Benchmarks Online

Skip Navigation Links

Page One

Campus Computing News

Summer Hours

Information Security Awareness

Get it while it lasts! UNT Support of SkillPort Computer-Based Training Ends November 2008

Today's Cartoon

RSS Matters

The Network Connection

Link of the Month

Helpdesk FYI

Short Courses

IRC News

Staff Activities

Subscribe to Benchmarks *Online*

Research and Statistical Support
University of North Texas

RSS Matters

Link to the last RSS article here: Mapping And Data Representation In Stata: Part 2 - Ed.

Getting Started with a Modern Approach to Regression

By Dr. Mike Clark, Research and Statistical Support Services Consultant

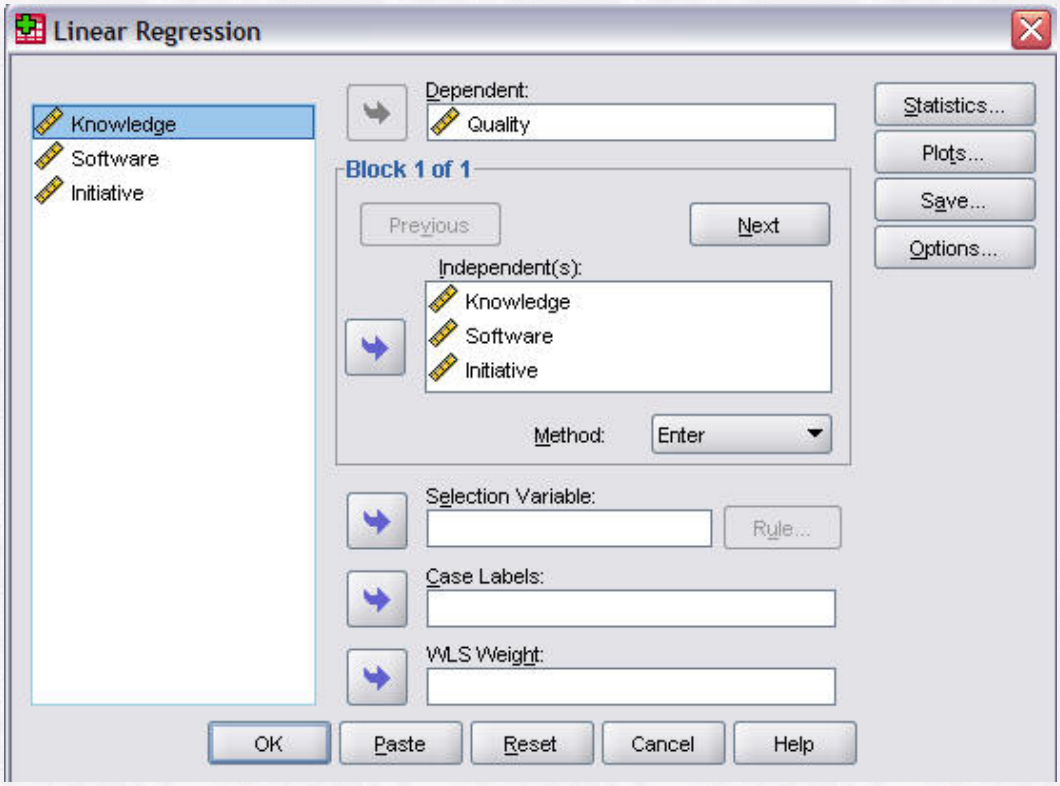
Multiple regression is one of the most widely employed statistical techniques in the social sciences, and unfortunately given its popularity, it is often poorly implemented as well. One can pretty much open any journal in which some study uses the technique and they will find all manner of issues from no mention of assumptions, poor attempts to ascertain whether assumptions have been met, no detail on outlier analysis and even if so, no usage of modern techniques to deal with it, poor approaches to determining variable importance, lack of validation etc. The purpose of this article is to provide a few examples of how one can deal with such issues, and the statistical package of choice will be R. For introductions on how to use it, start with the applications link here www.unt.edu/rss and click on R.

