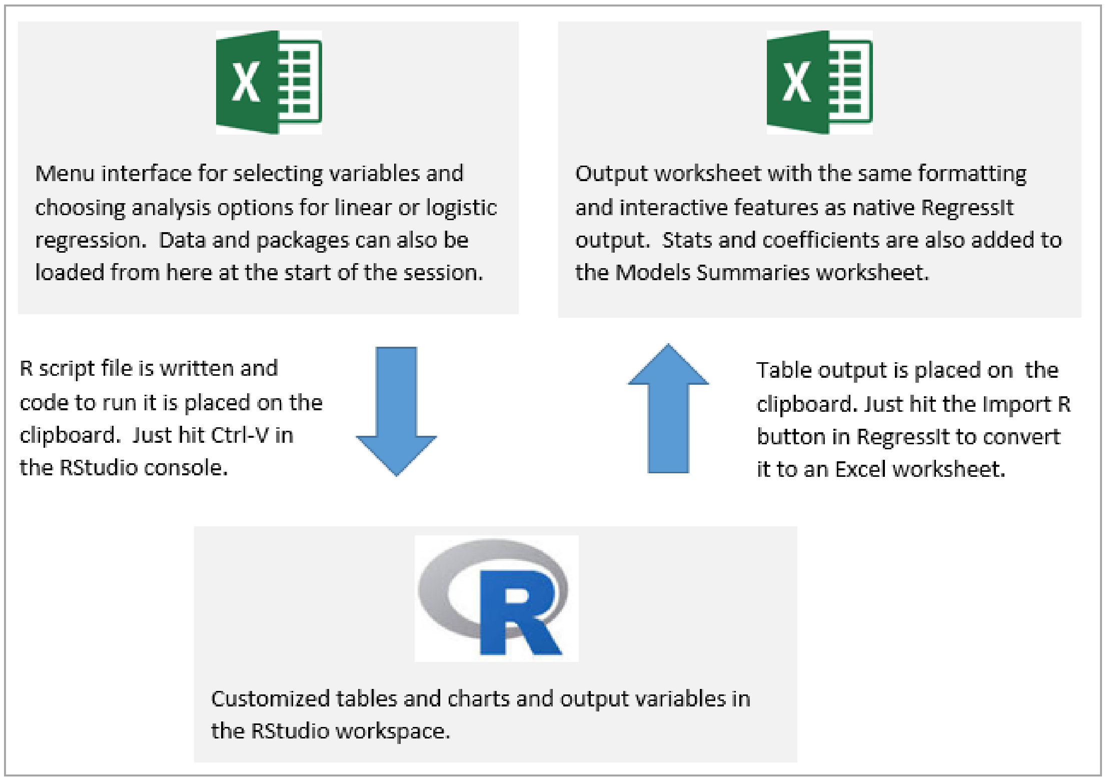
RegressIt

Excel to R and back[1]

The R interface in RegressIt allows the user to transfer data from an Excel file to a new data frame in RStudio, load packages, and run regression models with customized table and chart output by making selections from dialog boxes in Excel and doing nothing more in RStudio than to hit Control-V and Enter in order to paste and run lines of code from the clipboard. For the fitting of a regression model, RegressIt writes an R script file and then places a single line of code on the clipboard for running the script. If the "detailed output" option is chosen, the script might contain several hundred lines of code, most of which is used for customized outputs and creation of descriptive temporary variables. When the "basic output" option is chosen, only generic R output is created and the script typically contains between 10 and 30 lines of code depending on testing and variable selection options.

When a detailed-output script is run, it can not only produce output in RStudio but also write the table output to the clipboard for exporting it back to Excel. The Import-R button on the RegressIt ribbon handles all the details of creating a new worksheet, pasting the R output on it, reformatting the output to look like native RegressIt output, and adding an entry to the model summaries worksheet. From the Excel user's perspective, only a few extra keystrokes are require to do the model estimation in R rather than Excel.

Overall the process of specifying a model in Excel, using RStudio to run it, and importing the results back to Excel looks like this:



Menu interface for selecting variables and choosing analysis options for linear or logistic regression. Data and packages can also be loaded from here at the start of the session.

Output worksheet with the same formatting and interactive features as native RegressIt output. Stats and coefficients are also added to the Models Summaries worksheet.

R script file is written and code to run it is placed on the clipboard. Just hit Ctrl-V in the RStudio console.

Table output is placed on the clipboard. Just hit the Import R button in RegressIt to convert it to an Excel worksheet.

Customized tables and charts and output variables in the RStudio workspace.

[1] March 1, 2020

For those who normally work primarily in Excel, this feature provides a push-button interface for running linear and logistic regression models in R with customized table output that can be imported back to Excel in a single keystroke. This has exactly the same format as RegressIt's own native Excel output, including an audit trail on the model summaries worksheet and also including features such as applying color coding to highlight significance of coefficient estimates and their correlations and interactively de-selecting variables in the coefficient table for purposes of launching the next model. Chart output in RStudio is not automatically imported to Excel, but it can be copied and pasted piecemeal with a few more keystrokes, and the charts produced by the R script have descriptive titles that include the model name and some parameters of the model, as do the charts produced by RegressIt in Excel. Thus, RStudio can serve as a computational engine for work performed in Excel, providing additional modeling features such as stepwise variable selection and several different forms of out-of-sample testing, as well as the ability to handle very large models.[2] At the same time the Excel user can get a gentle introduction to the R programming environment and use it for cross-checking the accuracy of the results that RegressIt produces in Excel, if anyone is curious.

