Using rockchalk for Regression Presentations

Paul Johnson

June 18, 2012

1 Introduction

The rockchalk package is a collection of functions that I need, or my students might need, when I'm teaching about regression. The functions here divide into three categories.

1. Functions that help me prepare lectures and reports. The function to create LaTeX tables from regression output, outreg, falls into this category. It speeds up the preparation of lectures immensely to include table generating code that "just works" with R output. Some functions in R are very hard to use and get right consistently, especially where 3 dimensional plotting is concerned. That's where functions like mcGraph1, mcGraph2, mcGraph3, and plotPlane come in handy. These don't do any work that is particularly original, but they do help to easily make the multidimensional plots that turn out "about right" most of the time.

2. Functions simplify vital chores that are difficult for regression students. I keep track of the R idioms that bother the students and try to craft functions that simplify them. I have often asked students to plot several regression lines, "one for each sub-group of respondents," and this sometimes proves frustrating. The function plotSlopes is offered as my suggestion for creating interaction plots of "simple slopes". This handles the work of calculating predicted values and drawing them for several possible values of a third variable. plotPlane is along the same line. If students find that useful, they can then use the examples to build up more complicated drawings.

3. Functions that people often ask for, even if they might be unwise to use them. A function to estimate a "standardized regression" is offered. Although that is clearly unwise (in the eyes of many), some folks still want to calculate "beta weights." Some functions, such as meanCenter and residualCenter, are offered not because I need those tools, but because other people propose them to the students. Those procedures are, possibly, not truly helpful and in order to demonstrate that fact, I have to provide the functions.

2 Facilitating Collection of Summary Information

2.1 summarize: A replacement for summary

R's summary function has some limitations. For example, no indicators of diversity are included in the presentation. I'd like to see the standard deviation and/or the variance. For categorical variables, I'd like an indicator like entropy, which, roughly speaking, is a way of summarizing the chances that two observations (randomly drawn) do not fall into the same category.

As work progressed on a revision of summary, I realized there are other things about R's summary that I wanted to change. First, I want to separate the numeric from the categorical (factor) variables. Second, I want the option to have alphabetized columns in the output. Third, I want more workable objects to be returned. I find it almost impossible to use the returned value from summary for further analysis. It offers a large block of text in a table format.

In order to create more workable output objects, some re-arrangement of the work behind the scenes can be done. Basically, it is necessary that the summarize function return the more-workable, less easily printable output object, and then a print method is used to "pretty print" that output for the user.

The summarize function returns a list with two components, "numerics" and "factors". The numeric summary is a data frame, with named rows, that can be used for further analysis. Users who wish to summarize only the numeric variables can run summarizeNumerics instead, while others who want to summarize only factors can run summarizeFactors. The output from summarizeFactors is a list of factor summaries.

Consider the results of applying the summarize function to Fox's Chile data set from the car package:

```
data(Chile)
(summChile <- summarize(Chile))
```

```
$numerics
          age    income population  statusquo
0%      18.00 2.500e+03  3.750e+03 -1.803e+00
25%     26.00 7.500e+03  2.500e+04 -1.002e+00
50%     36.00 1.500e+04  1.750e+05 -4.558e-02
75%     49.00 3.500e+04  2.500e+05  9.686e-01
100%    70.00 2.000e+05  2.500e+05  2.049e+00
mean    38.55 3.388e+04  1.522e+05 -1.118e-08
sd      14.76 3.950e+04  1.022e+05  1.000e+00
var    217.80 1.560e+09  1.044e+10  1.000e+00
NA's     1.00 9.800e+01  0.000e+00  1.700e+01
N     2700.00 2.700e+03  2.700e+03  2.700e+03

$factors
          education                   region                     sex
 S            :1120.0000   SA            : 960.0000   F            :1379.0000
 P            :1107.0000   S             : 718.0000   M            :1321.0000
 PS           : 462.0000   C             : 600.0000   NA's         :   0.0000
 NA's         :  11.0000   N             : 322.0000   entropy      :   0.9997
 entropy      :   1.4900   M             : 100.0000   normedEntropy:   0.9997
 normedEntropy:   0.9401   NA's          :   0.0000   N            :2700.0000
 N            :2700.0000   entropy       :   2.0628
                           normedEntropy:   0.8884
                           N             :2700.0000
             vote
 N            : 889.0000
 Y            : 868.0000
 U            : 588.0000
 A            : 187.0000
 NA's         : 168.0000
 entropy      :   1.8264
 normedEntropy:   0.9132
 N            :2700.0000
```

A companion function is centralValues, which will provide only one number for each variable in a data frame. For numerics, it returns the mean, while for factor variables, it returns the mode.

```
centralValues(Chile)
```

```
  region population sex      age education   income     statusquo vote
1     SA   152222.2   F 38.54872         S 33875.86 -1.118151e-08    N
```

2.2 Easier newdata objects for predict

These two functions greatly facilitate the creation of "newdata" objects for use with R's predict methods. This code fits a regression model on the Chile data and then it extracts the model frame for further analysis.

```
m1 <- lm(statusquo ~ age + income + population + region + sex, data=Chile)
m1mf <- model.frame(m1)
```

If we were to run

```
predict(m1)
```

we would receive a predicted value for each observation. We want a smaller set of calculations, just predictions for a few cases we find interesting. In R, one should create a "newdata" data frame, which can then be used in a command like

```
predict(m1, newdata=myNewDF)
```

The newdata must include one column for each variable in the model, and those variables must have exactly the same names as were used in the regression formula. This makes it difficult for students to get the predictions they want without learning a lot about variable management in R.

I have sought to streamline this by developing a "can't miss" sequence of steps. My preferred approach is as follows. First, create a new data frame that sets the predictors at their central values (mean or mode). Then specify some alternate levels for particular predictors in a new data frame, and then combine those special interest variables with the central values data set.

In the context of model m1, suppose we want to formulate predictions for each of the 3 middle quantiles of the age variable and for the levels of region called "C" and "N". The summarize function and the centralValues functions are run first to collect up the needed information.

```
m1mfsumm <- summarize(m1mf)
m1cv <- centralValues(m1mf)
```

Use R's expand.grid function to create, mixAndMatch, a mix-and-match data frame of all combinations of the values for which we want to calculate predictions.

```
mixAndMatch <- expand.grid(age = summChile$numerics[2:4, "age"], region = c("C","N"))
mixAndMatch
```

```
  age region
1  26      C
2  36      C
3  49      C
4  26      N
5  36      N
6  49      N
```

There are many ways to put together the interesting combinations in the mixAndMatch data frame with the central values in m1cv. There is a danger that we may end up with columns that have duplicate names (there is a variable "age" in both m1cv and mixAndMatch). To avoid duplicate column names, I keep only the columns from m1cv that are not already in mixAndMatch when I join the two together.

```
mynewdf <- cbind(mixAndMatch, m1cv[ ,!colnames(m1cv) %in% colnames(mixAndMatch)])
mynewdf
```

```
  age region     statusquo   income population sex
1  26      C -0.008720772 33868.73     151750   F
2  36      C -0.008720772 33868.73     151750   F
3  49      C -0.008720772 33868.73     151750   F
4  26      N -0.008720772 33868.73     151750   F
5  36      N -0.008720772 33868.73     151750   F
6  49      N -0.008720772 33868.73     151750   F
```

The predicted values are called fit and added to mynewdf.

```
mynewdf$fit <- predict(m1, newdata = mynewdf)
mynewdf
```

```
  age region     statusquo   income population sex         fit
1  26      C -0.008720772 33868.73     151750   F -0.16296485
2  36      C -0.008720772 33868.73     151750   F -0.07867231
3  49      C -0.008720772 33868.73     151750   F  0.03090800
4  26      N -0.008720772 33868.73     151750   F  0.05611256
5  36      N -0.008720772 33868.73     151750   F  0.14040510
6  49      N -0.008720772 33868.73     151750   F  0.24998541
```

If one desires confidence intervals, the procedure is similar, although the data management is slightly more tedious.

```
mynewdf <- cbind(mixAndMatch, m1cv[ ,!colnames(m1cv) %in% colnames(mixAndMatch)])
preds <- predict(m1, newdata = mynewdf, interval="confidence")
mynewdf <- cbind(mynewdf, preds)
mynewdf
```

```
  age region     statusquo   income population sex         fit         lwr
1  26      C -0.008720772 33868.73     151750   F -0.16296485 -0.25825809
2  36      C -0.008720772 33868.73     151750   F -0.07867231 -0.16924382
3  49      C -0.008720772 33868.73     151750   F  0.03090800 -0.06387181
4  26      N -0.008720772 33868.73     151750   F  0.05611256 -0.06156769
5  36      N -0.008720772 33868.73     151750   F  0.14040510  0.02679847
6  49      N -0.008720772 33868.73     151750   F  0.24998541  0.13335550
         upr
1 -0.06767162
2  0.01189920
3  0.12568780
4  0.17379281
5  0.25401173
6  0.36661531
```

Table 1: My One Tightly Printed Regression

	Estimate (S.E.)
(Intercept)	40.224*
	(0.723)
x1	-2.364*
	(0.715)
N	100
$RMSE$	7.141
R^2	0.1
adj R^2	0.091

* $p \leq 0.05$

3 Better Regression Tables: Some outreg Examples.

On May 8, 2006, Dave Armstrong, a political science PhD student at University of Maryland, posted a code snippet in r-help that demonstrated one way to use the "cat" function from R to write LaTeX markup. That gave me the idea to write a LaTeX output scheme that would help create some nice looking term and research papers. I'd been frustrated with the LaTeX output from other R functions. I needed a table-maker to include all of the required information in a regression table without including a lot of chaff (in my opinion). I don't want both the standard error of b and the t value, I never want p values, I need stars for the significant variables, and I want a minimally sufficient set of summary statistics. In 2006, there was no function that met those needs.

