression.

The regression is executed using ordinary least squares and produces model coefficients, one for each of the factors, year, seller type and mileage. This regression technique relies on mathematical matrix operations.

Matrix Operations

This section includes a bundle of tools used to get the matrix inverse, transpose, and multiplication used to do the regression.

Utilities. The utilities are functions that I made myself to ease several computing processes used throughout my code.

Detailed Implementation

Upon opening the KSL Car Regression Tool workbook, the user will see the welcome page with brief instructions for using the tool. This welcome also warns users not to modify information in the hidden tabs. Fortunately the user should never have to know what is in the hidden tabs.

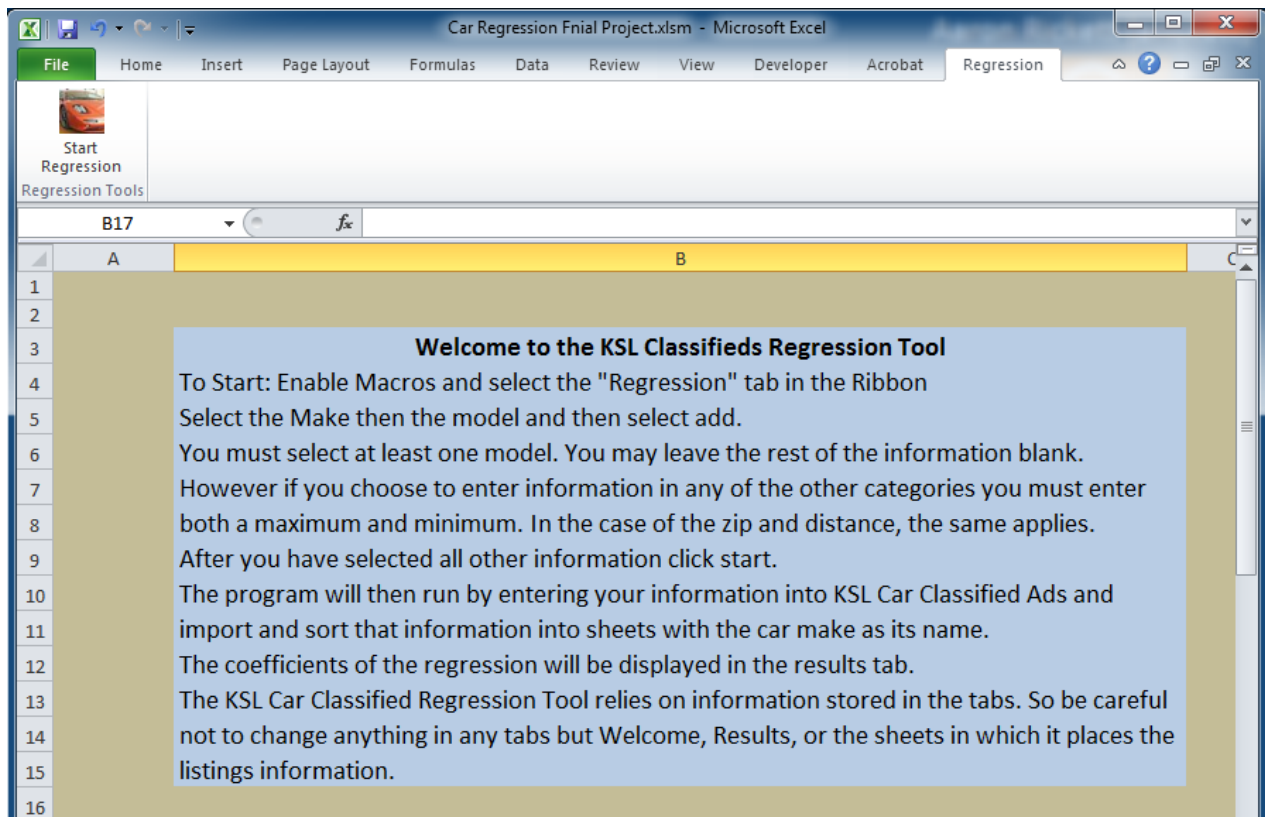


Figure 1: Welcome page and Ribbon Modification

Ribbon Customization

This button allows the user run the program without accessing the code. This is a good security measure so that a user does not accidentally change any code resulting in a program that no longer works properly. Upon clicking the start regression button the user form will appear.

