Using rockchalk for Quick & Consistent Regression Presentations

Paul Johnson

March 16, 2012

1 Introduction

The rockchalk package is an agglomeration of functions that I need when I'm teaching about regression. The functions here divide into three categories.

1. Functions that help me prepare lectures. 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 often ask 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. A funtion to estimate a "standardised 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 the use of my students. Those procedures are, possibly, not truly helpful and in order to demonstrate that fact, I have to provide the functions.

2 Some outreg Examples.

outreg was a function in search of a package for a long time. I didn't bother to build rockchalk until I had some other worthwhile functions. So it seems appropriate to start with outreg.

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.

These models are created with some simulated data.

```
set.seed(1234)
x1 <- rnorm(100)
x2 <- rnorm(100)
y1 <- 5*rnorm(100) - 3*x1 + 4*x2
```

Table 1: My One Tightly Printed Regression

	Estimate (S.E.)
(Intercept)	0.983
	(0.672)
x1	-2.713*
	(0.665)
N	100
$RMSE$	6.643
R^2	0.145
adj R^2	0.137

* $p \leq 0.05$

Table 2: My Spread Out Regressions

	Fingers	
	Estimate	S.E.
(Intercept)	0.983	(0.672)
x1	-2.713*	(0.665)
N	100	
$RMSE$	6.643	
R^2	0.145	
adj R^2	0.137	

* $p \leq 0.05$

```
y2 <- rnorm(100)+5*x2
dat <- data.frame(x1, x2, 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))
y3 <- rbinom(100, size=1, p=myilogit(scale(dat$y1)))
gm1 <- glm(y3~x1 + x2, data=dat)
```

In each of the floating tables, I have presented an example use of the "outreg" function along with the regression table that it creates.

Table 1 displays the default 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"))
```

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.

Table 3: My Two Linear Regressions Tightly Printed

(a) Tightly Formatted Columns

	Mine Estimate (S.E.)	Yours Estimate (S.E.)
(Intercept)	0.983 (0.672)	1.222* (0.546)
Billie	-2.713* (0.665)	.
x2	.	4.506* (0.531)
N	100	100
$RMSE$	6.643	5.458
R^2	0.145	0.423
adj R^2	0.137	0.417

* $p \leq 0.05$

(b) Two Columns Per Regression Model

	Mine		Yours	
	Estimate	S.E.	Estimate	S.E.
(Intercept)	0.983	(0.672)	1.222*	(0.546)
Billie	-2.713*	(0.665)	.	
x2	.		4.506*	(0.531)
N	100		100	
$RMSE$	6.643		5.458	
R^2	0.145		0.423	
adj R^2	0.137		0.417	

* $p \leq 0.05$

Table 4: My Three Linear Regressions in a Tight Format

	A Estimate (S.E.)	B Estimate (S.E.)	C Estimate (S.E.)
(Intercept)	0.983 (0.672)	1.222* (0.546)	0.818 (0.487)
I Forgot x1	-2.713* (0.665)	. 	-2.597* (0.482)
He Remembered x2	. 	4.506* (0.531)	4.442* (0.469)
N	100	100	100
$RMSE$	6.643	5.458	4.812
R^2	0.145	0.423	0.556
adj R^2	0.137	0.417	0.547

* $p \leq 0.05$

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)	0.983	(0.672)	1.222*	(0.546)	0.818	(0.487)
x1	-2.713*	(0.665)	.		-2.597*	(0.482)
x2	.		4.506*	(0.531)	4.442*	(0.469)
N	100		100		100	
$RMSE$	6.643		5.458		4.812	
R^2	0.145		0.423		0.556	
adj R^2	0.137		0.417		0.547	

* $p \leq 0.05$

```
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 variables in the R program 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 with some fine points of LaTeX table formatting. I also have not worried about the problem of restricting columns to use the exact same amount of horizontal space. In Table 5, we have regression output which is, in my opinion, completely acceptable for inclusion in a presentation or conference paper. Because the model labels are not equal in length, the columns are not equally sized. That is not a concern for me, at the moment, but I imagine it might be a concern for somebody. Perhaps, at some point, I may come back and deal with the problem that decimal values within columns should be vertically aligned (at least as an option). I don't want to make outreg output dependent on additional LaTeX packages.

Another feature of outreg is that it can present the estimates of different kinds of models. It can present the estimates from R's lm and glm functions in a single table. Consider Table 6, which resulted from the

Table 6: Combined OLS and GLM Estimates

	OLS:y1 Estimate (S.E.)	GLM: Categorized y1 Estimate (S.E.)
(Intercept)	0.983	0.417*
	(0.672)	(0.048)
x1	-2.713*	-0.114*
	(0.665)	(0.047)
x2	.	0.129*
		(0.046)
N	100	100
$RMSE$	6.643	
R^2	0.145	
adj R^2	0.137	
$Deviance$		21.525
$-2LLR(Model\chi^2)$		3.115

* $p \leq 0.05$

command

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

At one time, I was working on a similar presentation for mixed models estimated by lme4, but I stopped that effort because the lme4 package was changing rapidly and the format of its returned objects was not stable enough for a finalized presentation format. Eventually, I will include a method to display those mixed models.

3 plotPlane and plotCurves

The goal of plotPlane and plotCurves is to speed up the process of visualization in regression analysis. plotPlane offers a 3 dimensional drawing that uses R's persp function to do the heavy lifting. plotCurves tries to press that same information into a two dimensional display by drawing curves for several different values of a moderator variable. Both plotPlane and plotCurves allow nonlinear terms in the regression model that is being plotted. In that sense, they are more similar to R's own termplot function than to other similar tools.