Tables 1 through 6 present examples of table output that I am able to generate with outreg. The regression models are fit with some simulated data.

```
set.seed(1234)
x1 <- rnorm(100)
x2 <- rnorm(100, m=10)
x3 <- rnorm(100)
y1 <- 5*rnorm(100) - 3*x1 + 4*x2
y2 <- rnorm(100)+5*x2
dat <- data.frame(x1, x2, x3, y1, y2)
rm (x1, x2, y1, y2)
m1 <- lm (y1~x1, data=dat)
m2 <- lm (y1~x2, data=dat)
m3 <- lm (y1 ~ x1 + x2, data=dat)
myilogit <- function(x) exp(x)/(1 + exp(x))
dat$y3 <- rbinom(100, size=1, p=myilogit(scale(dat$y1)))
gm1 <- glm(y3~x1 + x2, data=dat)
dat$y4 <- 1 + 0.1 * dat$x1 - 6.9 * dat$x2 + 0.5 * dat$x1*dat$x2 + 0.2 * dat$x3 + rnorm(100,0, sd
    =10)
```

Table 1 displays the default outreg output, without any special options. The command is

```
outreg(m1)
```

In the literature, regression tables are sometimes presented in a tight column format, with the estimates of the coefficients and standard errors "stacked up" to allow multiple models side by side, while sometimes they are printed with separate columns for the coefficients and standard errors. The outreg option tight=F provides the two column style. In Table 2, I've also used the argument modelLabels to insert the word "Fingers" above the regression model. The command that produces the table is

```
outreg(m1, tight=FALSE, modelLabels=c("Fingers"))
```

The outreg function can present different models in a single table, as we see in Table 3. The default output uses the tight format, so there is no need to specify that explicitly. In part (a) of Table 3, we have tightly formatted columns of regression output that result from this command:

```
outreg(list(m1,m2), modelLabels=c("Mine","Yours"), varLabels = list(x1="Billie"))
```

Table 2: My Spread Out Regressions

	Fingers	
	Estimate	(S.E.)
(Intercept)	40.224*	(0.723)
x1	-2.364*	(0.715)
N	100	
$RMSE$	7.141	
R^2	0.1	
adj R^2	0.091	

* $p \leq 0.05$

Table 3: My Two Linear Regressions Tightly Printed

(a) Tightly Formatted Columns

	Mine Estimate (S.E.)	Yours Estimate (S.E.)
(Intercept)	40.224* (0.723)	-7.687 (5.542)
Billie	-2.364* (0.715)	.
x2	.	4.808* (0.549)
N	100	100
$RMSE$	7.141	5.639
R^2	0.1	0.439
adj R^2	0.091	0.433

* $p \leq 0.05$

(b) Two Columns Per Regression Model

	Mine		Yours	
	Estimate	(S.E.)	Estimate	(S.E.)
(Intercept)	40.224*	(0.723)	-7.687	(5.542)
Billie	-2.364*	(0.715)	.	
x2	.		4.808*	(0.549)
N	100		100	
$RMSE$	7.141		5.639	
R^2	0.1		0.439	
adj R^2	0.091		0.433	

* $p \leq 0.05$

Table 4: My Three Linear Regressions in a Tight Format

	A Estimate (S.E.)	B Estimate (S.E.)	C Estimate (S.E.)
(Intercept)	40.224* (0.723)	-7.687 (5.542)	-7.482 (5.104)
I Forgot x1	-2.364* (0.715)	. 	-2.24* (0.52)
He Remembered x2	. 	4.808* (0.549)	4.753* (0.506)
N	100	100	100
$RMSE$	7.141	5.639	5.193
R^2	0.1	0.439	0.529
adj R^2	0.091	0.433	0.519

* $p \leq 0.05$

Table 5: Three Regressions in the Spread out Format

	I Love love love really long titles		Hate Long		Medium	
	Estimate	(S.E.)	Estimate	(S.E.)	Estimate	(S.E.)
(Intercept)	40.224*	(0.723)	-7.687	(5.542)	-7.482	(5.104)
x1	-2.364*	(0.715)	.		-2.24*	(0.52)
x2	.		4.808*	(0.549)	4.753*	(0.506)
N	100		100		100	
$RMSE$	7.141		5.639		5.193	
R^2	0.1		0.439		0.529	
adj R^2	0.091		0.433		0.519	

* $p \leq 0.05$

To my eye, there is something pleasant about the less-tightly-packed format, as illustrated in part (b) of Table 3. Note that the only difference in the commands that produce those tables is the insertion of tight=FALSE.

```
outreg(list(m1,m2), tight=FALSE,  modelLabels=c("Mine","Yours"), varLabels = list(x1="Billie"))
```

In addition to using modelLables to provide headings for the 2 models, the other argument that was used in Table is 3 varLabels. It is often a problem that the variable names are terse, while a presentation must have a full name. So in Table 3, I've demonstrated how to replace the variable name x1 with the word "Billie". Any of the predictor variables can be re-named in this way. Another usage of varLabels is offered in an example with three models in Table 4, which is a result of

```
outreg(list(m1,m2,m3), modelLabels=c("A","B","C"), varLabels = list(x1="I Forgot x1", x2="He
    Remembered x2"))
```

As one can see, outreg gracefully handles the situation in which variables are inserted or removed from a fitted model.

I have not bothered too much with some fine points of LaTeX table formatting. In order to produce tables that are completely ready for publication in a journal, it would be necessary to use some special LaTeX packages to control the vertical alignment of columns and such. Doing so would make outreg more difficult for researchers to use, and I believe the benefit would be minimal. In Table 5, we have regression output which is, in my opinion, completely acceptable for inclusion in a presentation or conference paper. There are some warts: because the model labels are not equal in length, the columns are not equally sized.

Another feature of outreg is that it can present the estimates of different kinds of models. It can present

Table 6: Combined OLS and GLM Estimates

	OLS:y1 Estimate (S.E.)	GLM: Categorized y1 Estimate (S.E.)
(Intercept)	40.224* (0.723)	-1.232* (0.466)
x1	-2.364* (0.715)	0.031 (0.047)
x2	.	0.172* (0.046)
N	100	100
$RMSE$	7.141	
R^2	0.1	
adj R^2	0.091	
$Deviance$		21.799
$-2LLR(Model\chi^2)$		3.191

* $p \leq 0.05$

the estimates from R's lm and glm functions in a single table. Consider Table 6, which resulted from the command

```
outreg(list(m1,gm1),modelLabels=c("OLS:y1","GLM: Categorized y1"))
```

In the future, outreg will be enhanced to handle more types of regression models, including mixed (hierarchical) models.

4 plotSlopes, plotPlane and plotCurves

4.1 plotSlopes for linear models with moderator variables

Suppose the fitted model includes several variables,

$$\hat{y}_i = \hat{b}_0 + \hat{b}_1 x1_i + \hat{b}_2 x2_i + \hat{b}_3 x1_i x2_i + \hat{b}_4 x3_i. \quad (1)$$

We would like to visualize the effect of $x1$ on y for several values of $x2$, keeping $x3_i$ set at some reference value.

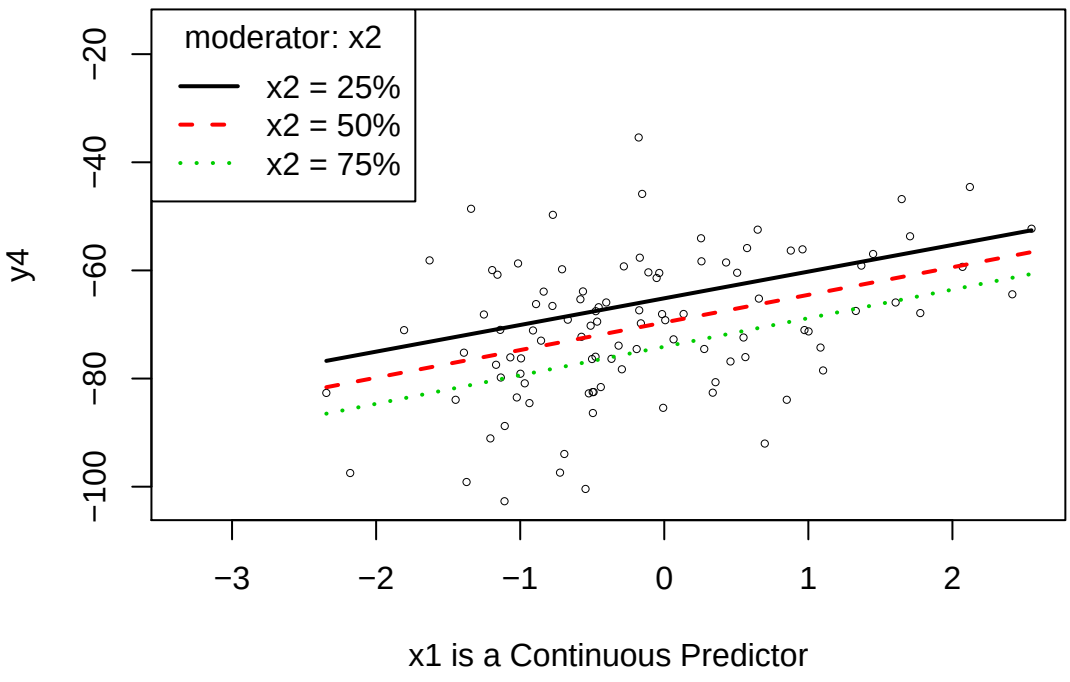
The plotSlopes function assists with that project. First, we fit a regression model, and then we pass that object to plotSlopes. Suppose we estimate the model in equation 1. Then use plotSlopes to illustrate the effect of $x1$ as it depends on $x2$.

```
m4 <- lm (y1 ~ x1*x2 + x3, data=dat)
plotSlopes(m4, plotx="x1", modx="x2", xlab="x1 is a Continuous Predictor")
```