User Form

The user uses the user form to select the car specification to run the search and the subsequent regression. Since there is only a need to enter in models the makes are there simply for convenience of the user. KSL will work fine if it you never enter the make of each model.

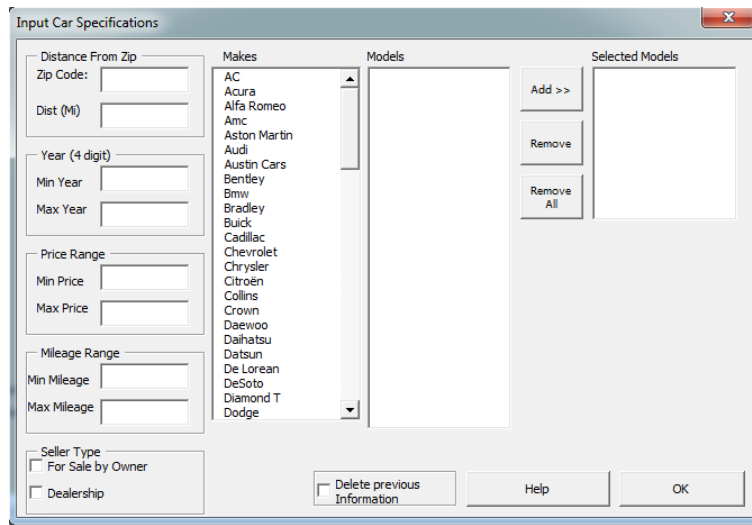


Figure 2: Select Make

Upon selecting a make from the first list box the model one will instantly populate with the corresponding models for the make selected. The user can then select a model and then click add to move it to the list box of selected models to use in the subsequent steps of the program.

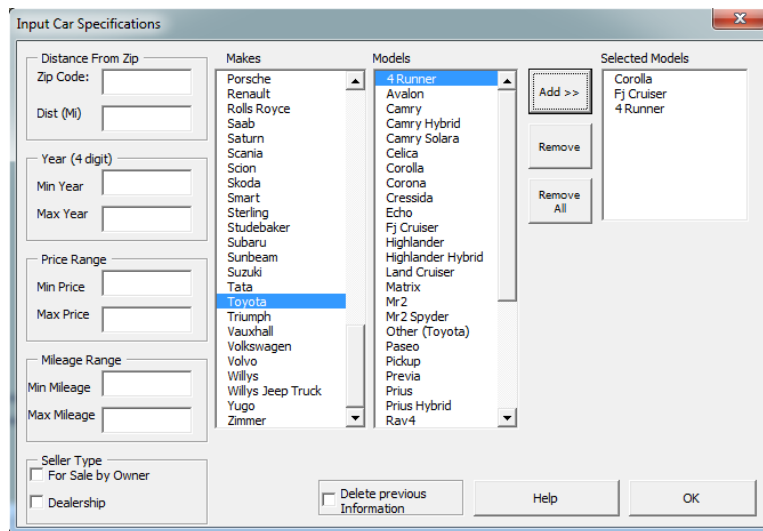


Figure 3: Select Model and click Add

If you decided to remove models that you had previously selected then you can select the model you wish to remove and then click remove. Removing all the selected models can be accomplished by clicking remove all. The makes and models information originated as a text file of a database language which had to be parsed in order to extract the makes and models.

The other text boxes work as one would expect. You can enter in any text you want, but the data will be checked for suitability before it is used by the web control. The delete previous option will enable the user

to start fresh and will clear all worksheets with information from prior regressions. Alternatively, the user may to update previous listings by reselecting the model. This will cause any additional listings to be added to the same sheet for that model.

After makes are selected and other information is entered the user can click OK to begin, at which point the data starts to be processed. Each of the pieces of data is checked. The min price is checked to be less than the max price. The min year is checked to be less than the max year and to be less than the current year plus 1.

One interesting thing is that KSL only has check boxes for every 10,000 miles and price for almost every \$1000. The mile and price data needed to be modified by rounding it in order to agree with the checkboxes in KSL Car Classifieds.