In this section, I create a new dependent variable y5 and then put the fitted model through the plotSlopes and plotPlane functions.

```
dat$y5 <- with(dat, -3*x1 + 0.5*log(x2^2) + 1.1*x2 + 2.2 *x1 * x2 + 3*rnorm(100))
m5 <- lm (y5 ~ log(x2*x2) + x1 * x2, data=dat)
```

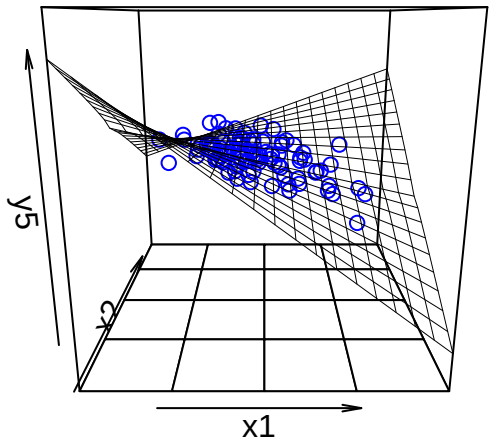
As illustrated in Figure 1, plotPlane allows the depiction of a 3 dimensional plane that "sits" in the cloud of data points. The variables that are not explicitly pictured in the plotPlane figure are set to reference values. As illustrated in Figure 2, plotCurves is a 2 dimensional depiction of the same information.

4 Plot and Test Simple Slopes

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. Suppse the fitted model is,

$$\hat{y}_i = \hat{b}_0 + \hat{b}_1 x1_i + \hat{b}_2 x2_i + \hat{b}_3 x1_i x2_i + x3_i.As \tag{1}$$

```
plotPlane(m5, plotx1="x1", plotx2="x2")
```




```
outreg(m5, tight=FALSE)
```

	Estimate	S.E.
(Intercept)	-0.002	(0.403)
log(x2 * x2)	0.423*	(0.156)
x1	-3.11*	(0.338)
x2	0.865*	(0.331)
x1:x2	1.872*	(0.355)
N	100	
$RMSE$	3.322	
R^2	0.605	
adj R^2	0.588	

* $p \leq 0.05$

Figure 1: plotPlane

```
plotCurves(m5, plotx="x1", modx="x2")
```

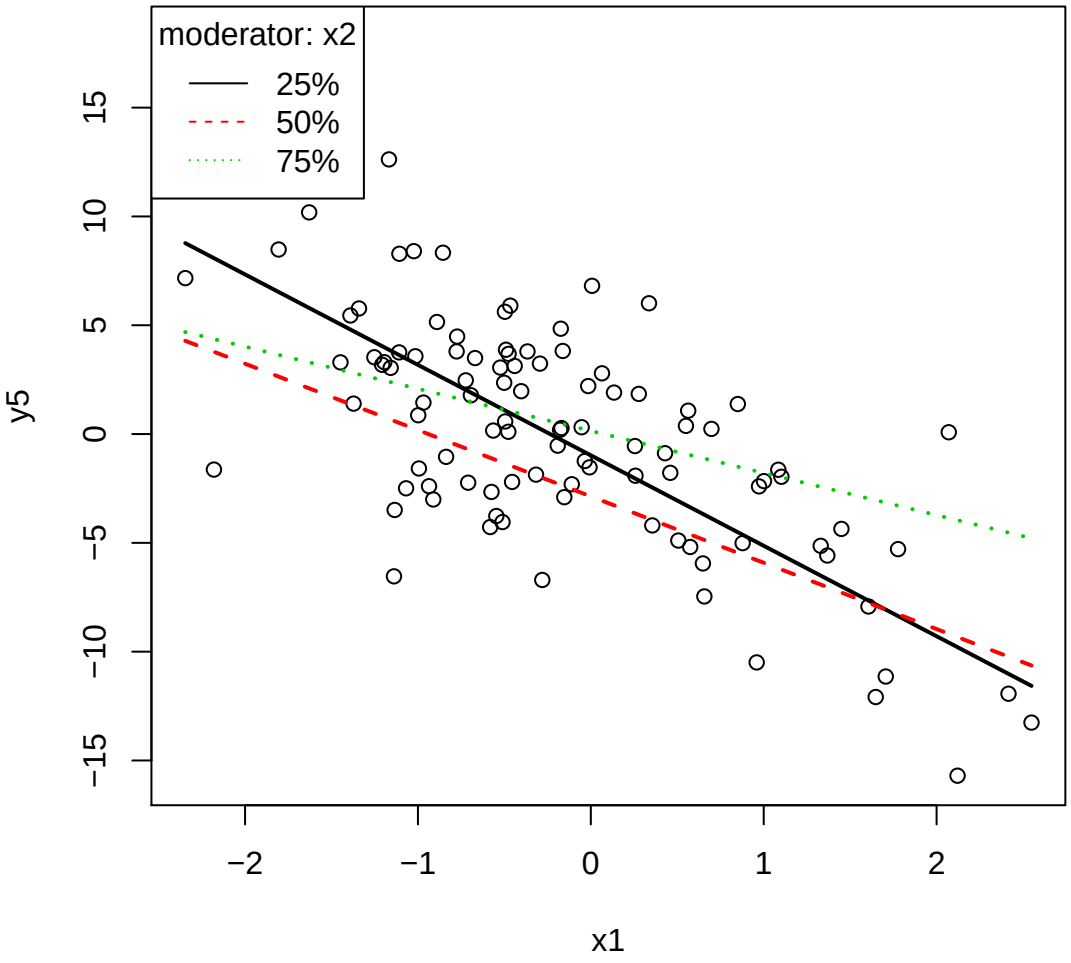


Figure 2: plotCurves

The idea is to consider the effect of $x1$ on y for several values of $x2$, keeping $x3_i$ set at some reference value (the mean for numeric variables). As a follow-up, one wants to test whether the plotted effects are statistically significantly different from zero.