For those who normally work primarily in R, it provides the capability to use Excel as a back end for producing easily shareable presentation quality output, including a publication-style model comparison table like the ones produced by the Stargazer package. The ability to sort the coefficient table on P-values and interactively delete insignificant ones can also be used to advantage while going back and forth between Excel and R, even when all of the estimation is being carried out in R. The Excel file can also serve as an easy-to-navigate audit trail for all models fitted. The same models can also be instantly refitted in Excel to get additional outputs, including RegressIt's own presentation-quality charts.

There is also an option to specify the name of an existing R model when choosing options within the R code dialog box in Excel. When this box is checked, the variable selections will be those of the indicated model. (If stepwise variable selection is also chosen, the variables in the existing model will be used as the starting set.) This feature can be used to generate Excel output for models already existing in the R workspace.

The following four pages show the sequence of screen shots for the process of fitting a model in this way. The same analysis is illustrated in a video that can be found at this link:

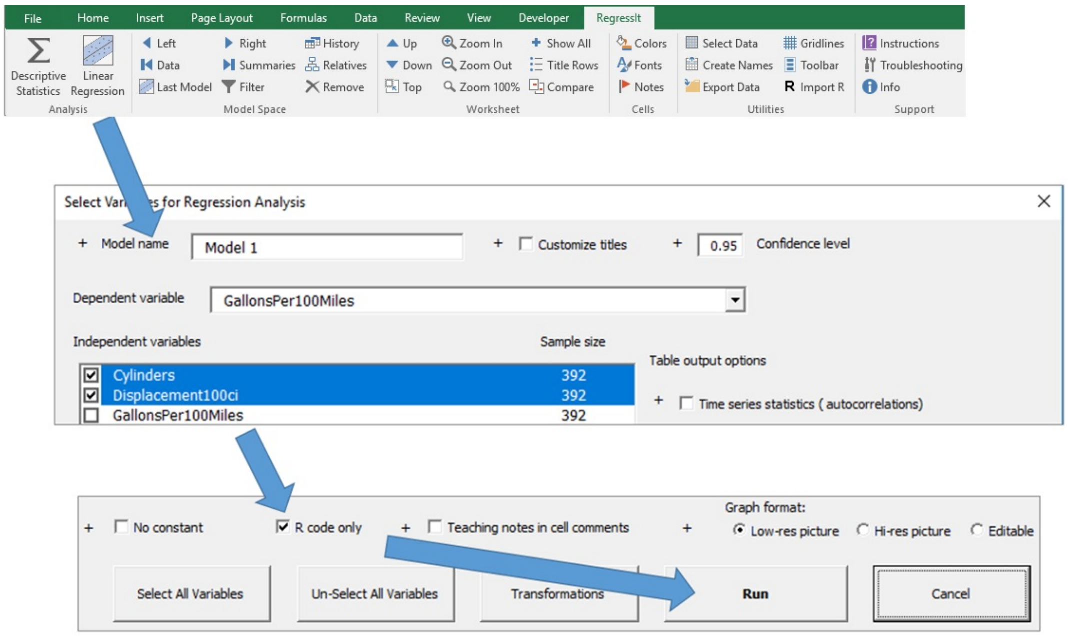
http://regressit.com/RegressIt_Excel_to_R_interface.mp4

More details of the data set and the analysis can be found on this web page:

http://regressit.com/excel-to-r.html

[2] On a well-equipped machine RegressIt can fit a linear regression model within Excel with up to 1M rows of data and up to 200 independent variables, but the practical limit is around 5M cells in the matrix of selected independent variables, and computation time can be slow (on the order of minutes) as that limit is approached. For logistic regression models the row limit is around 16K. Much larger models than these can be fitted in a few seconds via the R interface.

First, click the linear regression button on the RegressIt ribbon, select the variables you want to use, then check the R code only box before hitting Run:



Select Variables for Regression Analysis

+ Model name: Model 1 | + ☐ Customize titles | + 0.95 Confidence level

Dependent variable: GallonsPer100Miles

Independent variables	Sample size
☑ Cylinders	392
☑ Displacement100ci	392
☐ GallonsPer100Miles	392