KSL uses numbers for all of their buttons names. The button names for zip codes and makes do not correspond, but for the other information it does. To speed up process time these codes were copied to a text file and parsed and then stored in the workbook. The user form creates arrays with the zip and model button numbers for use by the web control. To see the hidden button numbers see the hidden sheet “Make_Model_Lemon”.

The user form then processes the information received from the distance query. Creates new sheets for each model not previously searched and calls the web query and regression subroutines.

Distance Query

Notice the zip code and distance text boxes. These boxes and nothing with similar functionality is found on the KSL Cars classifieds website. These boxes allow the user to specify a distance from a central zip code, from which they are willing to travel to look at a car they may be interested in buying. After clicking ok, the zip code is matched with a list of zip codes of all 6 states in which KSL operates on the “Make_Model_Lemon” worksheet to find the corresponding county. The county is then found on the “distMatrix” worksheet. The row on which it is found contains the distances between that county and all other counties comprising 224 counties in total. These distances are compared to the distance entered in the textbox. If the distance in the row is less than the distance specified by the user then the corresponding county is selected and placed in an array. Then all counties are used back on the “Make_Model_Lemon” worksheet to obtain all zip codes within those counties for individual entry as checkboxes on the KSL website.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1			83616	80022	83612	81101	80251	86512	80013	81147	81073	83250	83220
2		Distance	AdalID	AdamsCC	AdamsID	AlamosaCC	AlbanyWY	ApacheA	ArapahoeCC	ArchuletaCC	BacaCI	BannockID	BearLakeID
3	83616	AdalID	0	648.27	68.07	698.37	655.77	696.36	659.25	659.92	841.3	222.89	273.88
4	80022	AdamsCO	648.27	0	678.9	171.11	11.88	401.24	13.31	213.63	208	425.73	374.59
5	83612	AdamsID	68.07	678.9	0	742.32	687.2	755.37	690.51	708.05	877.5	255.81	306.43
6	81101	AlamosaCO	698.37	171.11	742.32	0	160.8	248.27	162.01	68.6	178	488.92	440.81
7	80251	AlbanyWY	655.77	11.88	687.2	160.8	0	393.59	3.88	205.39	197.5	433.47	382.31
8	86512	ApacheAZ	696.36	401.24	755.37	248.27	393.59	0	395.96	188.89	402.8	535.96	502.08
9	80013	ArapahoeCO	659.25	13.31	690.51	162.01	3.88	395.96	0	207.59	194.9	436.9	385.74
10	81147	ArchuletaCO	659.92	213.63	708.05	68.6	205.39	188.89	207.59	0	245.3	459.67	414.45
11	81073	BacaCO	841.32	208.03	877.46	177.98	197.45	402.78	194.93	245.32	0	621.85	571.04
12	83250	BannockID	222.89	425.73	255.81	488.92	433.47	535.96	436.9	459.67	621.9	0	51.16
13	83220	BearLakeID	273.88	374.59	306.43	440.81	382.31	502.08	385.74	414.45	571	51.16	0
14	84713	BeaverUT	419.47	432.33	482.19	377.46	432.26	283.86	436.05	315.44	553.6	301.27	291.63
15	83861	BenewahID	249.07	778.46	181.04	870.05	788.33	915.72	791.17	845.52	985	386.89	431.11
16	81044	BentCO	801.99	162.2	836.32	164.95	152.09	405.06	149.32	233.5	47.86	581.21	530.19
17	87110	BernalilloNM	790.15	338.95	842.74	168.02	328.4	155.78	329.4	151.09	271	601.33	558.65
18	82426	BigHornWY	416.42	359.76	414.04	497.64	371.24	644.74	373.06	502.03	565.2	243.81	221.77

Figure 4: The Table that holds the distance information

Web Control

The web control uses the processed data from the user form and the agent class and methods to control the internet by finding key elements of the HTML code of the KSL website and then executing them. In the web control there was substantial error handling because even with the rounded prices and miles amounts, not every mileage amount exists. Without error checking the web control would hang if it tried to access a check box that does not exist. For this reason the max mileage was increased in increments of 10,000 until a match is found, and the min mileage is decreased by 10,000 until a match is found. The prices button numbers were similarly handled but in increments of \$1,000.