The issues

The reasons for such poor practice probably stem from two things primarily: the first is simple lack of knowledge regarding the underlying issues, and the second is using poor software for the analysis. Regarding the first, many believe typical analyses are 'robust' to even moderate violations of assumptions. Now if you ask them what *robust* actually means you may get a variety of answers, but some will answer “robust to type I error” and *in general* that is a correct assessment. However specifically the answer would be 'usually' robust to type I error, but not always, sometimes inflating and in other cases leading to a more stringent alpha level than one would want. Now what about type II error, retaining the null hypothesis when it should have been rejected? Typically disaster. Depending on the situation even slight deviations from normality, homoscedasticity etc. can destroy statistical power (a simple sim to play around with can be found here), which is often much more of a problem in our analyses even when data situations are not problematic (Cohen, 1992). Furthermore, violations of assumptions may lead to biased and inefficient estimates, making inference suspect at best and impossible at worst if corrective measures have not been taken. This suggests that not only should assumptions be tested and outcomes reported as a common practice[1], something should be done in the face of those problems as they may result in missed effects, biased and/or inefficient estimates, incorrect probabilities and in general 'bad things *man*'. Regarding the second issue I will use SPSS's menu system as an example as a. SPSS is very popular in social and other science research and b. I venture to wager that most applied social scientists that use SPSS do not use syntax unless absolutely necessary unless they are of the 'ol' school' as the kids say (and often the syntax does not even provide for any more options). Furthermore, as academic research reporting implies only a relative few are using any of the expensive add-ons, this will assume only the base package options.

If one looks at the menu options (Figure 1) it at first may appear there are many things to choose from, and while some of them are things we want, there are also redundant options (10+ measures of outlierness might be considered overkill by some but I personally like having as many as possible), poor options (e.g. pairwise deletion would lead to biased results), and several things that would only be feasible if doing a particular type of analysis (sequential or stepwise procedure). Its main dialog box is standard fare choosing of the model and a sequential or stepwise approach, the latter of which itself is fairly limited compared to other packages that provide subsets regression and keeping variables based on a variety of fit indices that correct for model complexity rather than just statistical significance. But as a reminder, we are doing 'simultaneous' regression here. In the statistics box one has access to some good things like interval estimates for coefficients, partial and semi-partial correlations (SPSS calls the latter 'part'), and collinearity diagnostics. The plots box might be fairly daunting to the uninitiated, and one could easily create uninformative graphs. The SPSS help file won't be of much use, since it tells of only a single graph one could make but not how to interpret it. But at least you can create the graphs, though other packages provide meaningful ones by default. Many things are in the 'save' box but you would not need but a few of them, e.g. saving the residuals and a couple outlier measures such as Cook's distance would suffice. The prediction intervals are nice for graphical display but that's best left to the graphics menu as you won't get such a graph here.

Figure 1.



So, although there are several dialog boxes and seemingly many options, some useful, we have run into several problems.

1. One can get plots to assess normality and homoscedasticity assumptions, but no actual means to test the latter at all. For the former we'd have to save the residuals (not done by default) then assess those.

2. One can get outlier measures but no automatic visual display of influence.

3. There is no method of validating the model available.

4. There is no means to do anything if you actually have any problems (and that goes for the Regression syntax).

5. Options for stepwise methods and dealing with missing values are severely limited.

In the end, if an applied researcher even knew about the statistical issues with linear regression, if their statistical package know-how was limited to base SPSS menus/Regression syntax, there isn't any viable means to address nor deal with data problems. In short, a *good* regression analysis is not possible without serious programming skills, and additional, possibly inadequately tested macros and scripts one might obtain from the web. A good regression is possible in R even with minimal knowledge of the language and menus, and quite easily, so we turn to it now.

Fitting the model

First off, the model. I have some made up data involving 4 variables: We'll call them *Knowledge* of statistical issues, Statistical *Software* expertise, and *Initiative*, with an outcome variable of *Quality* of analysis. The thing about regression is that we have to fit the model before we can test assumptions and search for outliers. For the following, relevant code will be in maroon, output in blue, comments in green. Links are provided to help documents for specific functions. We won't even need a specific package for fitting with the lm function, but we'll be some later.