This is only truly interesting when there are interaction effects, of course, but I begin with a simple linear model. Recall that

```
m3 <- lm (y1 ~ x1 + x2, data=dat)
```

Figure 3 illustrates the plotSlopes function for two use cases. The first is

```
plotSlopes(m3, plotx="x1", modx="x2", xlab="x1 is a Continuous Predictor")
```

The plotx argument is variable x1, meaning that x1 will be the horizontal axis, and x2 serves as the moderator variable. By default, three hypothetical values of x2 are selected (in this case the quantiles 25%, 50%, and 75%). The second example in that figure illustrates user-selected values for the moderator.

```
plotSlopes(m3, plotx="x1", modx="x2", modxVals=c(-0.2, 0.5, 0.9), xlab="Continuous Predictor")
```

```
testSlopes(m3psa)
```

```
There were no interactions in the plotSlopes object, so testSlopes can't offer any advice.
```

That model is linear, so lines are parallel. We need to introduce some interaction effects in order to exploit the new functions proposed here. Suppose we generate a new dependent variable and fit a regression with an interaction:

```
dat$y4 <- with(dat, -3*x1 + 6*x2 - 0.17*x1*x2 + 5*rnorm(100))
m4 <- lm (y4 ~ x1 * x2, data=dat)
```

A figure with lines for some values of the moderator x2, along with hypothesis test for those estimates, is obtained with the following. The "simple slope" lines that model are presented in Figure 4.

```
m4ps <- plotSlopes(m4, plotx="x1", modx="x2", xlab="Continuous Predictor")
```

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 significant for each particular value of "modx," the moderator variable. The testSlopes function delivers those t tests. Each of the lines represents a test of the hypothesis that

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

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

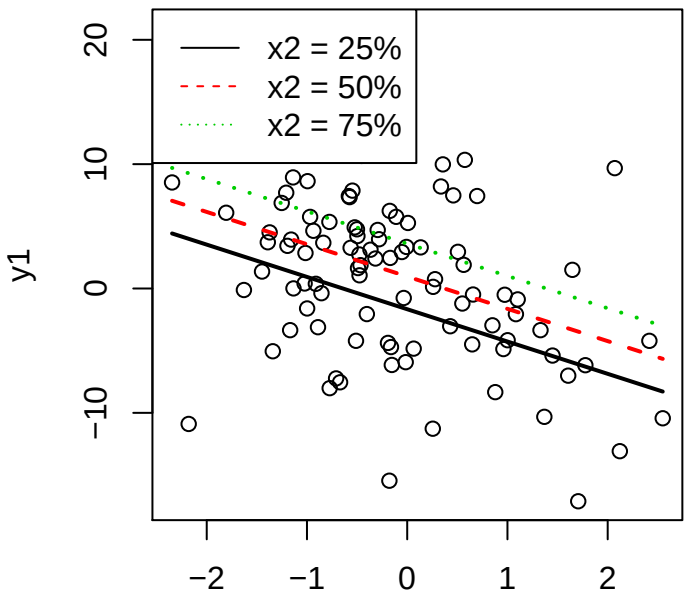
Following a suggestion of Preacher, Curran, and Bauer (2006), the testSlopes function also tries to calculate the Johnson-Neyman (1936) interpretation of the same test. It presents 2 diagnostic plots, as illustrated in Figure 5. Whereas West and Aiken would have us test the hypothesis that $\hat{b}_{simple\ slope} = \hat{b}_{plotx} + b_{plotx \cdot modx} modx$ is different from 0, J-N 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:

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

I am not entirely convinced that the J-N interpretation is useful, but calculating it was interesting. Nevertheless, the output of testSlopes(m4ps) is displayed in Figure 5.

The plotPlane function offers another visualization of the mutual effect of two predictors in m4. See Figure 6

At some point in the future, the ability to make plotSlopes and plotPlane work together will be introduced. So the user will be able to press the plane down into the 2 dimensional slopes plot, or the simple slopes can be depicted in the 3 dimensional plane. A preliminary rendering of what that might look like is presented in Figure 7



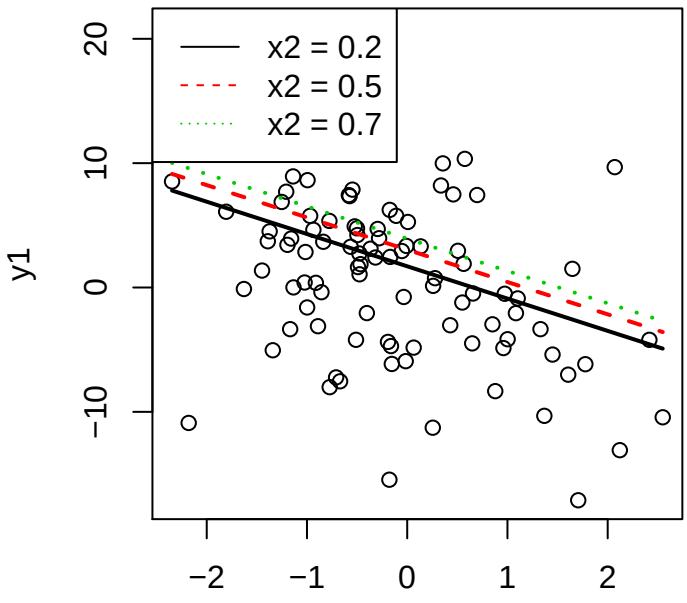


Figure 3: plotSlopes In an Additive Model

```
m4ps <- plotSlopes(m4, plotx="x1", modx="x2", xlab="Continuous Predictor")
```