Each of the sets of buttons is cleared when the web control loads a new model. This is because if KSL runs again in the same instance of internet explorer it will remember the previous values from the run. The commands for the selecting the model are strategically placed last so that so that the other specification do not need to be included in the loop, since all the specifications are being applied to all of the models selected. After all information is loaded the load results button is accessed.



Cars & Trucks (4,001,301 Served)

See Also: Auto Parts, Motorcycles, Dirt Bikes, Motorcycles, Road Bikes, Snowmobiles, Camp and Travel Trailers, Wheels and Tires, Auto Parts

Body:	Click Here	Year:	Click Here
Make:	Click Here	Price:	Click Here
Model:	Click Here	Dealer:	Click Here
State:		De:	Click Here
Show advanced search		Clear search...	

Model

Title	Total
<input type="checkbox"/> 1 Series	6
<input type="checkbox"/> 100	1
<input type="checkbox"/> 120	1
<input type="checkbox"/> 128i	3
<input type="checkbox"/> 135i	4
<input type="checkbox"/> 1800S	1
<input type="checkbox"/> 190E	6
<input type="checkbox"/> 200	4
<input type="checkbox"/> 200SX	6
<input type="checkbox"/> 220SE	3
<input type="checkbox"/> 240	8
<input type="checkbox"/> 240 GL	1

OK Cancel Clear


available Matches

ads

Sell A Car

Coupons

FEATURED AD



[Click here to learn how!](#)

List Newest to Oldest

Figure 5: Illustration of the Web Control submitting data

Then the web control parses to find the number of results returned. If it finds 0 the next model immediately begins loading.

If results are returned the Web Control continues to parse through the HTML until it finds the link to the page containing all of the information for each listing.

This link is then passed to the Web Query wizard for extraction.

Data Extraction

The data extraction uses the link from the web control to copy the information from the specific listing page and dump it into a worksheet called “dropsheet”. This is by far faster (and easier) than using the web control to do the same job.

	A	B	C	D	E	F
44	About this ad					
45	Cars & Trucks					
46	(4,001,301 Served)					
47	See Also: Auto Parts, Motorcycles, Dirt Bikes, Motorcycles, Road Bikes, Snowmobiles, Camp and Travel Trailers, Wheels and Tires, Auto I					
48						
49	List View					
50	Photo View					
51	My Car Ads					
52	Sell A Car					
53	Coupons					
54						
55	E-mail Seller					
56	Report Abuse					
57	2003 Nissan Altima					
58	Syracuse, UT 84075 - Apr 8, 2012					
59	\$790,000					
60	For Sale by Owner					
61	Ad: # 7884815					
62	Contact: Gary					
63	Cell: 801-608-2435					
64	Member since: Jan 12					
65	Page Views: 57					
66	Favorite of: 0 people					
67	Awesome Nissan Altima SL. 120k miles on car with new engine(35,000 miles) leather interior, heated seats, sunroof, power everything,					
68	Seller Information					
69	Seller Type:					
70	For Sale By Owner					