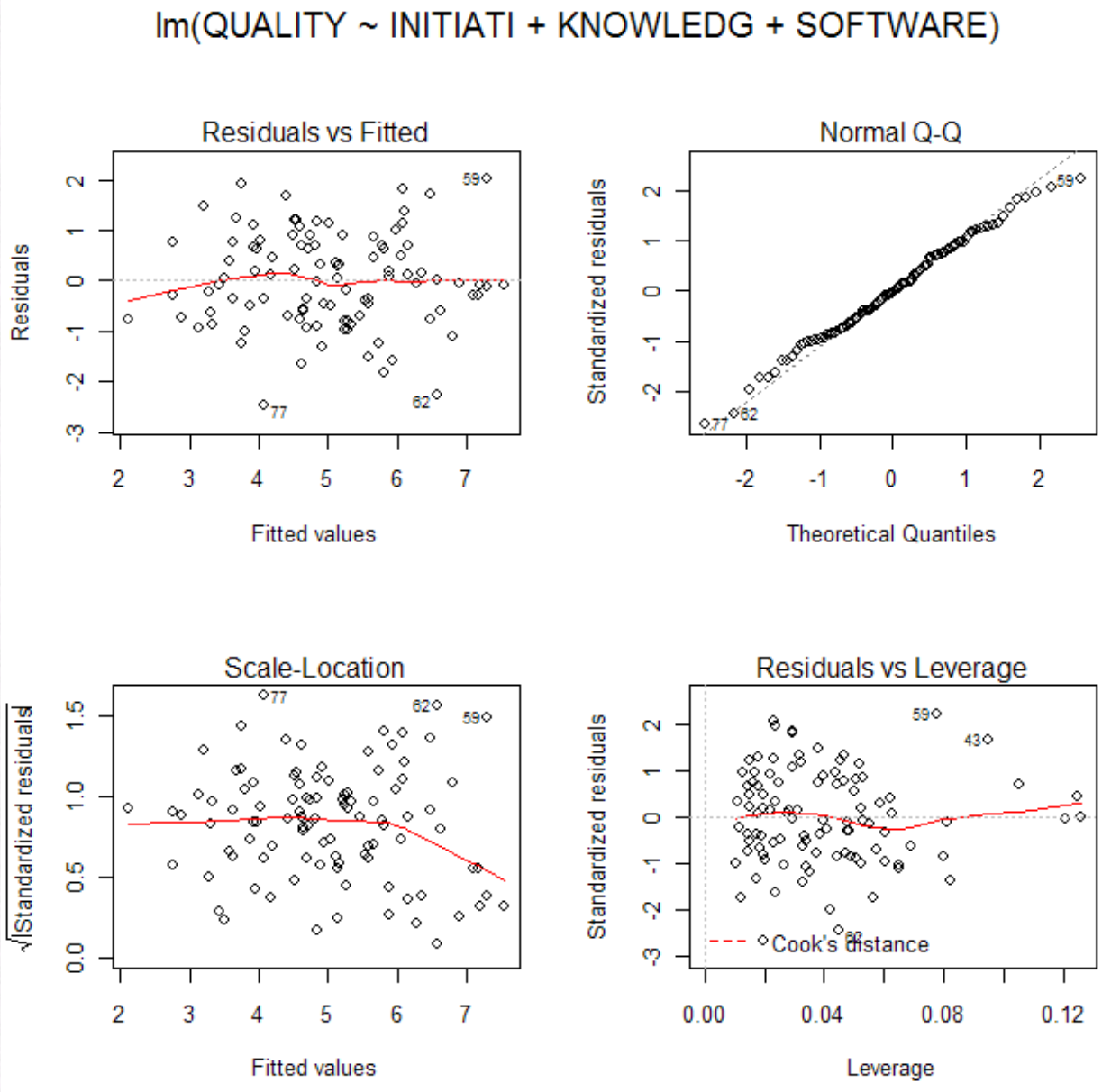
```
#the name model i.e. “Model” is arbitrarily chosen by me

Model <- lm(QUALITY~INITIATI+KNOWLEDG+SOFTWARE, data=Dataset)

summary(Model)


Call:

lm(formula = QUALITY ~ INITIATI + KNOWLEDG + SOFTWARE, data = Dataset)


Residuals:

    Min       1Q   Median       3Q      Max

-2.46385 -0.69550 -0.03513 0.69538 2.00725


Coefficients:

           Estimate Std. Error t value Pr(>|t|)

(Intercept) -2.06079    0.58588  -3.517 0.000667 ***

INITIATI     0.05661    0.01229   4.605 1.26e-05 ***

KNOWLEDG     0.07079    0.01111   6.372 6.46e-09 ***

SOFTWARE     0.01381    0.01265   1.091 0.277881

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.9441 on 96 degrees of freedom

Multiple R-squared: 0.6158, Adjusted R-squared: 0.6038

F-statistic: 51.29 on 3 and 96 DF, p-value: < 2.2e-16
```