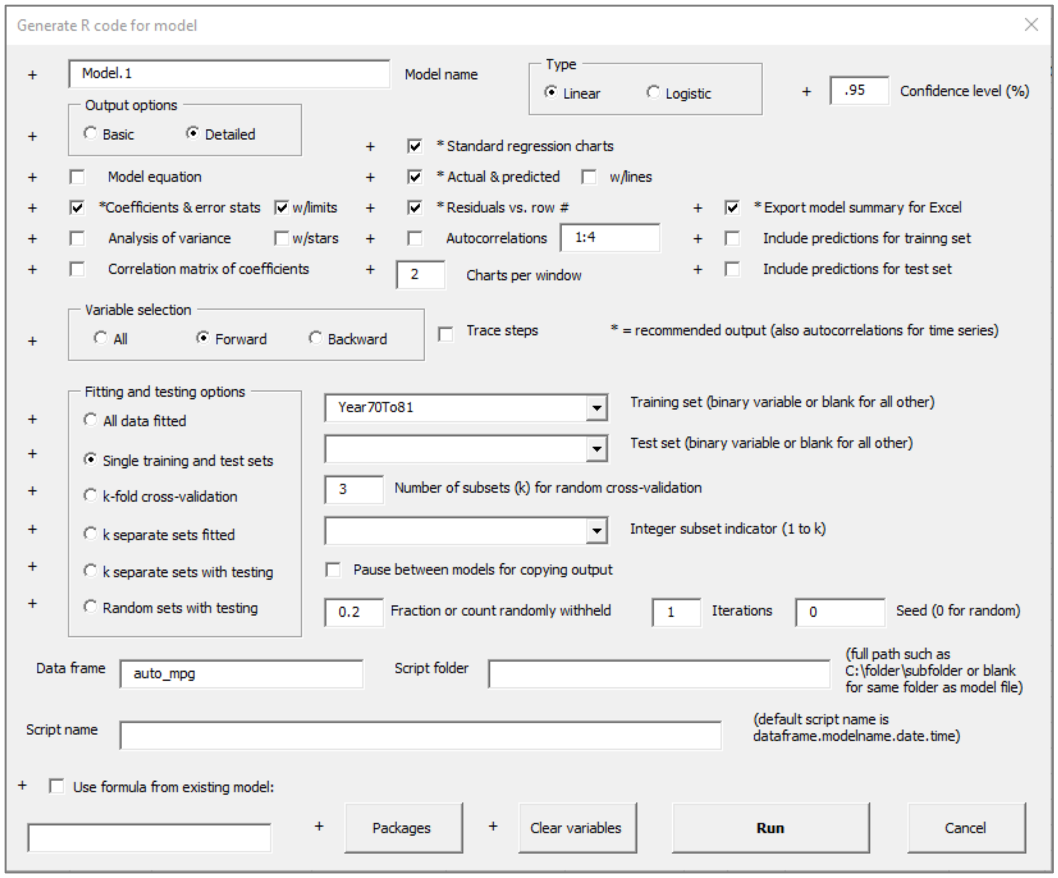
Table output options

+ ☐ Time series statistics (autocorrelations)

+ ☐ No constant | ☑ R code only | + ☐ Teaching notes in cell comments | +

Graph format: [✓ Low-res picture] Hi-res picture Editable

This will open the dialog box for specifying the model type (linear or logistic), table and chart output to be produced, variable selection method (all variables or stepwise), and testing method, if any. **Choose your desired model options and hit Run.**



Generate R code for model

+ Model.1 Model name

Type: [✓ Linear] Logistic

+ .95 Confidence level (%)

+ Output options: Basic [✓ Detailed]

+ ☐ Model equation
+ ☑ *Coefficients & error stats ☑ w/limits
+ ☐ Analysis of variance ☐ w/stars
+ ☐ Correlation matrix of coefficents

+ ☑ * Standard regression charts
+ ☑ * Actual & predicted ☐ w/lines
+ ☑ * Residuals vs. row #
+ ☐ Autocorrelations 1:4
+ 2 Charts per window

+ ☑ * Export model summary for Excel
+ ☐ Include predictions for trainng set
+ ☐ Include predictions for test set

+ Variable selection: All [✓ Forward] Backward

☐ Trace steps

* = recommended output (also autocorrelations for time series)

Fitting and testing options
+ All data fitted
+ [✓ Single training and test sets]
+ k-fold cross-validation
+ k separate sets fitted
+ k separate sets with testing
+ Random sets with testing

Year70To81 Training set (binary variable or blank for all other)

Test set (binary variable or blank for all other)

3 Number of subsets (k) for random cross-validation

Integer subset indicator (1 to k)

☐ Pause between models for copying output

0.2 Fraction or count randomly withheld | 1 Iterations | 0 Seed (0 for random)

Data frame: auto_mpg

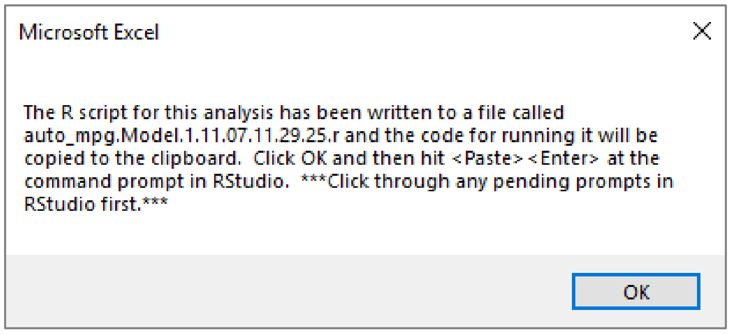
Script folder: (full path such as C:\folder\subfolder or blank for same folder as model file)

Script name: (default script name is dataframe.modelname.date.time)

+ ☐ Use formula from existing model:

3

The code for the analysis will be written to a script file and the "source" command line for running it will be placed on the clipboard. The default script name includes the data frame name, model name, and date and run time, although it can be customized:

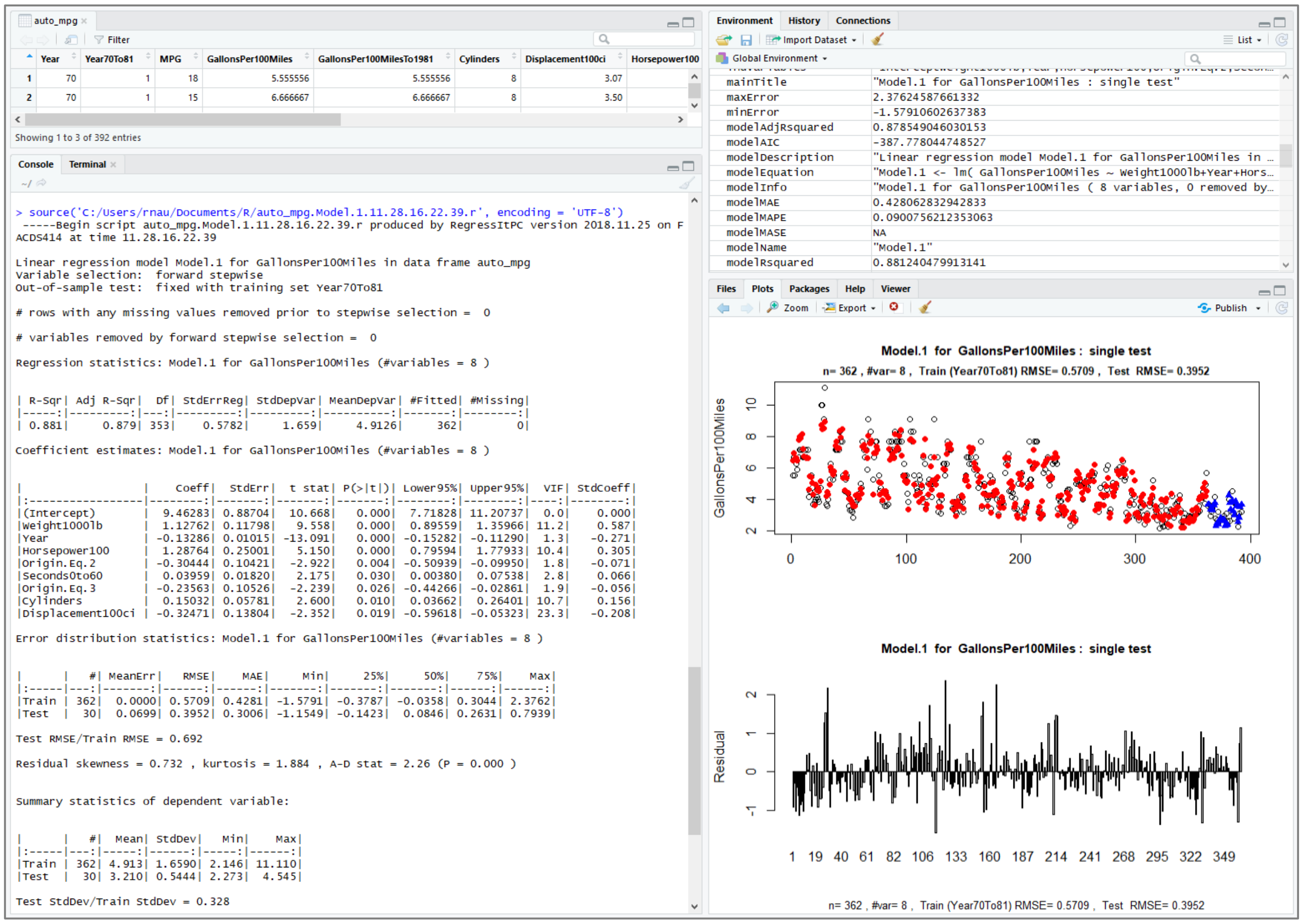


Microsoft Excel

The R script for this analysis has been written to a file called auto_mpg.Model.1.11.07.11.29.25.r and the code for running it will be copied to the clipboard. Click OK and then hit <Paste><Enter> at the command prompt in RStudio. ***Click through any pending prompts in RStudio first.***

OK

Now go back to RStudio and just hit Ctrl-V and Enter at the command prompt. That's all you need to type in RStudio! The output produced for this model is shown below. The standard 2x2 array of diagnostic plots for a regression model was also included and is in a separate plot window. Note the model-specific chart titles. A large number of temporary variables are also created to store the results of various calculations that were performed (some are lists, some are numbers, and some are arrays or data frames).



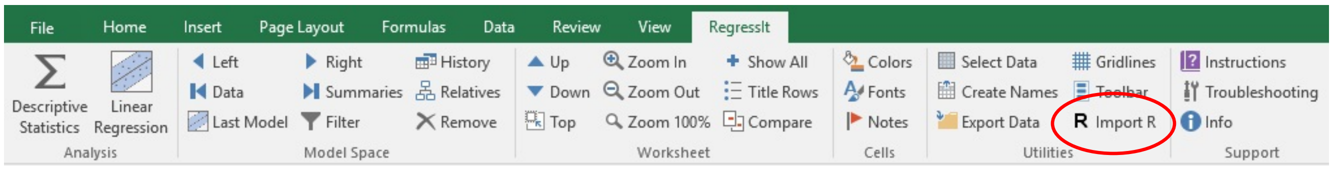
	Year	Year70To81	MPG	GallonsPer100Miles	GallonsPer100MilesTo1981	Cylinders	Displacement100ci	Horsepower100
1	70	1	18	5.555556	5.555556	8	3.07	
2	70	1	15	6.666667	6.666667	8	3.50	