Figure 6: The Drop Sheet

Data Collection

With the information in the drop sheet, data collection can now pull the information from the sheet and organize it in a sheet with the models name. This part included utilities to remove unwanted characters from the data that are pulled.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Time Ad Placed	AdNumber	Date	Price	Year	Seller Type	Mileage	Sold	City	State	Zip	Cell Number	Phone Number	Contact Name
2	4/12/2012 16:42	8025381	Apr 12, 2	15495	2010	Dealership	42886	Not Sold	Bounti	UT	84010		1800548133	PaulRainy
3	4/12/2012 16:15	8024789	Apr 12, 2	10495	2006	Dealership	64000	Not Sold	SaltLak	UT	84115	8016614332	8015487760	Matt
4	4/12/2012 16:06	8024749	Apr 12, 2	10750	2008	Specifications	77000	Not Sold	Sandy	UT	84070	8018421982		james
5	4/12/2012 16:04	5793744	Apr 12, 2	17895	2012	Dealership	31	Not Sold	Draper	UT	84020		1888791638	Kris
6	4/12/2012 15:54	8023905	Apr 12, 2	12950	2010	Dealership	52555	Not Sold	SaltLak	UT	84115		8014831011	Marcos
7	4/12/2012 15:54	8023888	Apr 12, 2	8376	2004	Dealership	120389	Not Sold	Ogden	UT	84401		8017840033	JustinBooth
8	4/12/2012 14:54	8022257	Apr 12, 2	11990	2008	Dealership	71892	Not Sold	Ogden	UT	84401		8017840033	JustinBooth
9	4/12/2012 14:54	8021579	Apr 12, 2	17988	2011	Dealership	9571	Not Sold					1888929945	
10	4/12/2012 15:02	8021364	Apr 12, 2	4495	2000	Dealership	170083	Not Sold	Midval	UT	84047		8015667799	DaveorScot
11	4/12/2012 14:02	8021205	Apr 12, 2	10595	2008	Dealership	47000	Not Sold	Lehi	UT	84043		8013418146	Stephen
12	4/12/2012 14:02	8019930	Apr 12, 2	9899	2007	For Sale By Ow	89596	Not Sold	Lindon	UT	84042	8013616770		Jonathon
13	4/12/2012 14:02	8019881	Apr 12, 2	12995	2010	Dealership	35000	Not Sold	Lehi	UT	84043		8013418146	Stephen
14	4/12/2012 14:02	8019875	Apr 12, 2	10995	2005	Dealership	67430	Not Sold	958s.st slc	UT		8012432443	8019531786	SalesDept.
15	4/12/2012 13:02	5793750	Apr 12, 2	17895	2012	Dealership	8	Not Sold	Draper	UT	84020		1888791638	Kris
16	4/12/2012 13:02	8018473	Apr 12, 2	4500	2002	For Sale By Ow	143700	Not Sold	IdahoF	ID	83401		2083517458	Matt
17	4/12/2012 13:02	8018404	Apr 12, 2	6900	2005	Dealership	93677	Not Sold	SLC	UT	84111	8016711435	8016356274	JoseorNico
18	4/12/2012 12:02	8018385	Apr 12, 2	4995	2000	Dealership	104214	Not Sold	Center	UT	84014		8019136566	Ben
19	4/12/2012 12:02	8018376	Apr 12, 2	12995	2010	Dealership	39000	Not Sold	Lehi	UT	84043		8013418146	Stephen
20	4/12/2012 12:02	8017091	Apr 12, 2	8995	2005	Dealership	92000	Not Sold	Murray	UT	84107	8015989289	8012882838	Sales
21	4/12/2012 12:02	8015611	Apr 12, 2	1500	1991	For Sale By Ow	194489	Not Sold	Provo	UT	84604	8016697768		Angel
22	4/12/2012 11:02	8015500	Apr 12, 2	3500	1999	For Sale By Ow	195000	Not Sold	Woods	UT	84087	8014142205	8014142205	Diego
23	4/12/2012 11:02	8015480	Apr 12, 2	9995	2007	Dealership	70000	Not Sold	Bounti	UT	84010		8019910421	jasonorMatt

Data Checking

After all listings for all models are dropped collected and sorted. The data checking begins. For each sheet containing data from the listings, data checking begins.

There are three pars to data checking.

1. The first part checks for duplicate listings and removes duplicates. Since KSL default list order is newest to oldest many people relist the same car very often. These duplicates have the potential to skew the results of a regression, so they must be removed first.
2. The data used in the regression is checked for completeness. If any number is missing it is either deleted or set to 0. It is set to 0 only if it is the mileage and the car is a new car sold by a dealership.
3. The second par checks for any information in each cars description that identifies the car as unsuitable for an investment opportunity. It does this by checking a listing of words that are used to describe the poorest quality cars. If any of those words are found then the information is deleted from the sheet. The list of words comes from the hidden sheet “Make_Model_Lemon”.

Matrix Loading.

After the data is checked it must be loaded into variants which can act like matrices. The prices are all loaded into a variant with $1 \times n$ dimensions. The other factors are found and entered into another variant with $n \times \text{number_of_factors}$ as its dimensions. Since matrices contain numbers, the seller type variable was entered as a binary entry, 1 for “For Sale by Owner” and 0 for “Dealership”.