At this point, if you are planning on doing a good regression analysis, you wouldn't bother to look at the outcome, except maybe to note if there is anything seriously wrong. However, being human, most of us will, so go ahead. This model does seem to be fitting well (which makes sense since I made it up to do so), but the point is that everything that could be interpreted at this point could be wrong, from p-values to coefficients. So lets test our assumptions so that we can feel confident in our interpretation. I can do this with the R-commander menu (type library(Rcmdr) at the command line to bring it up) but the code will be listed.

Diagnostics

The best way to start (and in my opinion, continue and end) is graphically, and graphs can be obtained through the *Models/Graphs/Basic diagnostic plots* menu, or simply typing plot(Model) at the command line.

Figure 2.



The Residuals vs. Fitted and Scale-Location are two versions of the same thing (the latter a more robust

of the former), and we are examining them with regard to two assumptions, linearity and homoscedasticity. The line should look like a child's freehand attempt at a straight line. The scatter of dots should look like the blob we see, no curvilinear pattern or fanning out on one or both ends. We're doing ok at this point for the most part, but we still will probably want to check the statistical assessment. The Normal Q-Q plot should also look pretty much as it is, with all the points lying on the indicated line (they typically start to get loose at the ends). Again, visual inspection would suggest we've met that assumption. The last graph regards outliers, but there is one I like better so I'll postpone discussion of it for now. First let's get actual statistical tests for the those graphs to back up what we see.

One can access the R-commander menus (Models/Numerical diagnostics/Breusch-Pagan...) but I will also provide the associated code. The following provides the Breusch-Pagan test for heteroscedasticity.

```
#As I've already got 'Model' specified, I don't need to put the formula.

library(lmtest)

bptest(Model, varformula = ~ fitted.values(Model), studentize=FALSE, data=Dataset)


Breusch-Pagan test

data: QUALITY ~ INITIATI + KNOWLEDG + SOFTWARE

BP = 0.1898, df = 1, p-value = 0.663
```

As with most tests of assumptions, statistical significance is not desired, but instead we will (illogically, actually) accept the null and go on our merry way. The following provides the Reset test for linearity and the Shapiro-Wilks test for normality (done on the residuals, which the normality assumption regards, not the predictors). The Reset test is in the same R-commander menu as the BP test, but while the Shapiro-Wilks test for normality is in the basic summaries menu, the residuals are not accessible as a variable there, and so testing that requires a visit to the command line. It is part of the base install of R and thus requires no additional package. A note for all you who have not been testing your multivariate normality assumption, there is mshapiro.test (and others) also that is just as easy to pull off.

```
#If you already had the lmtest library up, you would not need to call it again

     library(lmtest)      resettest(Model,      power=2:3,
type="regressor", data=Dataset)

      RESET test

 data:  QUALITY  ~  INITIATI  +  KNOWLEDG  +  SOFTWARE
RESET = 1.0169, df1 = 6, df2 = 90, p-value = 0.4195


shapiro.test(Model$residuals)

      Shapiro-Wilk normality test

 data:  Model$residuals  W  =
0.9919, p-value = 0.8165
```

Again, we're doing fine as our graphics suggested earlier. The following is the Durbin-Watson test, which can assess whether there is serial correlation among the residuals, i.e. it is a means to test if our observations are independent. While usually linearity stays a theoretical assumption regarding the manner of data collection and the DW test is mostly seen with time-series data, I go ahead and provide it here. Again, it is available via the R-commander menus if one wishes.

```
dwtest(Model, alternative="two.sided", data=Dataset)

data: QUALITY ~ INITIATI + KNOWLDEG + SOFTWARE DW =
 1.8803,  p-value  =  0.5453  alternative  hypothesis:
true autocorelation is not 0
```