Showing 1 to 3 of 392 entries

```
> source('C:/Users/rnau/Documents/R/auto_mpg.Model.1.11.28.16.22.39.r', encoding = 'UTF-8')
 -----Begin script auto_mpg.Model.1.11.28.16.22.39.r produced by RegressItPC version 2018.11.25 on F
ACDS414 at time 11.28.16.22.39

Linear regression model Model.1 for GallonsPer100Miles in data frame auto_mpg
Variable selection:  forward stepwise
Out-of-sample test:  fixed with training set Year70To81

# rows with any missing values removed prior to stepwise selection =  0

# variables removed by forward stepwise selection =  0

Regression statistics: Model.1 for GallonsPer100Miles (#variables = 8 )


| R-Sqr| Adj R-Sqr|  Df| StdErrReg| StdDepVar| MeanDepVar| #Fitted| #Missing|
|-----:|---------:|---:|---------:|---------:|----------:|-------:|--------:|
| 0.881|     0.879| 353|    0.5782|     1.659|     4.9126|     362|        0|

Coefficient estimates: Model.1 for GallonsPer100Miles (#variables = 8 )


|                  |    Coeff|  StdErr|  t stat| P(>|t|)| Lower95%| Upper95%|  VIF| StdCoeff|
|:-----------------|--------:|-------:|-------:|-------:|--------:|--------:|----:|--------:|
|(Intercept)       |  9.46283| 0.88704|  10.668|   0.000|  7.71828| 11.20737|  0.0|    0.000|
|Weight1000lb      |  1.12762| 0.11798|   9.558|   0.000|  0.89559|  1.35966| 11.2|    0.587|
|Year              | -0.13286| 0.01015| -13.091|   0.000| -0.15282| -0.11290|  1.3|   -0.271|
|Horsepower100     |  1.28764| 0.25001|   5.150|   0.000|  0.79594|  1.77933| 10.4|    0.305|
|Origin.Eq.2       | -0.30444| 0.10421|  -2.922|   0.004| -0.50939| -0.09950|  1.8|   -0.071|
|Seconds0to60      |  0.03959| 0.01820|   2.175|   0.030|  0.00380|  0.07538|  2.8|    0.066|
|Origin.Eq.3       | -0.23563| 0.10526|  -2.239|   0.026| -0.44266| -0.02861|  1.9|   -0.056|
|Cylinders         |  0.15032| 0.05781|   2.600|   0.010|  0.03662|  0.26401| 10.7|    0.156|
|Displacement100ci | -0.32471| 0.13804|  -2.352|   0.019| -0.59618| -0.05323| 23.3|   -0.208|

Error distribution statistics: Model.1 for GallonsPer100Miles (#variables = 8 )


|      |   #| MeanErr|   RMSE|    MAE|     Min|     25%|     50%|    75%|    Max|
|:-----|---:|-------:|------:|------:|-------:|-------:|-------:|------:|------:|
|Train | 362|  0.0000| 0.5709| 0.4281| -1.5791| -0.3787| -0.0358| 0.3044| 2.3762|
|Test  |  30|  0.0699| 0.3952| 0.3006| -1.1549| -0.1423|  0.0846| 0.2631| 0.7939|

Test RMSE/Train RMSE = 0.692

Residual skewness = 0.732 , kurtosis = 1.884 , A-D stat = 2.26 (P = 0.000 )


Summary statistics of dependent variable:


|      |   #|  Mean| StdDev|   Min|    Max|
|:-----|---:|-----:|------:|-----:|------:|
|Train | 362| 4.913| 1.6590| 2.146| 11.110|
|Test  |  30| 3.210| 0.5444| 2.273|  4.545|

Test StdDev/Train StdDev = 0.328
```

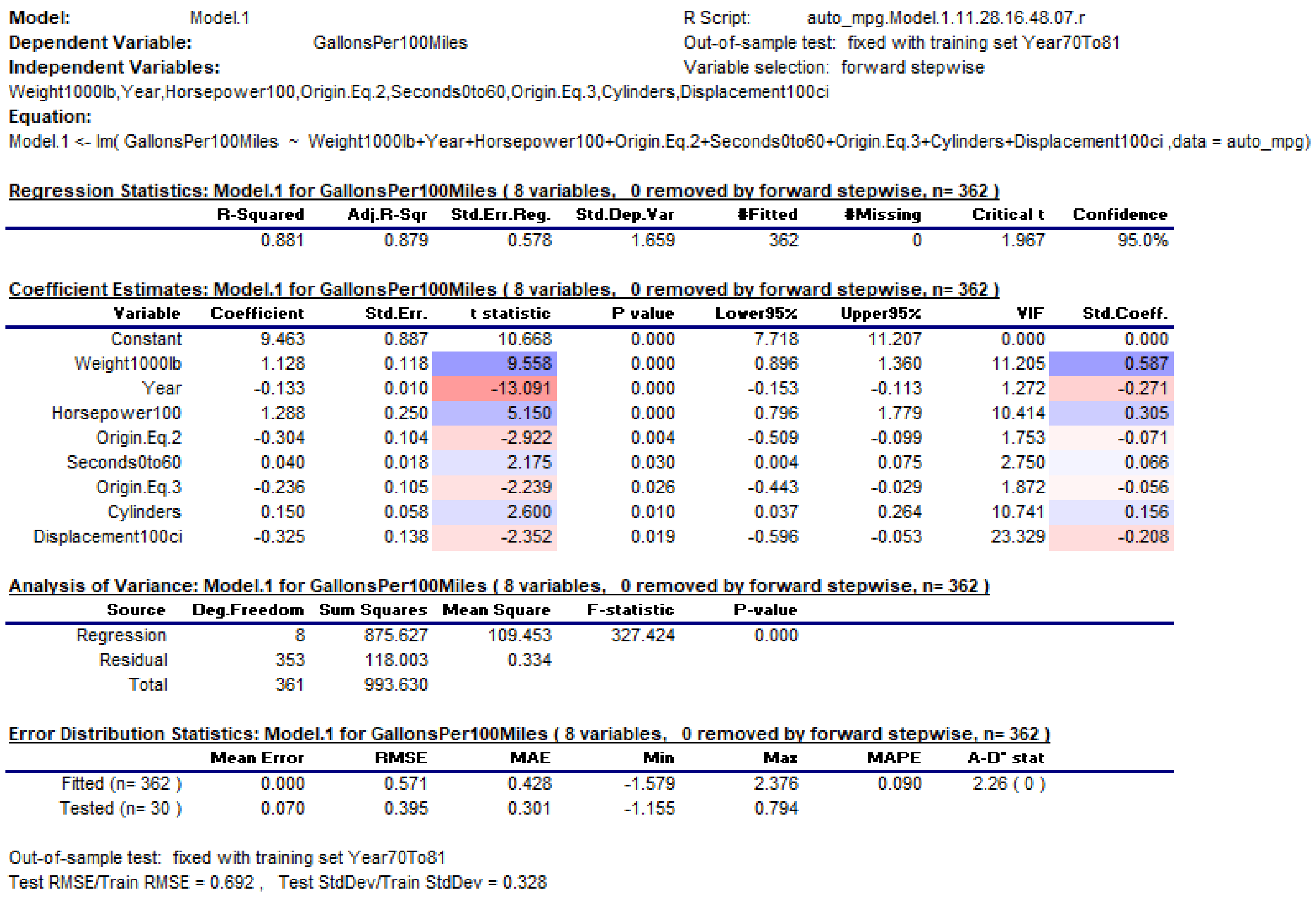
mainTitle	"Model.1 for GallonsPer100Miles : single test"
maxError	2.37624587661332
minError	-1.57910602637383
modelAdjRsquared	0.878549046030153
modelAIC	-387.778044748527
modelDescription	"Linear regression model Model.1 for GallonsPer100Miles in …
modelEquation	"Model.1 <- lm(GallonsPer100Miles ~ Weight1000lb+Year+Hors…
modelInfo	"Model.1 for GallonsPer100Miles (8 variables, 0 removed by…
modelMAE	0.428062832942833
modelMAPE	0.0900756212353063
modelMASE	NA
modelName	"Model.1"
modelRsquared	0.881240479913141