Regression.

An ordinary least squares regression is now done. The calculation involves the matrix operations transpose, multiplication and inverse. The equation is as follows.

$$\beta = (X^T X)^{-1} X^T Y$$

$$\mathbf{Yhat} = X \beta$$

Where β is the coefficient matrix and \mathbf{Yhat} is an expected value for the price given a set of factors.

The coefficients are then reported to the results page. The \mathbf{Yhat} can be used for further analysis but is out of the scope of this program.

	A	B	C	D
1	Model	Year	Seller Type	Mileage
2	Corolla	8.848302	-2031.0204	-0.07542
3				
4				
5				

Figure 7: The Results Page

Matrix Operations.

Since the variants that contained the data were unsuitable for the matrix tools that come with excel (price matrix is an $n \times 1$ matrix). It was needful to recreate these function that can handle the variants. This includes matrix transpose, matrix inverse and 2 functions for matrix multiplication, one for the $n \times 1$ case and for the $n \times n$ case. The matrix inverse operation uses the SVD class to use singular value decomposition instead of Gauss-Jordan elimination to obtain the inverse.

Utilities.

These include several helpful functions and subs that were employed numerous times throughout the project. It includes functions for removing certain characters from a string, array operations like array compliments, function to get rid of empty entries in arrays, a function to tell if an array is empty etc.

Learning

I learned that VBA programming can be difficult and time consuming. But it is also rewarding to see the creation of a new useful tool. I wish I had kept track of the hours but I think I have spent around 120

hours creating this program. I feel accomplished in that I can use this program to find a car with high value, and also have a great project to show potential employers.

I felt that my project used most of the major concepts covered in class. I used arrays and variants extensively. I created a user form which reacted to the event "change". In the case of the Makes list box. I used error checking extensively, as well as both tools to interact with the internet. I used created my own functions. I learned to parse through 3text files to get the information needed to create the program. I created and ran the distance query for almost 3days to obtain all of the distance information (25,088 iterations). I used many nested loops, and conditional statements.

There were even minor concepts covered that I learned, but did not learn in class. Such as arrays cannot be passed as optional arguments, you have to use variants. I learned about converting and working with dates. This is used when I got the time that each ad was made. I had to use the date serial and date value functions to get an accurate time reading for recent updates.

I learned that it is far easier to use the web query wizard when getting information from the internet, than the agent. The agent is best used to submit information over the internet. I learned several debugging techniques such as the lines:

```
If i = 67 then
```

```
i = 67
```

```
End if
```

This is to call place a breakpoint in the middle of a loop when i = 67, and then stepping through the code to check for errors with the locals window open.

I learned that it is very important to break my code into small manageable chunks. So that different components can be called at different times. If there is one thing I would change I would try to do that even more. I learned to create testing subs to check my functions and other subs.

I would like to add even more features to this program such as email and text message abilities. I would also like to schedule reruns every so often so that my program can find cars with great value while I'm at work. I would also like to do more with the regression to study the residuals and test if the regression meets the assumptions needed for a regression and maybe automate a transformation of variables.

Difficulties

I eventually resolved all of my difficulties, but some of them had me stumped for a long time. Such as while using the agent sometimes a string literal would execute just fine, but a variable version would not. I after talking to Dr. Allen, we discovered that converting the variable to an integer or long and then right back to a string worked.

I had a lot of difficulties with the agent. I then realized that for some reason the `a.explorer.goback` method caused the internet explorer to go back but the text html that I was reading from would not go back while using that command. For that reason it became necessary to get the URL while I was on the previous page and then follow the link by text. In order to move back and have the html text update.

Assistance

I received some assistance from Dr. Allen especially about using the agent. All I did consult the internet quite a bit for specific problems about functions and syntax I was not aware of. I did manage to find the SVD class online which allowed me to preform my own version of matrix inverse much more quickly than by doing the Gauss Jordan elimination that I learned in linear algebra class. However excel has a matrix inverse function called `MINVERSE` which works on multidimensional arrays. I however needed it to work for variants. So I used their code from Vanna. I also of course did not create the agent class.