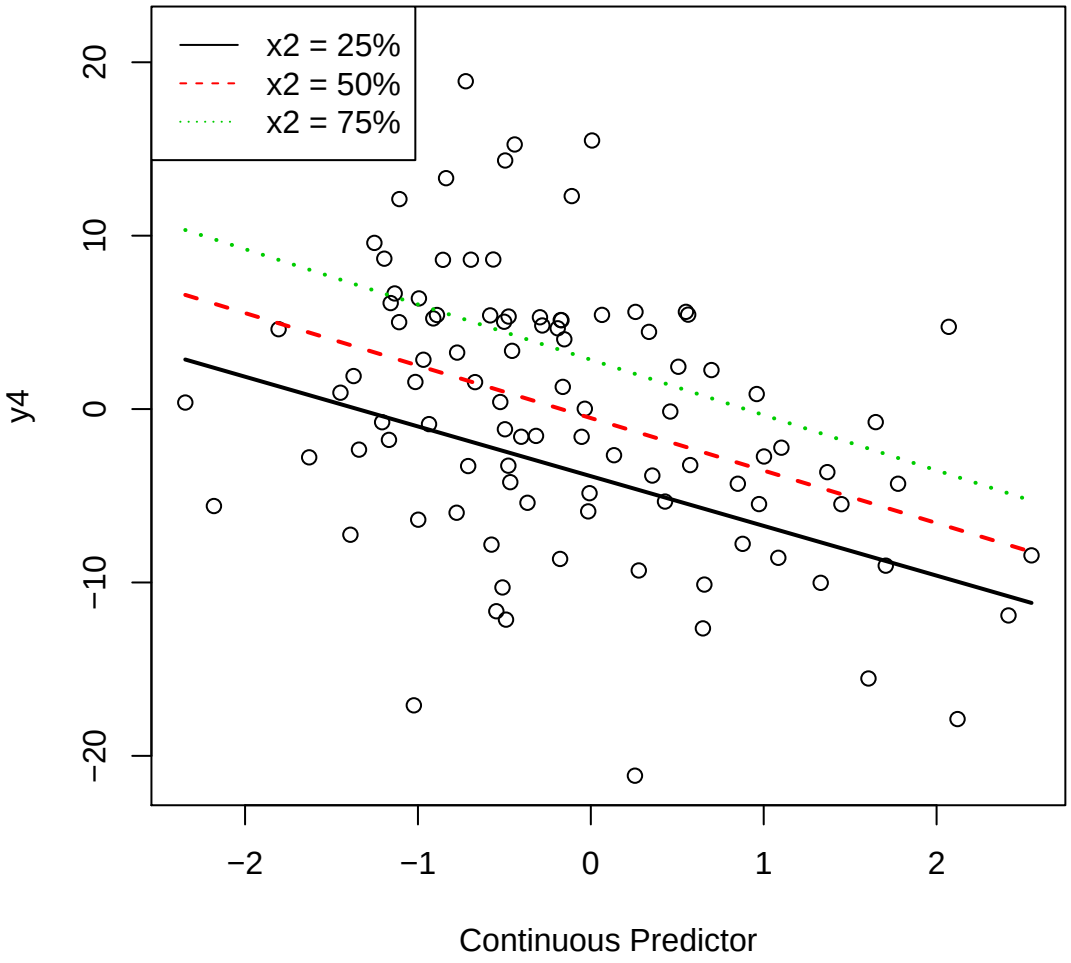


Figure 4: plotSlopes for an Interactive Model

```
testSlopes(m4ps)
```

```
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% -0.55927099 -2.868036  0.5075121 -5.651168 1.629298e-07
50%  0.03280328 -3.028238  0.4702790 -6.439237 4.744487e-09
75%  0.62764345 -3.189188  0.5965510 -5.346044 6.067756e-07
Values of modx INSIDE this interval:
       lo        hi
-2.377685  3.933088
cause the slope of (b1 + b2modx)plotx to be statistically significant
```