Lastly, go back to Excel and hit the "Import R" button.



This will place the output of the model on a new worksheet, as shown below, formatted in exactly the same way as a model run that is run within RegressIt. Numbers are formatted to show appropriate numbers of decimal places in blocks of 3. The exact values are available inside the cells. Every table has a model-specific title above it which is useful for audit trail purposes if the table is copied elsewhere by itself. (Charts produced in RegressIt also carry this information in their titles.) The display of these titles can be toggled on and off via buttons on the ribbon, and individual charts and tables can also be selectively hidden from view.

Interactive color coding of coefficient estimates (as well as residual autocorrelations and correlations of coefficient estimates if present) can be applied using the Colors button on the ribbon, and variables can be interactively deleted from the next model launched from this sheet:



Model: Model.1 — R Script: auto_mpg.Model.1.11.28.16.48.07.r

Dependent Variable: GallonsPer100Miles — Out-of-sample test: fixed with training set Year70To81

Independent Variables: — Variable selection: forward stepwise

Weight1000lb,Year,Horsepower100,Origin.Eq.2,Seconds0to60,Origin.Eq.3,Cylinders,Displacement100ci

Equation:

Model.1 <- lm(GallonsPer100Miles ~ Weight1000lb+Year+Horsepower100+Origin.Eq.2+Seconds0to60+Origin.Eq.3+Cylinders+Displacement100ci ,data = auto_mpg)

Regression Statistics: Model.1 for GallonsPer100Miles (8 variables, 0 removed by forward stepwise, n= 362)

R-Squared	Adj.R-Sqr	Std.Err.Reg.	Std.Dep.Var	#Fitted	#Missing	Critical t	Confidence
0.881	0.879	0.578	1.659	362	0	1.967	95.0%

Coefficient Estimates: Model.1 for GallonsPer100Miles (8 variables, 0 removed by forward stepwise, n= 362)

Variable	Coefficient	Std.Err.	t statistic	P value	Lower95%	Upper95%	VIF	Std.Coeff.
Constant	9.463	0.887	10.668	0.000	7.718	11.207	0.000	0.000
Weight1000lb	1.128	0.118	9.558	0.000	0.896	1.360	11.205	0.587
Year	-0.133	0.010	-13.091	0.000	-0.153	-0.113	1.272	-0.271
Horsepower100	1.288	0.250	5.150	0.000	0.796	1.779	10.414	0.305
Origin.Eq.2	-0.304	0.104	-2.922	0.004	-0.509	-0.099	1.753	-0.071
Seconds0to60	0.040	0.018	2.175	0.030	0.004	0.075	2.750	0.066
Origin.Eq.3	-0.236	0.105	-2.239	0.026	-0.443	-0.029	1.872	-0.056
Cylinders	0.150	0.058	2.600	0.010	0.037	0.264	10.741	0.156
Displacement100ci	-0.325	0.138	-2.352	0.019	-0.596	-0.053	23.329	-0.208

Analysis of Variance: Model.1 for GallonsPer100Miles (8 variables, 0 removed by forward stepwise, n= 362)

Source	Deg.Freedom	Sum Squares	Mean Square	F-statistic	P-value
Regression	8	875.627	109.453	327.424	0.000
Residual	353	118.003	0.334		
Total	361	993.630			