The plotx argument is variable x1, meaning that x1 will be on the horizontal axis, and x2 serves as the moderator variable. plotSlopes requires that the plotx argument must be the name of a numeric variable, but modx may be the name of either a numeric or a factor variable.

When modx is a numeric variable, then some particular values must be selected for calculation of predictive lines. By default, three hypothetical values of plotx are selected (the quantiles 25%, 50%, and 75%). Any other variables in the model are set at central values, which would be the mean for a numeric variable and the mode for a categorical variable. The user can also use the modxVals argument to specify a different selection of values of the moderator for plotting.

Figure 1 illustrates the plotSlopes function for two use cases. The first is the default selection of values for the moderator. The second example in that figure illustrates user-selected values for the moderator, which in this case are {8, 10.5, 12}.



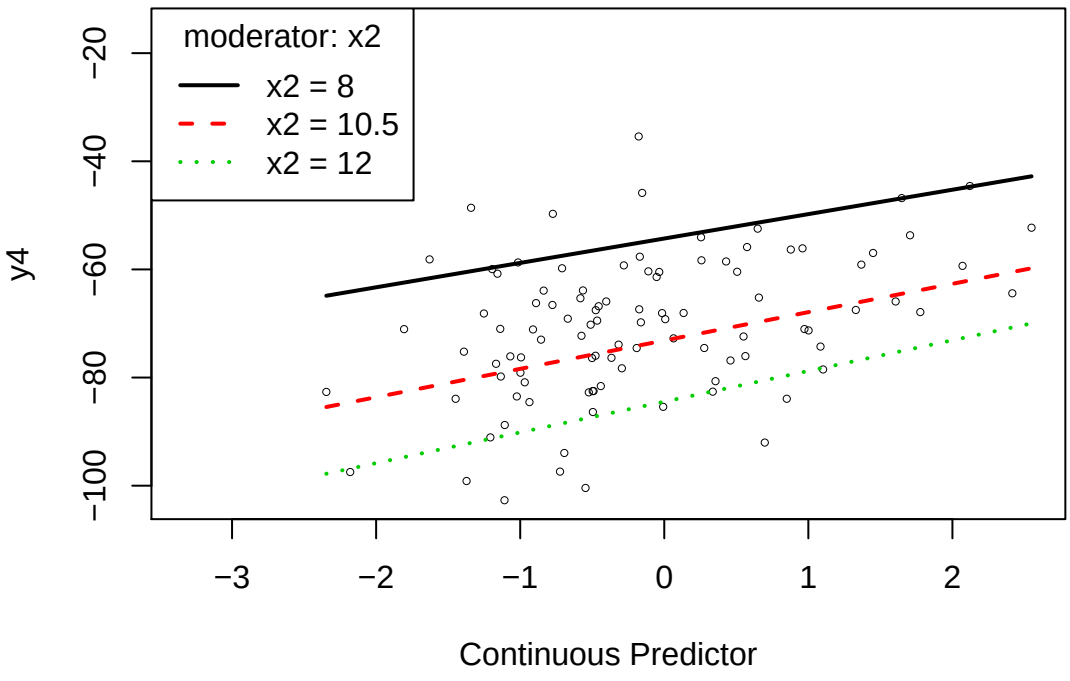


Figure 1: plotSlopes Illustrated

plotSlopes is intended for linear models with multiplicative interactions, but it will also draw ordinary linear models. It returns an object that may facilitate further analysis. The output object includes the newdata object that was used to construct the plots that are draw. That output object is used by the function testSlopes, which is discussed below.

The plotSlopes function also works well if the moderator is a categorical variable. When modx is categorical, the points and lines are drawn with colors that represent the categories of the plotted responses. Suppose we have a four-valued categorical variable, "West","Midwest", "South", and "East". If that variable is used in an interaction in the regression model, then the plotSlopes output will include four lines, one for each region. Like other plot functions in R, the col option can be used to adjust the colors to suit the taste of the user. In Figure 2, the categorical variable is x4.

4.2 testSlopes, a companion of plotSlopes

In psychology, methodologists have recommended the analysis of "simple slopes" to depict the effect of several variables in a 2 dimensional plot. This is most often of interest in the analysis of regression models with interactive terms.

Aiken and West (and later Cohen, Cohen, West, and Aiken) propose using the t test to find out if the effect of the "plotx" variable is statistically significantly different from zero for each particular value of the moderator variable. The user should first run plotSlopes, and then submit the output object to testSlopes. The usual use case would be the following:

```
m4 <- lm (y1 ~ x1*x2 + x3, data=dat)
m4ps <- plotSlopes(m4, plotx="x1", modx="x2", xlab="x1 is a Continuous Predictor")
testSlopes(m4ps)
```

The output from that testSlopes usage is illustrated in Figure 3.

The hypothesis tests reported by testSlopes should be understood as follows. Each of the lines in the output from plotSlopes, say Figure 1, can be tested to find out if its "simple slope" is different from zero. The tests calculated by testSlopes represent the null hypothesis that

$$H_0 : 0 = \hat{b}_{simple\ slope} = \hat{b}_{plotx} + b_{plotx \cdot modx} modx \tag{2}$$

where $modx$ is the moderator variable and $plotx$ is plotted on the horizontal axis in the plotSlopes output.

Following a suggestion of Preacher, Curran, and Bauer (2006), the testSlopes function also tries to calculate the Johnson-Neyman (1936) interpretation of the same test. It presents a plot, as illustrated in Figure 3. The J-N test would have us ask, "for which values of the moderator would the value $\hat{b}_{simple\ slope}$ be statistically significantly different from zero?" The J-N calculation requires the solution an equation that is quadratic in the value of the moderator variable, $modx$. The interval of values of $modx$ associated with a statistically significant effect of $plotx$ on the outcome is determined from the computation of a T statistic for $\hat{b}_{simple\ slope}$. The J-N interval is the set of values of $modx$ for which the following holds:

$$\hat{t} = \frac{\hat{b}_{simple\ slope}}{std.err(\hat{b}_{simple\ slope})} = \frac{\hat{b}_{simple\ slope}}{\sqrt{\widehat{Var(\hat{b}_{plotx})} + modx^2 \widehat{Var(\hat{b}_{plotx \cdot modx})} + 2modx \widehat{Cov(\hat{b}_{plotx}, \hat{b}_{plotx \cdot modx})}}} \geq T_{\frac{\alpha}{2}, df} \tag{3}$$

Suppose there are two real roots, $root1$ and $root2$. The values of $modx$ for which the slope is statistically significant may be a compact interval, $[root1, root2]$, or it may two open intervals, $(-\infty, root1]$ and $[root2, \infty)$. The J-N interpretation is most useful when the moderator is a continuous variable and the result specifies an interval inside the range of the moderator. In quite a few cases, the J-N interval is outside the observed range of the moderator, which makes it either difficult to interpret or irrelevant.

4.3 plotCurves

plotCurves generalizes the plotting capability of plotSlopes. plotCurves should be able to handle any regression formulas that include nonlinear transformations. Models that have polynomials or terms that are logged (or otherwise transformed) can be plotted. In that sense, plotCurves is rather similar to R's own termplot function. The difference is that plotCurves allows for moderator variables, which implies that one can draw several different curves to represent separate groups.