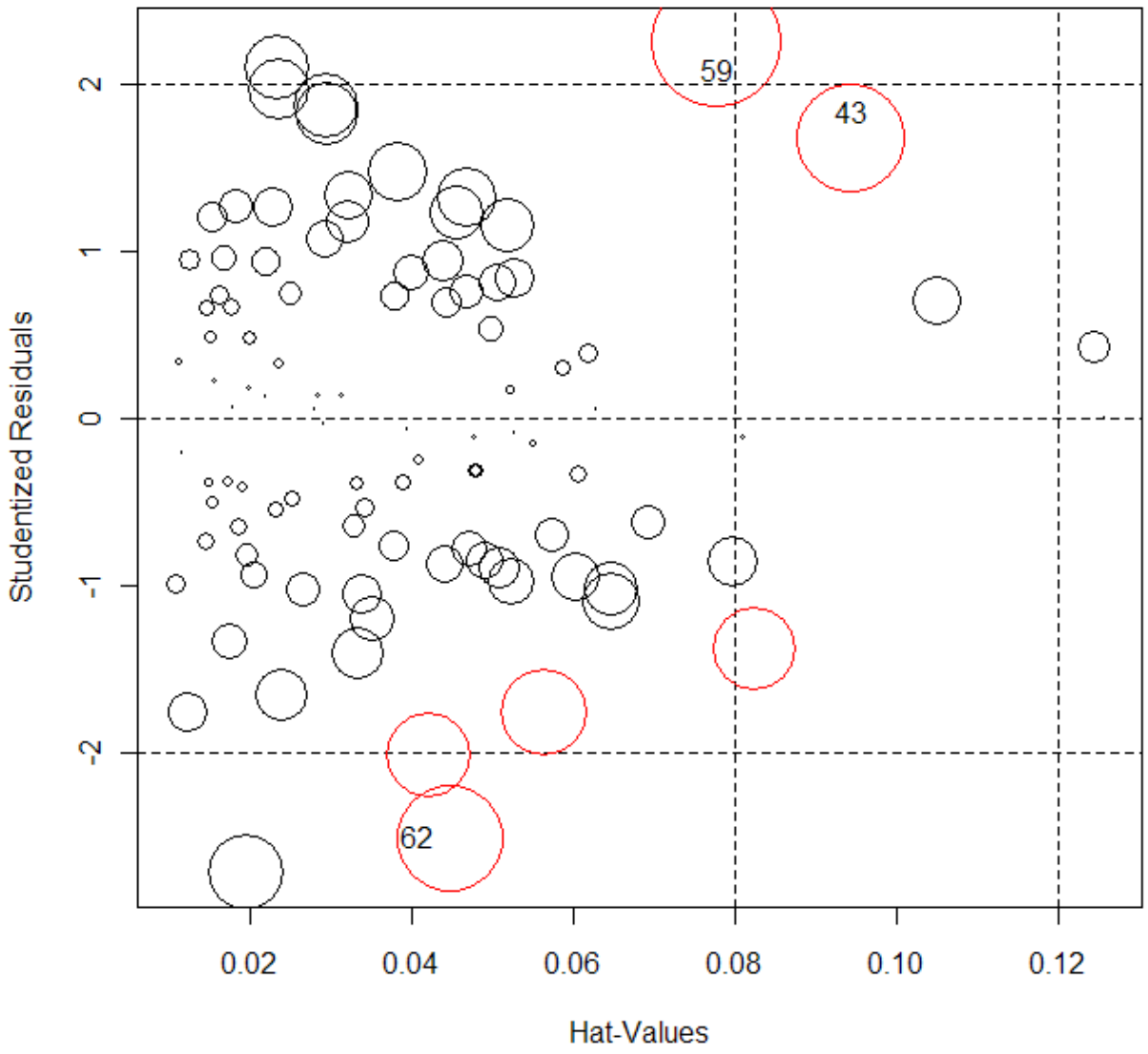
No problems there. We'll next assess multicollinearity, which is essentially redundant information