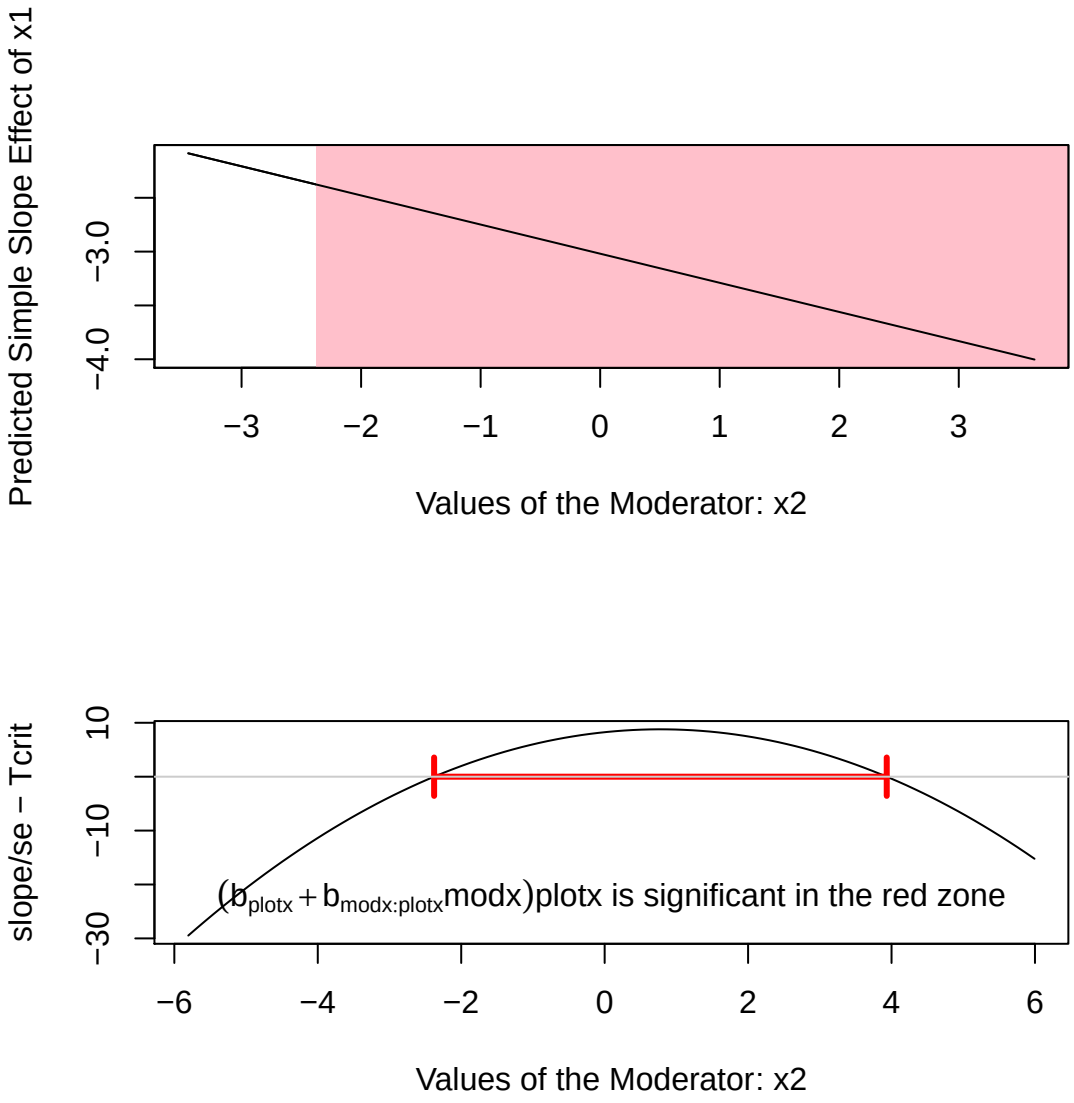


Figure 5: testSlopes for an Interactive Model

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

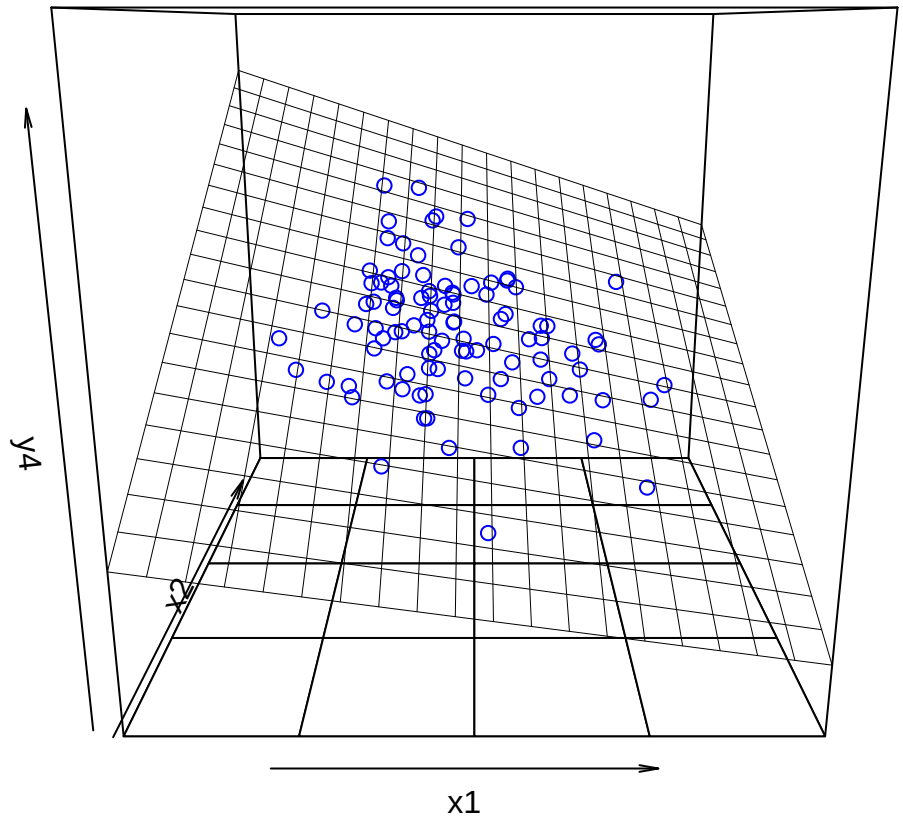
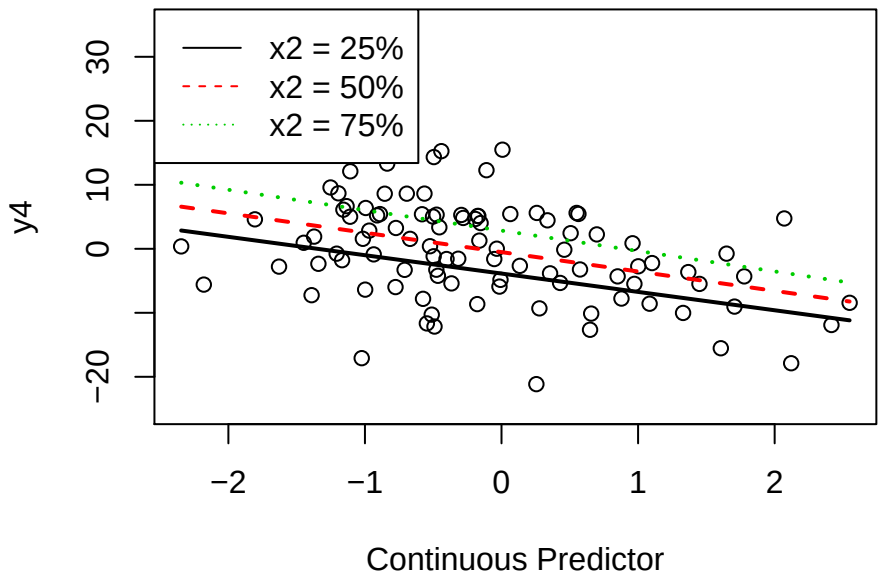