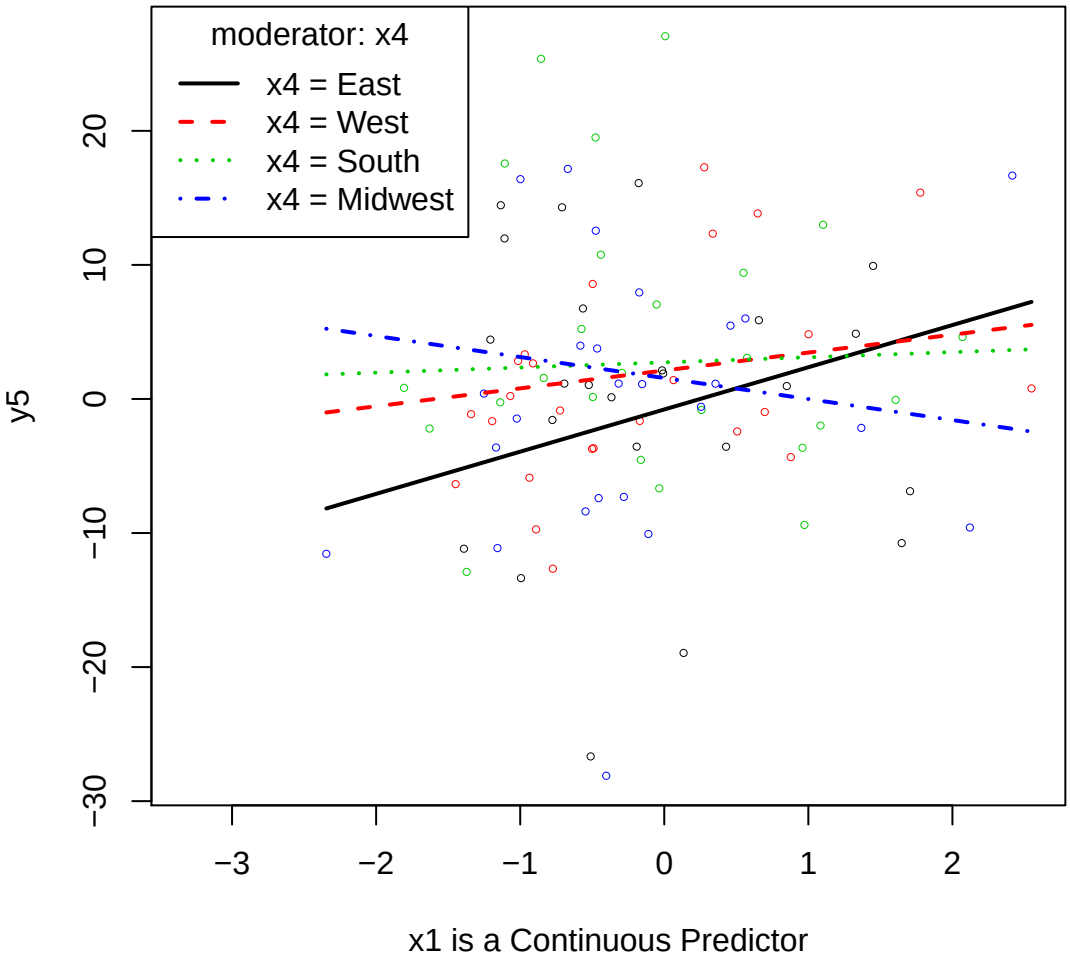


Figure 2: plotSlopes with a Categorical Moderator

```
testSlopes(m4psa)
```

```
These are the straight-line "simple slopes" of the variable x1
 for the selected moderator values.
        "x2"    slope Std. Error  t value     Pr(>|t|)
25%  9.440729 4.933532  1.0227332 4.823870 5.353731e-06
50% 10.032803 5.106538  0.9475171 5.389388 5.132885e-07
75% 10.627643 5.280352  1.1999348 4.400532 2.823809e-05
Values of modx INSIDE this interval:
       lo        hi
 7.801719 12.711204
cause the slope of (b1 + b2modx)plotx to be statistically significant
```

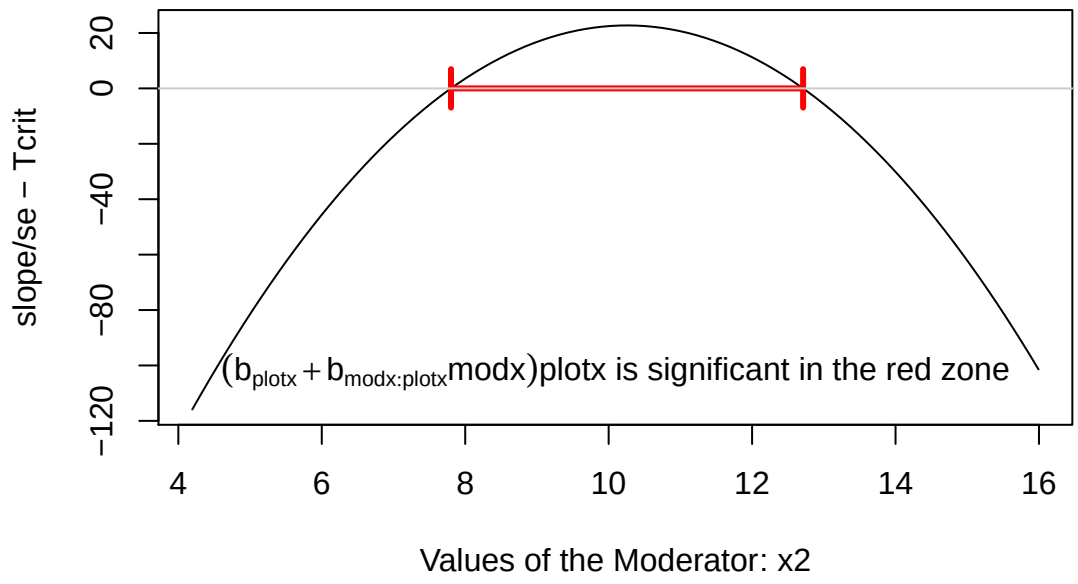


Figure 3: testSlopes for an Interactive Model

Suppose a dependent variable y5 is created according to a nonlinear process.

$$y5_i = -3x1_i + 1.5 * log(x2) + 1.1x2_i + 2.2x1_i \times x2_i + e_i \quad (4)$$

4.4 plotPlane

The persp function in R works well, but its interface is too complicated for most elementary and intermediate R users. To facilitate its use for regression users, the plotPlane is offered.

The plotPlane function offers a visualization of the mutual effect of two predictors in m4. See Figure 5 for the plot created by

```
plotPlane(m4, plotx1="x1", plotx2="x2")
```

plotPlane is designed to work like plotCurves, to tolerate nonlinear components in the regression formula. As illustrated in Figure 6, plotPlane allows the depiction of a 3 dimensional curving plane that "sits" in the cloud of data points. The variables that are not explicitly pictured in the plotPlane figure are set to central reference values. As illustrated in Figure 4, plotCurves is a 2 dimensional depiction of the same model.

At some point in the future, the ability to make plotSlopes and plotPlane work together will be introduced. The user will be able to see how the two and three dimensional graphs relate to each other. A preliminary rendering of what that might look like is presented in Figure 7. It is as if we can "press the plane down" into the 2-D slopes plot, or the 2-D simple slopes can be depicted in the 3 dimensional plane.

5 Standardized, Mean-Centered, and Residual-Centered Regressions

5.1 Standardized regression

Many of us learned to conduct regression analysis with SPSS, which (historically, at least) reported both the ordinary regression coefficients as well as a column of coefficients obtained from a regression in which each of the predictors in the design matrix had been "standardized." That is to say, each variable, for example $x1_i$, was replaced by an estimated $Z - score : (x1_i - \overline{x1})/std.dev.(x1_i)$. A regression fitted with those standardized variables is said to produce "standardized coefficients." These standardized coefficients, dubbed "beta weights" in common parlance, were thought to set different kinds of variables onto a common metric. While this idea appears to have been in error (see, for example, King 1986), it still is of interest to many scholars who want to standardize their variables in order to compare them more easily.

The function standardize was included in rockchalk to facilitate lectures about what a researcher ought not do. standardize performs the complete, mindless standardization of all predictors, no matter whether they are categorical, interaction terms, or transformed values (such as logs). Each column of the design matrix is scaled to a new variable with mean 0 and standard deviation 1. The input to standardize should be a fitted regression model. For example:

```
m4 <- lm (y4 ~ x1 * x2, data=dat)
m4s <- standardize(m4)
```

It does seem odd to me that a person would actually want a standardized regression of that sort, and the commentary included with the summary method for the standardized regression object probably makes that clear.

```
summary(m4s)
```

```
All variables in the model matrix and the dependent variable
were centered. The centered variables have the letter "s" appended to their
non-centered counterparts, even constructed
variables like `x1:x2` and poly(x1,2). We agree, that's probably
ill-advised, but you asked for it by running standardize().

The rockchalk function meanCenter is a smarter option, probably.

The summary statistics of the variables in the design matrix.
         mean std.dev.
```

```
m6 <- lm (y5 ~ log(x2*x2) + x1 * x2, data=dat)
plotCurves(m6, plotx="x2", modx="x1")
```