among the variables, such that one variable's variance is largely shared by the others. A way to test this is to simply regress the variable in question on the others and note the R^2 value. *Tolerance,* as given by SPSS for example, is 1- that R^2 just mentioned, and the *variance inflation factor* is the inverse of tolerance. Usually, if the VIF is in the neighborhood of 10, we might be concerned, as it would suggest that the vast majority of the variable's variance is accounted for by the other variables, and may lead to inefficient parameter estimates.

```
vif(Model)

INITIATI KNOWLDEG SOFTWARE
1.678325 1.370830 1.778163
```

Still doing well. Now back to that influence plot to help us assess potential outlying cases, which I obtained through that same Models/Graphs/ menu earlier or can get with influencePlot(Model) from the car package. The X axis is the leverage statistic, which is a measure of influence, the Y axis is the studentized residual value, and the size of the bubble reflects the case's Cook's distance. In this manner it provides three measures of 'outlierness' in one 2-d graph. Vertical reference lines are drawn at twice and three times the average hat value, horizontal reference lines at -2, 0, and 2 on the studentizedresidual scale. Case 59 for example, may be of concern. In any case we were going to run a robust version of regression for comparison anyway.

Figure 3.



Robust regression check

There are many ways in which to run a regression which is resistant to outliers and often performs better in heteroscedastic situations, and more are being developed (see Wilcox, 2001 for an introduction). Some are simple in concept, e.g. Least-trimmed squares regression, while others, for example those based on M-estimators, can get more technical. Here I follow Tukey's guideline:

> "… just which robust/resistant methods you use is not important – what is important is that you use some. It is perfectly proper to use both classical and robust/resistant methods routinely, and only worry when they differ enough to matter. But when they differ, you should think hard." (Tukey, 1979)

So for me, the thing to do is simply check, and R has whole packages devoted to robust techniques. I will use the 'robustbase' package for this example.

```
#modelrob is again an arbitrary name

library(robustbase)
```

```
modelrob=lmrob(Model)
summary(modelrob)

#Partial output
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.04399    0.56972  -3.588 0.000527 ***
INITIATI     0.05526    0.01195   4.622 1.18e-05 ***
KNOWLEDG     0.07157    0.01045   6.851 6.99e-10 ***
SOFTWARE     0.01415    0.01253   1.130 0.261397
---
Signif. Codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Robust residual standard error: 0.9919
```

Comparison to our previous output suggests hardly any change in coefficients, which, given the outcome of our previous diagnostic checks, was to be expected. Squaring the correlation of the robust fitted values to the Quality DV gave almost the same R^2 as the original. It is safe to assume at this point that our least squares estimates are okay and worth keeping.

Validating the model and adjusting R^2

Some may see validation for stepwise exploratory endeavors though sadly often not even there, which renders them completely useless analyses in my opinion. However the point is that validation can and should be performed for *any* regression analysis that has an adequate sample size. Many are familiar with simple validation techniques, such as cross-validation using training and test sets. However sample sizes that are needed may be prohibitive with such approaches, and many others are not available in some standard statistical packages. The bootstrapping technique will allow us, again assuming you have a reasonable sample for the model in question to begin with, to use the cross validation technique multiple times by resampling (with replacement) from the original data set to create even several hundred training and test sets for validation. Details can be found in Harrell (2001), where further original references to Efron and others may be found also (Harrell created the Design package used here). In short, you are doing many, cross-validations, and getting an average estimate of bias in your R^2 metric.

```
library(Design)
valmodel=ols(formula=QUALITY~INITIATI+KNOWLEDG+SOFTWARE, data=Dataset, x=T, y=T)
validate(valmodel, method="boot", B=500)


Iteration 500
          index.orig  training       test     optimism index.corrected   n
R-square   0.6158219 0.6188035 0.59984122  0.018962291      0.59685961 500
MSE        0.8557567 0.8286495 0.89135369 -0.062704177      0.91846090 500
Intercept  0.0000000 0.0000000 0.01487108 -0.014871076      0.01487108 500
Slope      1.0000000 1.0000000 0.99780735  0.002192654      0.99780735 500
```

The key statistic for our purposes regards the first row. Going across we see our original variance

accounted for, the average training set R^2, the average test set R^2, the optimism, which is the difference between training and test, and finally the corrected version of R^2 which is the optimism subtracted from the original. This serves as our bias-adjusted R square which is based on the specifics of the dataset rather than a heuristic as the typical "Adjusted R^2" reported is.