Figure 6: plotPlane for the Interactive Model



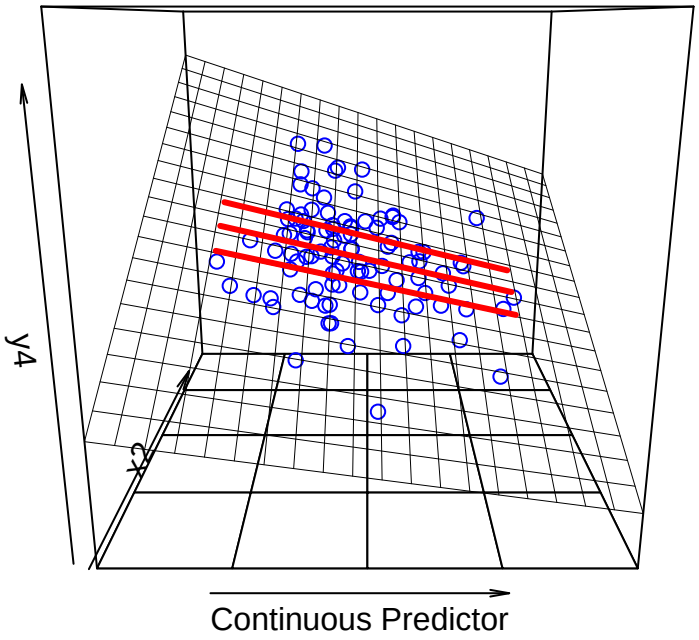


Figure 7: Making plotSlopes and plotPlane work Together

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 variables here have the same names as their
non-centered counterparts, but they are centered, 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().

Observe, the summary statistics of the variables in the design matrix.
        mean std.dev.
y4         0        1
x1         0        1
x2         0        1
`x1:x2`    0        1

Call:
lm(formula = y4 ~ x1 + x2 + `x1:x2`, data = stddat)

Residuals:
     Min       1Q   Median       3Q      Max
-1.67885 -0.42993  0.04379  0.37134  1.11788

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.329e-17  5.717e-02   0.000    1.000
x1          -3.750e-01  5.808e-02  -6.456 4.39e-09 ***
x2           7.221e-01  5.879e-02  12.283  < 2e-16 ***
`x1:x2`     -3.258e-02  5.931e-02  -0.549    0.584
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5717 on 96 degrees of freedom
Multiple R^2: 0.6831,   Adjusted R^2: 0.6732
F-statistic: 68.97 on 3 and 96 DF,  p-value: < 2.2e-16
```

5.2 Mean-centering

In contrast with a standardized regression, a mean-centered regression is one in which one or more of the predictors has been "mean centered" before the design matrix is constructed. The rockchalk package includes a meanCenter function that can, center some or all of the predictors before the design matrix is constructed. It can also standardize those variables *before* the design matrix is constructed.

Table 7: Comparing Ordinary and Standardized Regression

	Not Standardized		Standardized	
	Estimate	S.E.	Estimate	S.E.
(Intercept)	-0.705	(0.469)	0	(0.057)
x1	-3.019*	(0.468)	-0.375*	(0.058)
x2	5.658*	(0.461)	0.722*	(0.059)
x1:x2	-0.271	(0.493)	.	
‘x1:x2‘	.		-0.033	(0.059)
N	100		100	
$RMSE$	4.624		0.572	
R^2	0.683		0.683	
adj R^2	0.673		0.673	

* $p \leq 0.05$

Does a regression model with mean-centered predictors have better statistical properties than a regression that uses the variables as they are originally presented. Some, most notably Aiken and West (1991) and Cohen, et al. (2002) argued that the answer is an emphatic “yes.” In retrospect, it appears this advice was mistaken, especially where the amelioration of multicollinearity is the primary purpose (Echambadi and Hess (2007)). Nevertheless, the issue is of more than passing interest to many applied researchers, who have experienced the frustration of having their results “destroyed” by the inclusion of additional terms involving products of variables that are already in their models.

It is often noted (by researchers and students alike) that the estimates of the ordinary linear regression are affected in surprising ways by the introduction of nonlinear expressions. Suppose we begin with an ordinary linear model.

$$y_i = b_0 + b_1 x1_i + b_2 x2_i + e_i \tag{4}$$

Then we add, for example, a squared term,

$$y_i = b_0 + b_1 x1_i + b_2 x2_i + b3 x2_i^2 + e_i \tag{5}$$

or an “interaction effect”,

$$y_i = b_0 + b_1 x1_i + b_2 x2_i + b3(x1_i \cdot x2_i) + e_i \tag{6}$$

In both of these cases, practitioners have long been bothered by the fact that the estimate of b_1 or b_2 in model (4) might be “statistically significant” (significantly different from 0, that is), but when the last term is added, the standard errors of the estimates grow larger and “nothing is significant anymore.”

Aiken and West (1991) and Cohen, et. al (2002) contended that the apparent instability of the coefficients is a reflection of “inessential collinearity” among the predictors, due to the fact that $x1$ and $x2$ are in fact correlated with the new terms, $x2^2$ or $x1 \cdot x2$. Their recommendation is that practitioners ought to mean-center those predictors, to replace $x1$ by $(x1_i - \overline{x1})$ and $(x2_i - \overline{x2})$. In some cases, it appears as though the use of mean-centered variables does indeed address the multicollinearity problem, making the t-statistics look “bigger” and the p values are smaller.

While the superficial evidence for mean-centering seemed compelling, it turns out to be a complete mirage. Mean-centering does not solve the problem of multicollinearity, it merely changes the point at which we evaluate it. This point is made most emphatically by Echambadi and Hess (2007), who argue that mean-centering has no effect (not one “iota” of an effect!) on multicollinearity.

In order to help students and researchers explore this controversy, the rockchalk includes the function meanCenter. Mean-centering makes it easier to tell, at a glance, the model’s predicted value for the case that is situated at the mean.

Table 8: Comparing Ordinary and Mean-Centered Regression

	Linear		Not Centered		Mean Centered	
	Estimate	S.E.	Estimate	S.E.	Estimate	S.E.
(Intercept)	-534.699*	(94.088)	411.048	(369.767)	484.681*	(16.383)
x1	10.066*	(1.774)	-9.38	(7.564)	9.793*	(1.764)
x2	10.847*	(1.755)	-8.429	(7.498)	11.006*	(1.743)
x1:x2	.		0.39*	(0.147)	0.39*	(0.147)
N	400		400		400	
$RMSE$	307.098		304.808		304.808	
R^2	0.239		0.252		0.252	
adj R^2	0.235		0.246		0.246	

* $p \leq 0.05$

For this example, I use a function to generate data called genCorrelatedData. It is included with rockchalk. The "true model" from which the data is produced is

$$y_i = 2 + 0.1x1_i + 0.1x2_i + 0.2 \cdot (x1_i \times x2_i) + e_i \tag{7}$$

The usual "course of affairs" would observe the following sequence of events. Three regressions will be estimated, they are summarized in Table 8. First (the first column), the researcher "explores" a linear specification,

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

The coefficients of $x1$ and $x2$ appear to be statistically significant, a very gratifying regression indeed. Second (the second column in Table 8), an interaction term is added to the model. That interaction term, the product variables $x1 \times x2$, is estimated in R with

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

This specification leads to a model that includes the main effect variables $x1$ and $x2$, as well as their product, which is labeled $x1 : x2$ in the output. When most of us see that second column for the first time, we think "Holy Cow! My regression went to hell!" The situation does appear dire. While the coefficients for the variables $x1$ and $x2$ did seem to be substantial in the first model, the introduction of the interactive effect renders everything statistically insignificant.

Now comes the "magic" of mean centering. If we replace $x1$ and $x2$ with their mean centered counterparts, and then calculate the interaction variable as the product of those two centered variables, we get a "great" regression, which is presented in the third column of Table 8. Everything appears to be significant, order has been restored in the land of the more-or-less linear model.