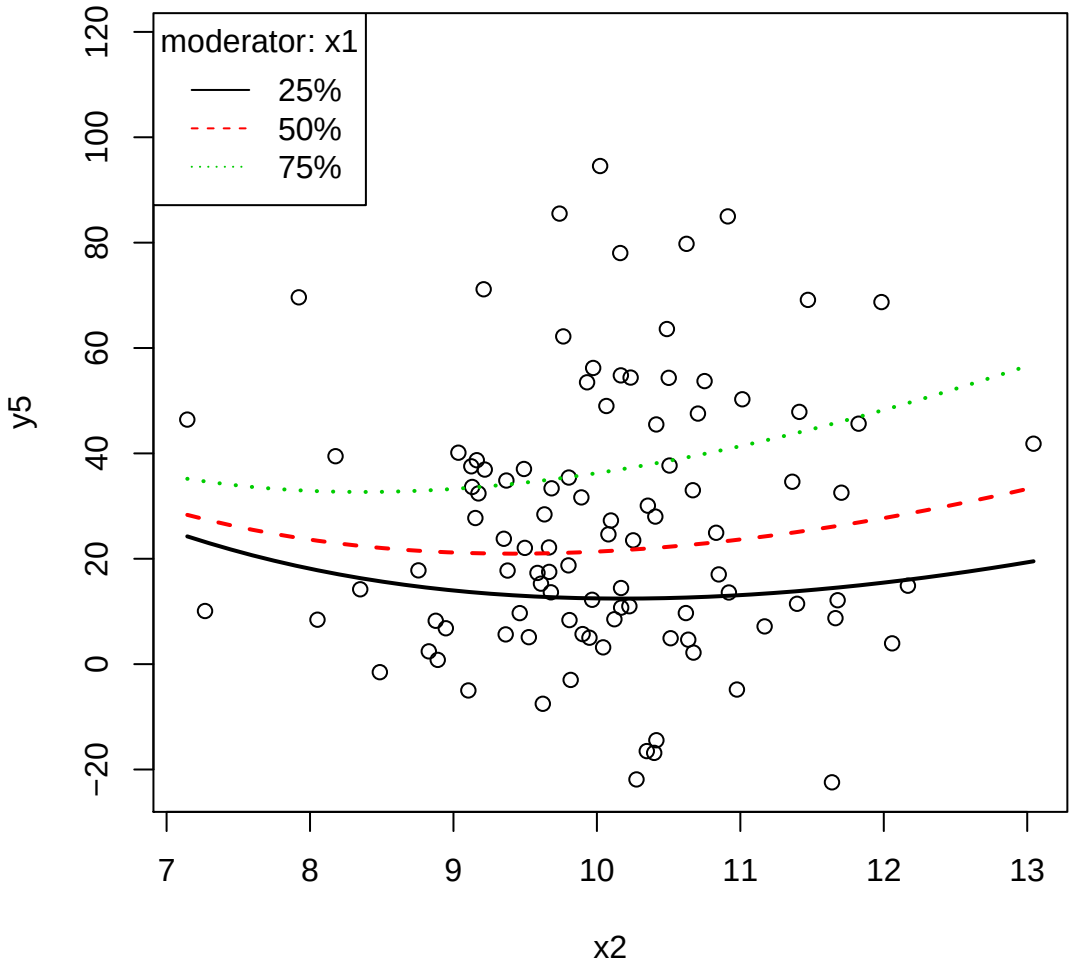


Figure 4: plotCurves

```
p100 <- plotPlane(m4, plotx1="x1", plotx2="x2")
```

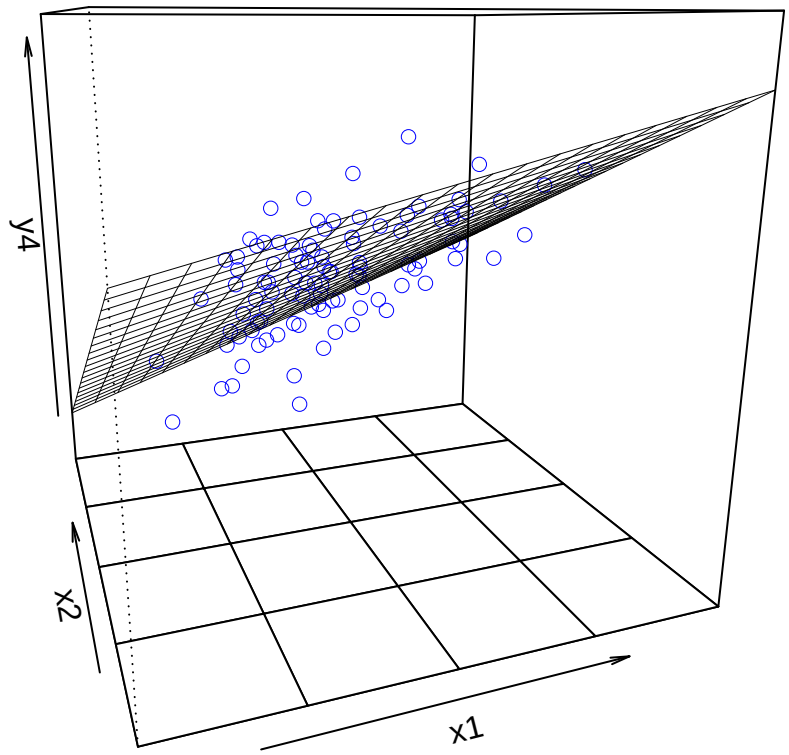
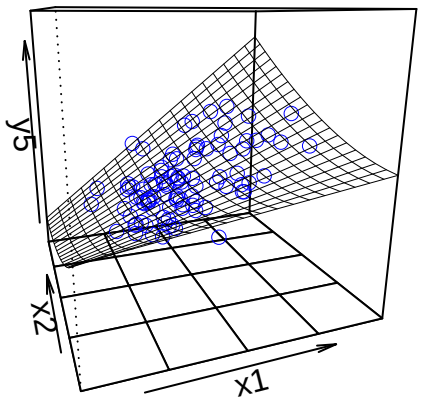


Figure 5: plotPlane for the Interactive Model

```
plotPlane(m6, plotx1="x1", plotx2="x2")
```

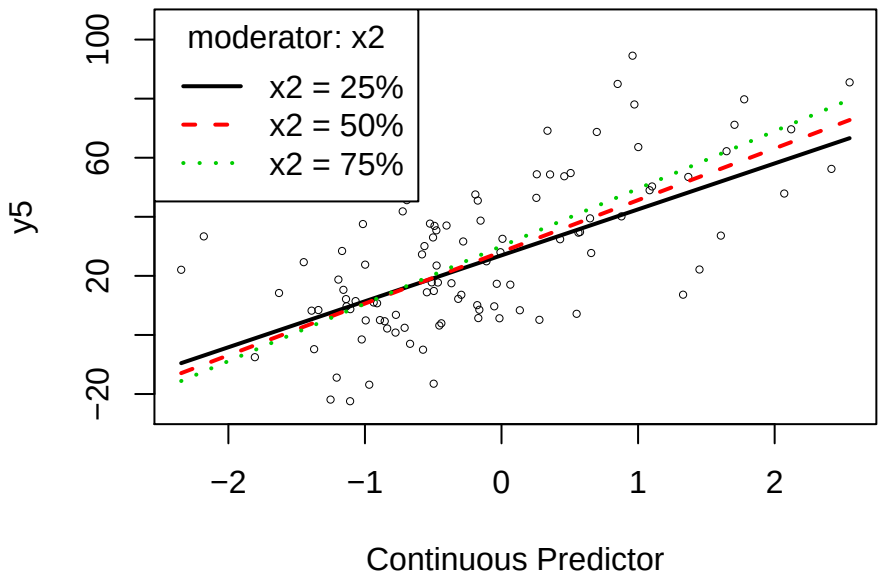



```
outreg(m6, tight=FALSE)
```

	Estimate	(S.E.)
(Intercept)	277.661	(275.117)
log(x2 * x2)	-105.706	(106.922)
x1	-15.519	(20.095)
x2	23.719	(21.758)
x1:x2	3.292	(2.029)
N	100	
$RMSE$	18.656	
R^2	0.474	
adj R^2	0.452	

* $p \leq 0.05$

Figure 6: plotPlane



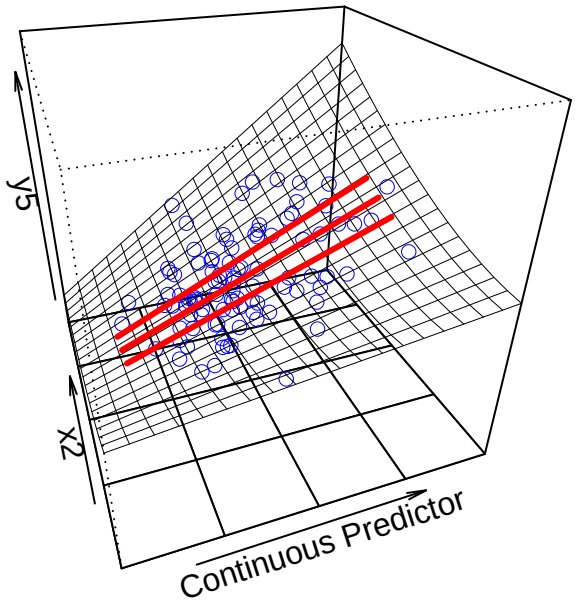


Figure 7: Making plotSlopes and plotPlane work Together

Table 7: Comparing Ordinary and Standardized Regression

	Not Standardized		Standardized	
	Estimate	(S.E.)	Estimate	(S.E.)
(Intercept)	6.291	(9.276)	0	(0.07)
x1	2.118	(9.753)	.	
x2	-7.567*	(0.921)	.	
x1:x2	0.296	(0.984)	.	
x1s	.		0.162	(0.746)
x2s	.		-0.595*	(0.072)
‘x1:x2s‘	.		0.224	(0.747)
N	100		100	
$RMSE$	9.241		0.704	
R^2	0.52		0.52	
adj R^2	0.505		0.505	