Confidence Interval for the R^2

Like a mean or anything else we are estimating in a sample, point estimates do not allow us to get an estimate of the uncertainty in our guess, and effect sizes vary from sample to sample as do other statistics. To obtain that understanding of uncertainty we will construct a confidence interval for this R^2 using the MBESS package and basic information about the model such as the sample size and number of predictors as follows. Note that I have input the bias-corrected R^2.

```
 library(MBESS)

ci.R2(R2=.597, N=100, K=3, conf.level=.95)

$Lower.Conf.Limit.R2
[1] 0.4475404

$Prob.Less.Lower
[1] 0.025

$Upper.Conf.Limit.R2
[1] 0.6996761

$Prob.Greater.Upper
[1] 0.025
```

So the 95% CI for the bias-adjusted R^2 is .448 to .700.

Which variable is most important?

Satisfied with the model's integrity at this point, we might want to now take an explanatory approach to determine variable importance. With scales of different measures, raw coefficients can't provide this information by themselves, so we'll have to use something else. Many use standardized coefficients for this purpose, and if one is larger than another they claim it is more important. This is unsatisfactory, and equivalent to saying that just because one mean being higher than another entails it is statistically/meaningfully so. Given sampling variability, orderings that are not far apart have the potential to reorganize their ranks upon the next sample collected. The following uses a bootstrap approach to create confidence intervals around metrics of importance so that one can demonstrate statistical differences among them. The statistic of choice for me is the average semi-partial (Lindeman, Merenda, & Gold, 1980; Kruskal, 1987), which measures a variable's squared semi-partial at all possible points of entry into the model, and takes the average of those. It has the added bonus of decomposing R^2 into each variable's contribution to it. Furthermore, the relaimpo package will allow us to determine whether their contributions are statistically distinct.

```
library(relaimpo)

calc.relimp(Model)

Proportion of variance explained by model: 61.58%

Metrics are not normalized (rela=FALSE).

Relative importance metrics:

lmg

INITIATI 0.2138923

KNOWLEDG 0.2800459

SOFTWARE 0.1218837
```

These are the average semi-partials, called LMG after for the authors of the first known source of the

method. There are other options, e.g. Adding rela= TRUE would normalize them to sum to 100%, and each value would be the percentage of R^2 it accounts for. This may be preferable since emphasis would be put toward a bias corrected R^2 for the model. As it is above, they sum to the model's original R^2. Knowledge of statistical techniques *seems* to be most important in the model, but let's find out for sure.

The following will provide the confidence intervals for the above metrics as well as confidence intervals for the *difference between them*, and so provides a statistical test at a specified alpha level.

```
#lmgci is the arbitrary object name

lmgci=boot.relimp(Model, boot = 1000)

booteval.relimp(lmgci)


#Partial output

Confidence interval information ( 1000 bootstrap replicates, bty= perc ):

Relative Contributions with confidence intervals:


                           Lower  Upper

            percentage  0.95 0.95   0.95

INITIATI.lmg 0.2139      ABC  0.1279 0.3143

KNOWLEDG.lmg 0.2800      AB_  0.1980 0.3742

SOFTWARE.lmg 0.1219      _BC  0.0624 0.1972


Letters indicate the ranks covered by bootstrap CIs.

(Rank bootstrap confidence intervals always obtained by percentile method)

CAUTION: Bootstrap confidence intervals can be somewhat liberal.

Differences between Relative Contributions:


                                    Lower  Upper

                     difference 0.95 0.95   0.95

INITIATI-KNOWLEDG.lmg -0.0662          -0.2335 0.0907

INITIATI-SOFTWARE.lmg  0.0920          -0.0258 0.2088

KNOWLEDG-SOFTWARE.lmg  0.1582       *   0.0440 0.2765

*indicates that CI for difference does not include 0.
```

The output first provides 95% CIs for the metrics themselves, while the second bit provides them for the difference between any two LMG statistics. Standard reporting based on standardized coefficients would have ranked them as knowledge first, initiative second, and software expertise as even being nonsignificant. However here we can see the low end of the software variable suggests a meaningful contribution (about 10% of our R^2), and the only difference among their orderings we might feel confident enough to make would be between statistical knowledge and software expertise, the former contributing more than the latter.