```
Model-constructed interactions such as "x1:x3" are built from centered variables
```

There is a good argument (actually an invincible argument), that the mean-centering effect is a complete and total illusion. The first piece of evidence should be that the coefficient of the interactive effect in columns 2 and 3 is identical. The root mean square and R^2 estimates are identical. And, if we look into the situation a little more closely, we find that the models produce identical predicted values! The 3 dimensional plots of the predicted values of the two models are compared in Figure 8.

The curves in Figure 8 are identical, of course. The only difference is that the one on the left, the one for the original non-transformed data, has the "y axis" positioned at the front-left edge of the graph, while the centered one re-positions the y axis into the center of the data. There are two reasons why the mean-centering "seems to" help multicollinearity, when in fact it has no effect at all. First, an interaction model is a nonlinear model. The slope of an effect is different at every point in the X1,X2 plane. This, of course, means that multicollinearity is not a global attribute of the data, but rather it is a local attribute, so that the effect of multicollinearity is more obvious when the slope is flat than when it is steep. Since the regression fit measures multicollinearity at the origin, where the y axis is "stuck into the ground," it only

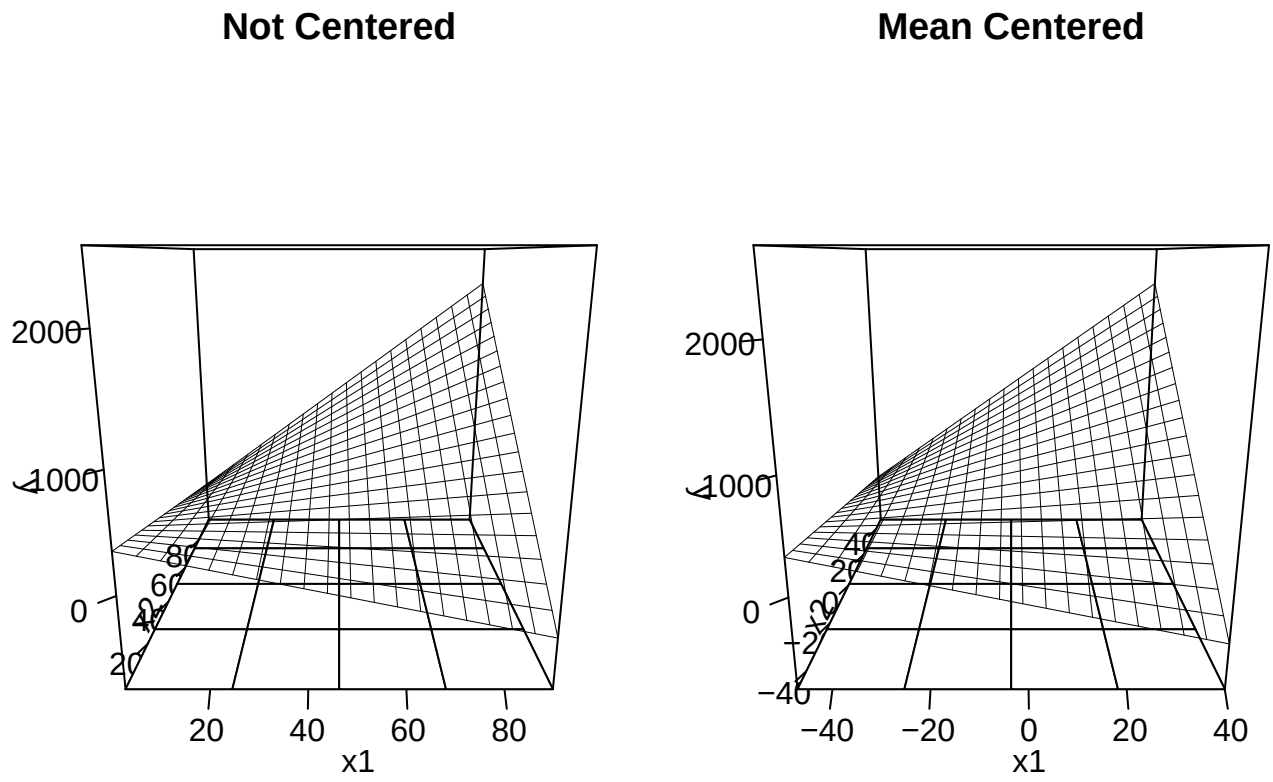


Figure 8: Mean Centered and Uncentered Fits Identical

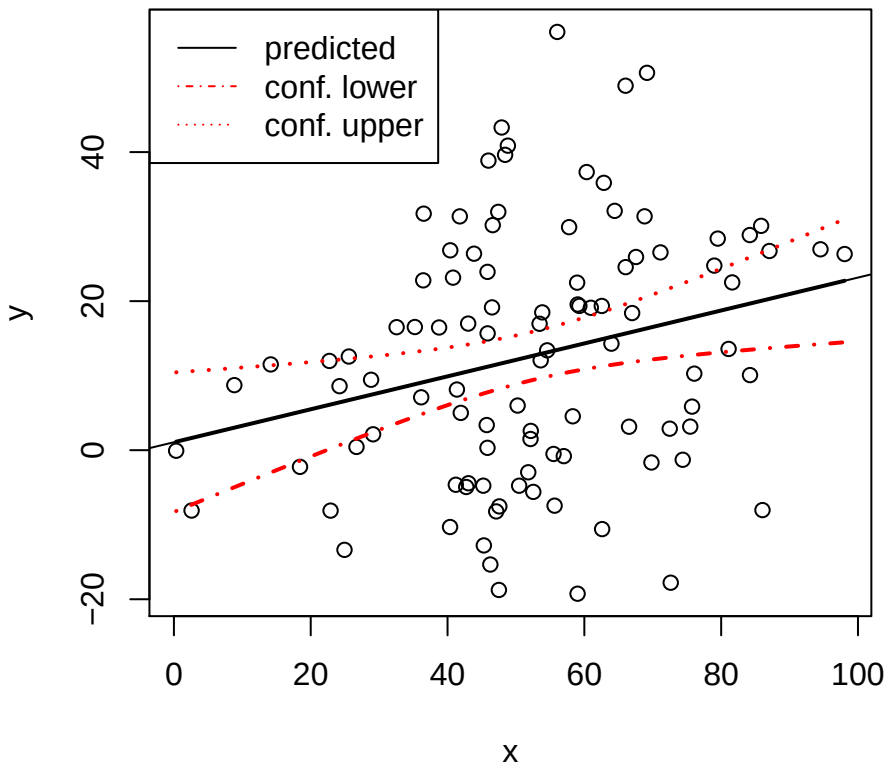


Figure 9: The Hourglass

makes sense that re-positioning the axis would affect our assessment of collinearity. Second, the standard error of predicted values is hour-glass shaped–smaller in the center of the data cloud, wider at the edges. Students of elementary regression have no-doubt seen the confidence interval plot similar to Figure 9. The "mean centered" regression is a snapshot of the standard errors in the small part of the hourglass, while the non-centered regression is a snapshot at the outer edge. They are, of course, the exact same model, and the results differ only superficially.

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. In addition, I have several lectures for an intermediate regression class on this issue and they can be found under `http://pj.freefaculty.org/guides/stat`.

5.3 Residual-centering

The argument against mean-centering is that it makes absolutely no difference. Perhaps the same cannot be said of mean-centering's cousin, "residual-centering."

Residual-centering is another way to deal with the problem that the constructed variable representing the interaction, "$x1 \times x2$," will sometimes cause multicollinearity, exaggerating standard errors, making t-statistics small and p-values big.

We would still like to estimate a model

$$y_i = b_0 + b_1 x1_i + b_2 x2_i + b3(x1_i \times x2_i) + e_i, \tag{8}$$