* $p \leq 0.05$

```
y4s          0         1
x1s          0         1
x2s          0         1
`x1:x2s`     0         1

Call:
lm(formula = y4s ~ x1s + x2s + `x1:x2s`, data = stddat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.05325 -0.52401  0.04822  0.46293  1.36599

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.984e-16  7.037e-02   0.000    1.000
x1s          1.620e-01  7.460e-01   0.217    0.829
x2s         -5.948e-01  7.236e-02  -8.220 9.71e-13 ***
`x1:x2s`     2.244e-01  7.465e-01   0.301    0.764
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7037 on 96 degrees of freedom
Multiple R^2: 0.5198,   Adjusted R^2: 0.5048
F-statistic: 34.64 on 3 and 96 DF,  p-value: 2.924e-15
```

5.2 Mean-centered Interaction Models

Sometimes people will fit a model like this

$$y_i = b_o + b_1 x1_i + b_2 x2_i + e_i \tag{5}$$

and then wonder, “is there an interaction between $x1_i$ and $x2_i$?” The natural inclination is to run this model,

```
m1 <- lm(y ~ x1*x2)
```

or its equivalent

```
m2 <- lm(y ~ x1 + x2 + x1:x2)
```

For a variety of reasons, researchers have been advised that they should not run the ordinary interaction model. Instead, they should “mean center” the variables $x1$ and $x2$ before entering them into the regression model. That is, they should replace $x1_i$ with $(x1_i - \overline{x1})$ and $x2_i$ with $(x2_i - \overline{x2})$, so that the fitted model is actually

$$y_i = b_o + b_1(x1_i - \overline{x1}) + b_2(x2_i - \overline{x2}) + b_3(x1_i - \overline{x1})(x2_i - \overline{x2}) + e_i \tag{6}$$

This is easy enough to do in R, but it can be tedious to center the variables and then run the model. To make it easier for users to compare the results of the "ordinary interaction" and the "mean centered" model, this package includes a function meanCenter. meanCenter will receive a model, scan it for interaction terms, and then center the variables that are involved in interactions. It is used as follows. First, fit any regression, such as m4 above, the same one with which the standardize function was demonstrated. Pass the output object to the meanCenter function.

```
m4mc <- meanCenter(m4)
summary(m4mc)
```

```
These variables were mean-centered before any transformations were made on the design matrix.
[1] "x1c" "x2c"
The centers and scale factors were
             x1c      x2c
mean   -0.1567617 10.04124
scale   1.0000000  1.00000
The summary statistics of the variables in the design matrix (after centering).
              mean std.dev.
y4       -70.49561 13.13142
x1c        0.00000  1.00441
x2c        0.00000  1.03219
x1c:x2c   -0.02605  0.96090

The following results were produced from:
meanCenter.default(model = m4)

Call:
lm(formula = y4 ~ x1c * x2c, data = stddat)

Residuals:
     Min       1Q   Median       3Q      Max
-26.9622  -6.8810   0.6332   6.0789  17.9374

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -70.4879     0.9244 -76.252  < 2e-16 ***
x1c           5.0892     0.9413   5.406 4.69e-07 ***
x2c          -7.6133     0.9009  -8.451 3.13e-13 ***
x1c:x2c       0.2959     0.9843   0.301    0.764
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.241 on 96 degrees of freedom
Multiple R^2: 0.5198,   Adjusted R^2: 0.5048
F-statistic: 34.64 on 3 and 96 DF,  p-value: 2.924e-15
```

The default settings for meanCenter cause it to center only the variables involved in an interaction, and it leaves the others unchanged. If the user wants all of the numeric predictors to be mean-centered, the usage would be

```
m4mc <- meanCenter(m4, centerOnlyInteractors = FALSE)
summary(m1mc4)
```

By default, it does not standardize while centering (but the user can request standardization). Users who want to standardize the variables that are centered can use the argument standardize=TRUE. The option centerDV causes the dependent variable to be centered as well.

5.3 Residual-centered Models

Residual-centering is another adjustment that has been recommended for models that include interactions or squared terms. Like mean-centering, it is recommended mainly as a way to ameliorate multicollinearity.

I think of residual-centering as follows. Suppose we fit the linear model, with no interaction

$$y = c_0 + c_1 x1 + c_2 x2 + e_i. \tag{7}$$

Suppose that those parameter estimates, $\hat{c}_1$, $\hat{c}_2$, are the "right ones". We want to estimate the interactive model,

$$y_i = b_o + b_1 x1_i + b_2 x2_i + b_3 x1_i \times x2_i + e_i, \tag{8}$$

but if we estimate that, it will "ruin" our estimates for the effects of $x1$ and $x2$. So we proceed by constraining the fitted coefficients in the interactive model so that the main effects remain the same. That is to say, require

that the parameter estimates of $x1$ and $x2$ must match match estimates from equation 7. Effectively, $\hat{b}_1 = \hat{c}_1$ and $\hat{b}_2 = \hat{c}_2$.

How can this be done in a convenient, practical way? The answer: use "residual-centering." First, estimate the following regression, in which the left hand side is the interaction product term:

$$(x1_i \times x2_i) = d_0 + d_1 x1_i + d_2 x2 + u_i \tag{9}$$

The residuals from that regression are, by definition, orthogonal to both $x1$ and $x2$. Call those fitted residuals $\widehat{u}_i$. We fit the interactive model using $\widehat{u}_i$ in place actual product term $(x1_i \times x2_i)$.

$$y_i = b_0 + b_1 x1_i + b_2 x2_i + b3\widehat{u}_i + e_i, \tag{10}$$

In essence, we have taken the interaction $(x1_i \times x2_i)$, and purged it of its parts that are linearly related to $x1_i$ and $x2_i$ separately.

The rockchalk function residualCenter handles this for the user. Like meanCenter, the user has to fit an interactive model first, and the result object is passed to residualCenter like so:

```
m4rc <- residualCenter(m4)
summary(m4rc)
```

```
Call:
lm(formula = y4 ~ x1 + x2 + x1.X.x2, data = mfnew)

Residuals:
     Min       1Q   Median       3Q      Max
-26.9622  -6.8810   0.6332   6.0789  17.9374

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.8586     9.0821   0.755    0.452
x1            5.0366     0.9249   5.445 3.97e-07 ***
x2           -7.6250     0.9000  -8.472 2.82e-13 ***
x1.X.x2       0.2959     0.9843   0.301    0.764
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.241 on 96 degrees of freedom
Multiple R^2: 0.5198,   Adjusted R^2: 0.5048
F-statistic: 34.64 on 3 and 96 DF,  p-value: 2.924e-15
```

The objects created by residualCenter are assigned the class "rcreg" and the package includes summary, print, and predict methods for these objects. In the regression output, the residual-centered interaction term is labeled as $(x1Xx2)$.

5.4 Why are we bothering with mean-centering and residual-centering in the first place?

In the long run, think the correct answer will be, "we were mistaken." Nevertheless, the advice that one ought to mean-center or residual-center in regression analysis has become quite widely established. The primary advocates of "mean-centering" have been Aiken and West (1991), who integrated that advice into the very widely used regression textbook, *Applied Multiple Regression/Correlation for the Behavioral Sciences* (Cohen, et. al , 2002). The advice that one ought to mean-center the predictors has been picked up in other fields. One statistics text for biologists notes, "We support the recommendation of Aiken & West (1991) and others that multiple regression with interaction terms should be fitted to data with centered predictor values" (Quinn and Keough, 2002, Chapter 6).

In order to understand how mean-centering came to seem like a "magic bullet," it is necessary to re-trace some steps to find out how we arrived in our current situation. For this example, I used the function genCorrelatedData in rockchalk. The "true model" from which the data is produced is

$$y_i = 2 + 0.1x1_i + 0.1x2_i + 0.2 \cdot (x1_i \times x2_i) + e_i, \tag{11}$$

where $e_i \sim N(0, 300^2)$ and $\rho_{x1,x2} = 0.4$.

Virtually everybody who has experimented with regression has had the "what the heck happened to my predictors?" experience. Please consider the Table 8. In the first column, we have the ordinary linear specification

Table 8: Comparing Regressions

	Linear		Interaction		Mean-centered		Residual-centered	
	Estimate	(S.E.)	Estimate	(S.E.)	Estimate	(S.E.)	Estimate	(S.E.)
(Intercept)	-562.148*	(98.822)	86.498	(385.418)	533.508*	(16.808)	-562.148*	(98.57)
x1	11.958*	(1.874)	-1.6	(8.009)	.		11.958*	(1.869)
x2	10.194*	(1.696)	-3.053	(7.795)	.		10.194*	(1.691)
x1:x2	.		0.273	(0.157)	.		.	
x1c	.		.		12.077*	(1.87)	.	
x2c	.		.		10.49*	(1.7)	.	
x1c:x2c	.		.		0.273	(0.157)	.	
x1.X.x2	.		.		.		0.273	(0.157)
N	400		400		400		400	
$RMSE$	315.489		314.685		314.685		314.685	
R^2	0.245		0.251		0.251		0.251	
adj R^2	0.241		0.245		0.245		0.245	

* $p \leq 0.05$

```
lm(y ~ x1 + x2, data=dat2).
```

The coefficients of $x1$ and $x2$ appear to be "statistically significant," a very gratifying regression indeed. It appears we might have found something!