Summary

While it may have seemed quite a bit to pull off, assuming we didn't use any menus the entire code is 15 lines including summary calls and takes only a few seconds to obtain *all* of the output. As a

comparison, for basic regression in SPSS it would have taken many more lines to produce as much as it could of the R output, but it simply would not be able to do most of it. So technically with less work one can perform real tests of assumptions, a robust check on the analysis, validation of the model, an interval estimate of the bias-corrected R^2, and obtain tests for a variable importance metric that decomposes R^2.

Here I did not do many of the options that were available for each function, and as was noted, this is 'pretty' data which is rare in the social sciences. But even with this minimal approach I was able to pull off an analysis that was just as or more interpretable and more accurate one than a standard one would have been, and one in which I can feel much more confidence regarding the results.

Given modern desktop computing capabilities, the time has long since passed to still be using methods without regard to the developments of the past 30 years. Standard texts have always pointed out the issues, but now even some introductory ones provide solutions, the implementation of which are found in many statistical packages, some of which, like R, are free. Just concluding 'caution must be taken in the interpretation of results' is a poor way of dealing with the problems of data. You could say that for any analysis despite the techniques employed including these. What would be a more accurate statement for those that do not use modern methods to get the sentiment across would be 'these results are completely suspect because care was not taken to perform the analysis well'.

Hopefully those who read this can see how easy it can be to pull off, and begin their journey to doing better statistical analyses.

Summary of code

```
#red lines could have been done via R-commander menus, assumes data has already been imported
 and called 'Dataset'

library(Rcmdr) #brings up the menu system

library(lmtest) #diagnostics

library(robustbase) #robust regression

library(Design) #validation, bias assessment

library(MBESS) #CI for the R2

library(relaimpo) #variable importance



Model <- lm(QUALITY~INITIATI+KNOWLEDG+SOFTWARE, data=Dataset)

summary(Model)

bptest(Model, varformula = ~ fitted.values(Model), studentize=FALSE, data=Dataset)

resettest(Model, power=2:3, type="regressor", data=Dataset)

shapiro.test(Model$residuals)

dwtest(Model,alternative="two.sided",data=Dataset)

vif(Model)

modelrob=lmrob(Model)

summary(modelrob)

valmodel=ols(formula=QUALITY~INITIATI+KNOWLEDG+SOFTWARE, data=Dataset, x=T, y=T)

validate(valmodel, method="boot", B=500)

ci.R2(R2=.597, N=100, K=3, conf.level=.95, Random.Predictors=TRUE)

calc.relimp(Model)
```

```
lmgci=boot.relimp(Model, boot = 1000)

booteval.relimp(lmgci)
```

Footnotes

[1] I have at times actually stopped reading research reports that don't even bother to mention anything about the testing of assumptions, since I can never know if the results are what they say they are, much less any theoretical conclusions based on them.

References

Cohen, J. (1992). A power primer. *Psychological Bulletin, 112,* 155-159.

Harrel, F.E. (2001). *Regression Modeling Strategies.*

Lindeman, R.H., Merenda, P.F. and Gold, R.Z. (1980). *Introduction to Bivariate and Multivariate Analysis.*

Kruskal, W. (1987). Relative importance by averaging over orderings. *The American Statistician,41*, 6-10.

Tukey, J.W. (1979). “Robust techniques for the user". In R.L. Launer & G.N. Wilkinson. (Eds.), *Robustness in Statistics*.

Wilcox, R. (2003). *Applying Contemporary Statistical Techniques.*

Wilcox, R. (2002). *Fundamentals of Modern Statistical Methods: Substantially Improving Power and Accuracy.*

Wilcox, R. (1997). *Introduction to Robust Estimation and Hypothesis Testing.*

Return to top