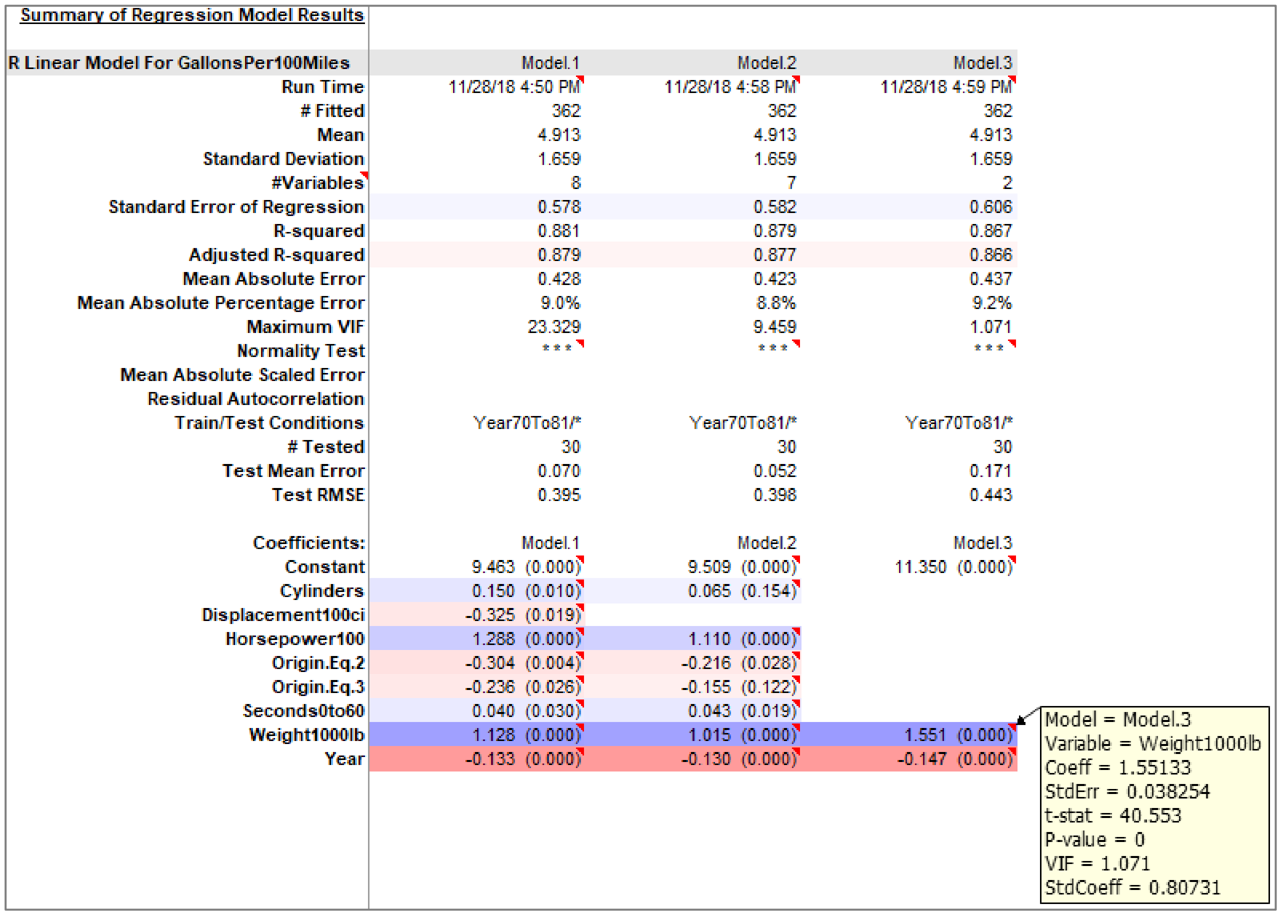
Error Distribution Statistics: Model.1 for GallonsPer100Miles (8 variables, 0 removed by forward stepwise, n= 362)

	Mean Error	RMSE	MAE	Min	Max	MAPE	A-D* stat
Fitted (n= 362)	0.000	0.571	0.428	-1.579	2.376	0.090	2.26 (0)
Tested (n= 30)	0.070	0.395	0.301	-1.155	0.794		

Out-of-sample test: fixed with training set Year70To81

Test RMSE/Train RMSE = 0.692 , Test StdDev/Train StdDev = 0.328

The **model summaries worksheet** in the Excel file provides side by side comparisons of model statistics. Color coding of coefficients by significance can be toggled on and off here as on the model output worksheets. Additional detail is stored within cells in the form of cell comments that pop up when the mouse is moved over them, as shown below. This table was produced from the output of three models fitted in RStudio. It looks exactly the same as for models fitted in RegressIt, except that it includes extra rows for out-of-sample test results. (The linear regression procedure within RegressIt does not do out-of-sample testing.)



Summary of Regression Model Results

R Linear Model For GallonsPer100Miles	Model.1	Model.2	Model.3
Run Time	11/28/18 4:50 PM	11/28/18 4:58 PM	11/28/18 4:59 PM
# Fitted	362	362	362
Mean	4.913	4.913	4.913
Standard Deviation	1.659	1.659	1.659
#Variables	8	7	2
Standard Error of Regression	0.578	0.582	0.606
R-squared	0.881	0.879	0.867
Adjusted R-squared	0.879	0.877	0.866
Mean Absolute Error	0.428	0.423	0.437
Mean Absolute Percentage Error	9.0%	8.8%	9.2%
Maximum VIF	23.329	9.459	1.071
Normality Test	***	***	***
Mean Absolute Scaled Error			
Residual Autocorrelation			
Train/Test Conditions	Year70To81/*	Year70To81/*	Year70To81/*
# Tested	30	30	30
Test Mean Error	0.070	0.052	0.171
Test RMSE	0.395	0.398	0.443
Coefficients:	Model.1	Model.2	Model.3
Constant	9.463 (0.000)	9.509 (0.000)	11.350 (0.000)
Cylinders	0.150 (0.010)	0.065 (0.154)	
Displacement100ci	-0.325 (0.019)		
Horsepower100	1.288 (0.000)	1.110 (0.000)	
Origin.Eq.2	-0.304 (0.004)	-0.216 (0.028)	
Origin.Eq.3	-0.236 (0.026)	-0.155 (0.122)	
Seconds0to60	0.040 (0.030)	0.043 (0.019)	
Weight1000lb	1.128 (0.000)	1.015 (0.000)	1.551 (0.000)
Year	-0.133 (0.000)	-0.130 (0.000)	-0.147 (0.000)

Model = Model.3
Variable = Weight1000lb
Coeff = 1.55133
StdErr = 0.038254
t-stat = 40.553
P-value = 0
VIF = 1.071
StdCoeff = 0.80731

The logistic regression option for R models will generate specialized outputs such as binary classification tables which are also included in output imported back to Excel. There is a separate version of RegressIt for the PC ("RegressItLogistic") which performs logistic regression in Excel with out-of-sample testing and interactive tables and charts.