Unable to leave well enough alone, the researcher wonders, "is there an interaction between $x1$ and $x2$?" The second column in Table 8 summarizes the regression that includes an interaction term. That interaction model, which adds the product variable $x1 \times x2$, is estimated in R with

```
lm(y ~ x1 * x2, data=dat2)
```

A quick scan of column two usually lead to the "what the heck?" or "Holy Cow!" response. The regression went to hell! Neither of the key variables, $x1$ nor $x2$, is "statistically significant" any more. While the coefficients for the variables $x1$ and $x2$ did seem to be substantial in the first model, the introduction of the interactive effect appears to ruin the whole thing. What should be done when adding a product term seems to "ruin" a regression model?

Cohen, et al. refer to the apparent instability of the coefficients as a reflection of "inessential collinearity" among the predictors, due to the fact that $x1$ and $x2$ are correlated with the new term, $x1 \times x2$. They advised their readers to "mean center" their predictors, to subtract the mean of each predictor from the observed model and run the regression again.

Mean-centering seems to help. The result of the meanCenter function is displayed in the third column of Table 8. It appears that the estimates for the slopes are "significant again" and we have "solved" the problem of inessential collinearity.

The solution, however, is simply an illusion. Technical rebuttals have been published (Kromrey, J. D., & Foster-Johnson, L. , 1998; Echambadi and Hess, 2007), but applied researchers continue to use the practice. The argument against mean-centering is really quite simple. It has no effect. There is no benefit. The ordinary model and the mean-centered models are actually exactly the same in every important way. The technical critiques have focused on the multicollinearity issue, but they leave open the possibility that mean-centering may facilitate interpretation of the estimates. The presentation here should convince the reader that even the interpretation is not facilitated by mean-centering.

The first hint of trouble is in the fact that the coefficient of the interactive effect in columns 2 and 3 is identical. Those coefficients are the same because they are estimates of a constant, the cross partial derivative $\partial^2 y / \partial x1 \partial x2$. That is to say, when the different models try to estimate the same coefficient, they get the same result. Note as well that the root mean square and R^2 estimates are identical. Is it possible that the mean-centered regression could really be "better" if its fit statistics are not altered?

The models only appear different because we sometimes forget that we are studying a nonlinear problem when the regression model includes interactions. To assist in the visualization of this situation, we can use

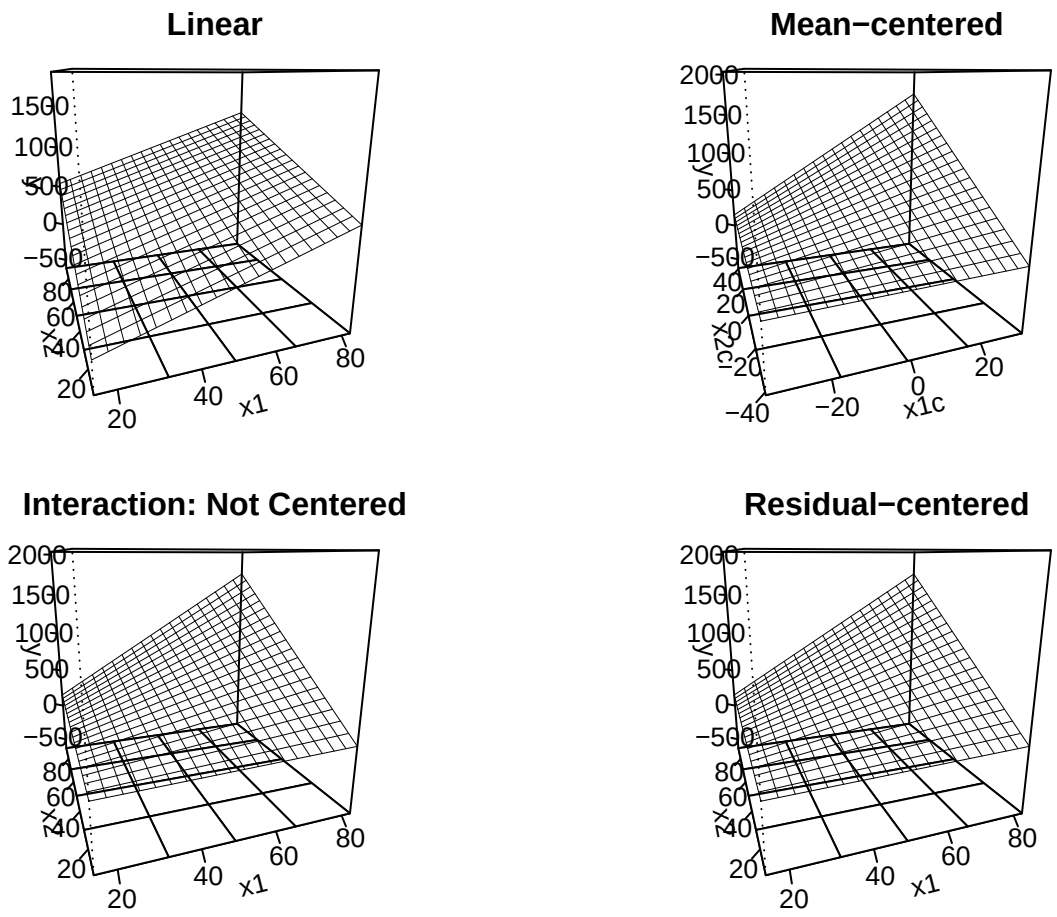


Figure 8: Predicted Planes from Centered and Uncentered Fits Identical

some functions in rockchalk. In Figure 8, we see that the ordinary interaction and mean-centered models produce identical predicted values!

The curves in Figure 8 are identical. With the original, non-transformed data, the vertical (y) axis is positioned at the front-left edge of the graph, while the centered one re-positions the y axis into the center of the graph. The coefficient estimates from a regression model on a nonlinear surface depend on the location of the y axis. At some points, the estimates will be large, at some points they will be small. Mean-centering may accidentally reposition the axis to a location that has "better" (bigger?) point estimates. The estimated standard errors, of course, also change as we move the y axis about. In the middle of the data, the point estimates of the standard errors are generally smaller than they are when the axis is positioned on the periphery of the data. It can be shown that, from either fitted model, the estimated slopes and standard errors at any given point in the data are exactly the same. Included with the rockchalk package, in the examples folder, one can find a file called "residualCentering.R" that walks through this argument step by step.

If mean-centering does not help, perhaps residual-centering will address the problem. Residual-centering is offered as a new, improved sort of mean-centering. Where mean-centering seems to reduce the damage done by inessential collinearity, residual-centering completely eliminates it. Because the residual-centered interaction variable is, by definition, completely uncorrelated with the other variables in the model, the problem of collinearity seems to be completely solved.

The fourth column in Table 8 presents the residual centered results. The parameter estimates are a little larger, the standard errors are a bit smaller. In the way that applied researchers usually look at situations

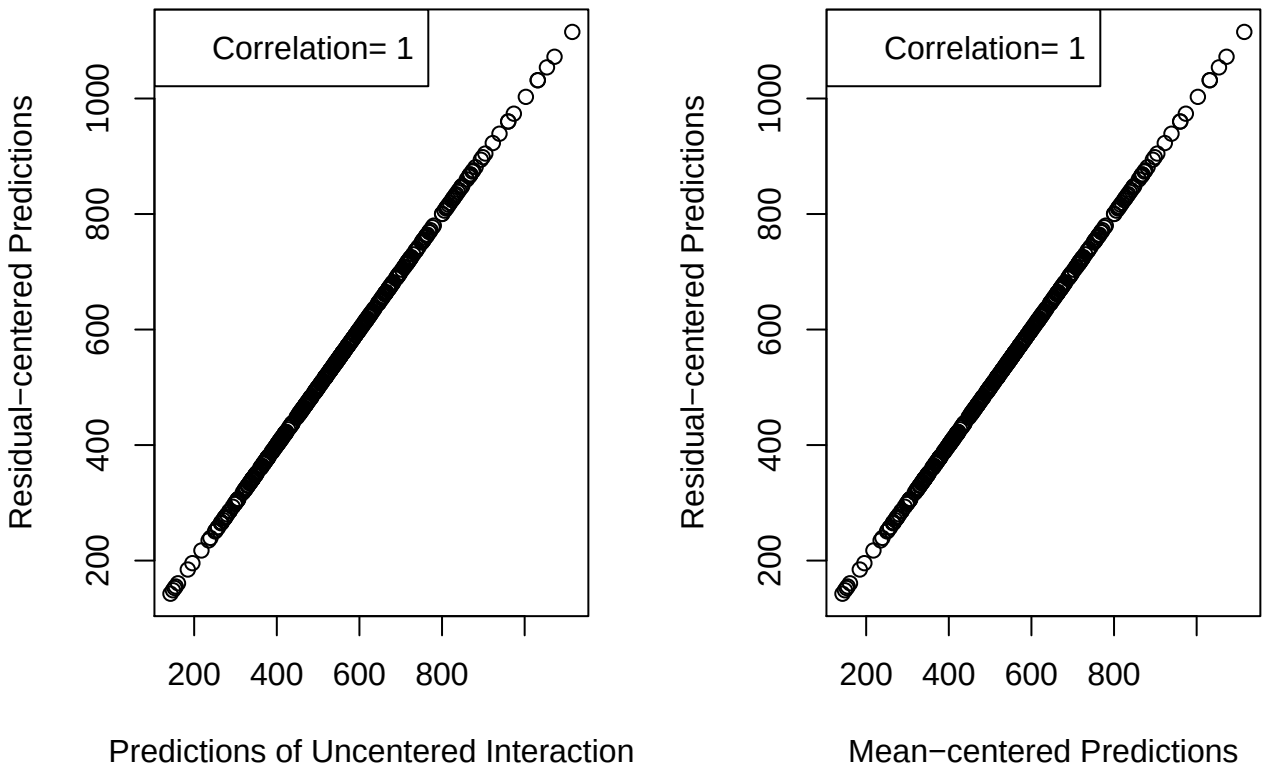


Figure 9: Predicted Values of Mean and Residual-centered Models

like this, it is a "better" model.