More detailed instructions:

1. Launch Excel and RStudio.
2. In Excel, open the data file, and open the RegressIt program file if it has not been installed on the add-ins menu to load automatically.
3. Load the data into an R data frame if it is not already present in the RStudio session. To do this, hit the Export-data button on the RegressIt ribbon, click through a couple of prompts, then hit Ctrl-V and Enter at the command prompt in RStudio. What happens here is that RegressIt writes the contents of the Excel data worksheet to a CSV file whose default name is the sheet name and whose default location is the Excel file location, although other names and locations can be specified. (Before this step it is good to be thoughtful about naming the data worksheet in the Excel file, not just leaving it as "Sheet1".) This operation also places a line of R code on the clipboard for reading the CSV file into a data frame. If a CSV file of that name already exists, it is not overwritten, and only the code for reading it into a data frame is exported. This whole sequence of steps requires 4 mouse clicks or keystrokes.
4. Choose variables for the first regression model. Click the linear regression button [or the logistic regression button if present] on the RegressIt ribbon, select the dependent and independent variables, check the R-code-only box, and hit Run. This will open the dialog box for R code options which is pictured above.
5. RegressIt requires about a dozen packages to be loaded in RStudio before fitting the first model. To do this, click the Packages button in the R code dialog box and then hit Ctrl-V followed by Enter at the command prompt in RStudio. This only needs to be done once.
6. Select the model type (linear or logistic), the desired table and chart output, the variable selection method (all variables or forward or backward stepwise), and the testing options. The latter options include out-of-sample testing with a fixed training set and test set, k-fold cross-validation using either randomly determined folds or pre-selected folds, fitting of multiple models to disjoint subsets of the data with or without testing on the complementary sets, and fitting of multiple models to randomly selected subsets of a given size. When stepwise selection is used in connection with any of the options for multiple models, it is applied separately to each model. It is sometimes interesting to do this in conjunction with the random-subset option to see a demonstration of how the model identification can vary with the sample. (The results may be dramatic if there is some amount of multicollinearity.) The stepwise option will also potentially fit different models to different disjoint subsets, which may be appropriate if they are qualitatively different populations.
7. When plots are selected in the R code dialog box, they will be generated in the Plots window in RStudio. One of the options for linear models is the standard 2x2 chart array produced by the "plot" command. Usually only 2 of these are of interest: the plot of residuals versus predictions and the normal quantitle plot. RegressIt also provides 3 other plot options: actual and predicted values versus row number (with out-of-sample forecasts distinguished by different point types and colors), residuals versus row number, and residual autocorrelations. These options are especially important for time series data, but they are also useful more generally if there is any systematic sorting or grouping of the data by row or if it is of interest to find the location of unusual points. RegressIt automatically adds customized titles to all of the plots, which include the model name, dependent variable name, and some relevant statistics.

   Important note: at run time it is necessary for the plot window to be large enough to generate the requested plots, otherwise a "figure margins too large" error will appear in the plot window and an "invalid graphics state" error will appear at the command prompt. In some cases this error condition will also cause further code execution to fail. If so, type "dev.off()" at the command prompt to clear the error.

8. If the Export-model-summary-for-Excel option is chosen, the table output of the model will be written to the clipboard, formatted in exactly the same way as native RegressIt output. *After the model is fitted in R, you just need to go back to Excel and hit the Import-R button on the RegressIt ribbon to create a new worksheet, import the contents of the clipboard, format the tables in RegressIt style, and add a column of statistics to the Model Summaries worksheet.*
9. The Colors, Fonts, and Remove buttons work exactly the same way on output worksheets produced by R as they do on worksheets produced within RegressIt (interactive color coding by significance and interactive variable removal).
10. There are also options to export the tables of training set and test set results. They will appear below the summary output on the worksheet. Plots are not included by default when the Excel export option is chosen, however they can be easily copied and pasted to the Excel file in bitmap format by using the Export button above the Plots window.
11. If one of the multiple-subset options has been chosen, the default output that is exported to Excel is a summary table showing just a single line of statistics for each model, as well as their average across models. *If you want to export the complete results for each model to separate worksheets in the Excel file, check the box that says Pause-between-models-for-copying-output, and then hit the Import-R button on the RegressIt ribbon after each pause*.
12. One of the other R code options is a check-box and accompanying text field at the bottom of the screen for specifying the name of an existing model in the RStudio workspace. If this box is checked and a model name is entered, the variables of that model will be used for the new model to be run in RStudio, rather than the variables (if any) that were selected on the previous screen in Excel. If one of the stepwise variable selection options is chosen, the variables in the specified model will be used as the full model on which stepwise selection will be performed. This feature can be used to do various kinds of testing of an already-developed model as well as to produce prettier output in Excel for it.
13. The script files produced by RegressIt can be viewed by going to File/Open-file on the RStudio menu. Typically they contain quite a lot of code, although most of it is devoted to creating descriptive variables, building the customized outputs, and dealing with special cases such as empty test sets or missing values of dependent or independent variables. You can see the created objects in the Global Environment pane. Those which consist of arrays or models or data frames can be viewed in the Source pane by clicking on them. The existing variables are erased every time a new model is run, so that they are always in synch with the latest model. If you want to erase them all after you are finished with your regression analysis, hit the Clear-variables button in the R code dialog box and then hit Ctrl-V and Enter at the command prompt in RStudio.