And, yet again, our high hopes are dashed. In the end, we will find out that the residual-centered model is completely equivalent to the ordinary interaction model and the mean-centered model. My effort to visualize this was initially frustrated by the difficulty of writing a predict function that worked for arbitrary new data objects, but once that was finished, the result became completely clear. The predicted values of the ordinary interactive model, the mean-centered model, and the residual-centered models are exactly the same. Perhaps the most persuasive case is found in Figure 9.

The conclusion is this. One can code a nonlinear model in various ways, all of which are theoretically and analytically identical. There are superficial differences in the estimates of the coefficients of the various specifications, but these differences are understandable in light of the changes in the design matrix. The connection between the observed values of the predictors and the predicted values remains the same in all of these specifications.

Why do the coefficients differ if the models are actually the same? Recall that we are estimating the slopes of a curving plane, and so estimates of the marginal effects of $x1$ and $x2$ will depend on the point at which we are calculating the slopes. Mean-centering and residual-centering are different methods for re-positioning the y axis. The interactive model has a constant mixed partial derivative, so the estimate of the interaction coefficient is not affected by the position of the y axis. The other coefficients, however, do change.

It is possible to translate between the estimates of any one of these fitted models and the estimates of

another. The derivation proceeds as follows. The ordinary model is

$$y_i = b_0 + b_1 x1_i + b_2 x2_i + b3(x1_i \times x2_i) + e1_i \tag{12}$$

The mean-centered model is

$$y_i = c_0 + c_1(x1_i - \overline{x1}) + c_2(x2_i - \overline{x2}) + c_3(x1_i - \overline{x1}) \times (x2_i - \overline{x2}) + e2_i \tag{13}$$

In order to compare with equation 12, we would re-arrange like so

$$y_i = c_0 + c_1(x1_i) - c_1\overline{x1} + c_2(x2_i) - c_2\overline{x2} + c_3(x1_i x2_i + \overline{x1x2} - \overline{x1}x2_i - \overline{x2}x1_i) + e2_i \tag{14}$$

$$y_i = c_0 + c_1(x1_i) - c_1\overline{x1} + c_2(x2_i) - c_2\overline{x2} + c_3(x1_i x2_i) + c_3\overline{x1x2} - c_3\overline{x1}x2_i - c_3\overline{x2}x1_i) + e2_i \tag{15}$$

$$y_i = \{c_0 - c_1\overline{x1} - c_2\overline{x2} + c_3\overline{x1x2}\} + \{c_1 - c_3\overline{x2}\}x1_i + \{c_2 - c_3\overline{x1}\}x2_i + c_3(x1_i x2_i) + e2_i \tag{16}$$

One can then compare the parameter estimates from equations 12 and 16 in order to understand the observed changes in fitted coefficients after changing from the ordinary to the mean-centered coding. Both 12 and 16 include a single parameter times $(x1_i x2_i)$, leading one to expect that the estimate $\hat{b}_3$ should be equal to the estimate of $\hat{c}_3$ (and they are, as we have found). Less obviously, one can use the fitted coefficients from either model to deduce the fitted coefficients from the other. The following equalities describe that relationship.

$$\hat{b}_0 = \hat{c}_0 - \hat{c}_1\overline{x1} - \hat{c}_2\overline{x2} + \hat{c}_3\overline{x1x2} \tag{17}$$
$$\hat{b}_1 = \hat{c}_1 - \hat{c}_3\overline{x2} \tag{18}$$
$$\hat{b}_2 = \hat{c}_2 - \hat{c}_3\overline{x1} \tag{19}$$
$$\hat{b}_3 = \hat{c}_3 \tag{20}$$

The estimated fit of equation 13 would provide estimated coefficients $\hat{c}_j$, $j = 0, ..., 3$, which would then be used to calculate the estimates from the noncentered model.

The estimation of the residual centered model requires two steps. First, estimate a regression

$$(x1_i \times x2_i) = d_0 + d_1 x1_i + d_2 x2_i + u_i \tag{21}$$

from which the predicted value can be calculated:

$$\widehat{(x1_i \times x2_i)} = \hat{d}_0 + \hat{d}_1 x1_i + \hat{d}_2 x2_i. \tag{22}$$

The residual of that regression

$$\widehat{u}_i = (x1_i \times x2_i) - \widehat{(x1_i \times x2_i)} \tag{23}$$

is used as the "residual-centered" predictor in place of $(x1_i \times x2_i)$ in equation 12.

$$y_i = h_0 + h_1 x1_i + h_2 x2_i + h_3\{x1_i \times x2_i - \widehat{x1_i \times x2_i}\} + e3_i \tag{24}$$

Replacing $\widehat{x1_i \times x2_i}$ with $\hat{d}_0 + \hat{d}_1 x1_i + \hat{d}_2 x2_i$, 24 becomes

$$y_i = h_0 + h_1 x1_i + h_2 x2_i + h_3\{x1_i \times x2_i - \hat{d}_0 - \hat{d}_1 x1_i - \hat{d}_2 x2_i\} + e3_i \tag{25}$$
$$= h_0 + h_1 x1_i + h_2 x2_i + h_3\{x1_i \times x2_i\} - h_3\hat{d}_0 - h_3\hat{d}_1 x1_i - h_3\hat{d}_2 x2_i\} + e3_i \tag{26}$$
$$\{h_0 - h_3\hat{d}_0\} + \{h_1 - h_3\hat{d}_1\}x1_i + \{h_2 - h_3\hat{d}_2\}x2_i + h_3\{x1_i \times x2_i\} + e3_i \tag{27}$$

As in the previous comparison of models, we can translate coefficient estimates between the ordinary specification and the residual-centered model. The coefficient estimated for the product term, $\hat{h}_3$, should be

equal to $\hat{b}_3$ and $\hat{c}_3$ (and it is!). If we fit the residual centered model, 24, we can re-generate the coefficients of the other models like so:

$$b_0 = \hat{c}_0 - \hat{c}_1\overline{x1} - \hat{c}_2\overline{x2} + \hat{c}_3\overline{x1x2} = h_0 - h_3\hat{d}_0 \quad (28)$$

$$b_1 = \hat{c}_1 - \hat{c}_3\overline{x2} = h_1 - h_3\hat{d}_1 \quad (29)$$

$$b_2 = \hat{c}_2 - \hat{c}_3\overline{x1} = h_2 - h_3\hat{d}_2 \quad (30)$$

From the preceding, it should be clear enough that the three models are equivalent, in the sense that the parameter estimates (parameters, predicted values, and so forth) from any one can be translated into the terminology of the other.

References

[1] Aiken, L. S., & West, S. G. (1991). *Multiple Regression: Testing and Interpreting Interactions.* Newbury Park, Calif: Sage Publications.

[2] Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2002). *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences* (Third.). Routledge Academic.

[3] Echambadi, R., & Hess, J. D. (2007). Mean-Centering Does Not Alleviate Collinearity Problems in Moderated Multiple Regression *Models. Marketing Science*, 26(3), 438–445. doi: 10.1287/mksc.1060.0263

[4] Kromrey, J. D., & Foster-Johnson, L. (1998). Mean Centering in Moderated Multiple Regression: Much Ado about Nothing. *Educational and Psychological Measurement*, 58(1), 42 –67. doi:10.1177/0013164498058001005

[5] Little, T. D., Bovaird, J. A., and Widaman, K. F. (2006). On the Merits of Orthogonalizing Powered and Product Terms: Implications for Modeling Interactions Among Latent Variables. *Structural Equation Modeling*, 13(4), 497-519.

[6] Preacher, K. J., Curran, P. J., & Bauer, D. J. (2006). Computational Tools for Probing Interactions in Multiple Linear Regression, Multilevel Modeling, and Latent Curve Analysis. *Journal of Educational and Behavioral Statistics*, 31(4), 437–448.

[7] Quinn, G.P, and Keough, Michael J. (2002) Experimental Design and Data Analysis for Biologists. Cambridge University Press.The primary advocates of "mean-centering" have been Aiken and West (1991), who then integrated that advice into the very widely used regression textbook, *Applied Multiple Regression/Correlation for the Behavioral Sciences* (Cohen, et. al , 2002).