but we can't. More accurately, we can estimate it, but we don't like the results we get. We don't like the large standard errors and huge p-values.

So here's the plan. If we fit the linear model–with no interaction

$$y = c_0 + c_1 x1 + c_2 x2 + e2_i \tag{9}$$

we get parameter estimates that we like for the effects of $x1$ and $x2$. We will proceed by constraining the fitted coefficients in the full, interactive model so that the main effects remain the same. That is to say, fit 8, but force the fit so that the coefficients of $x1$ and $x2$ match equation 9. Effectively, we impose the restriction that $\hat{b}_1 = \hat{c}_1$ and $\hat{b}_2 = \hat{c}_2$.

How can this be done in a convenient, practical way? We need an estimate of the interaction effect that is orthogonal to (uncorrelated with) both $x1$ and $x2$. That's where residual-centering comes into the picture. Estimate the following regression, in which the left hand side is the interaction product term:

$$(x1_i \times x2_i) = d_0 + d_1 x1_i + d_2 x2 + u_i \tag{10}$$

The residuals from that regression are, by definition, orthogonal to both $x1$ and $x2$. Let's use that residual as a new indicator of the interaction effect. Call it $(x1Xx2)$. Then we fit an equation like (8), but instead of the actual product term $(x1_i \times x2_i)$, we include the residual from the fitted regression (10).

$$y_i = b_0 + b_1 x1_i + b_2 x2_i + b3(x1Xx2) + e_i, \tag{11}$$

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 "residual centered" regression adds the new variable, $(x1Xx2)$, and it leaves the effect coefficients for variables $x1$ and $x2$ unchanged.

Table 9 compares the parameter estimates of the models with interaction terms. One is the usual, non-centered, model, and the others are the mean-centered and residual-centered estimates. One peculiar, important, fact is that the estimate of the interaction coefficient is the same in all three models. They are not just similar. They are identical. The fact that the usual and mean-centered estimates are the same has, of course, been noted in Cohen, et al (2002), but the significance was missed. As argued in the previous section of this essay, the two models have more in common than just that one coefficient. The models–all three of them– are actually identical.

```
Model-constructed interactions such as "x1:x3" are built from centered variables
```

Does the identicality of the interaction estimates indicate that the residual-centered regression is also equivalent to the mean-centered and usual specifications? It appears the answer is, "yes." Emphatically. In order to make this perfectly clear, we need to calculate predicted values for a residual-centered regression model. In rockchalk-1.5.1, a predict method was introduced for that purpose. To calculate a prediction, it is necessary to specify the values of all of the predictors, and then the fitted models for each of the interactions is used to calculate residuals that can be used as interaction terms in the final model.

With predicted values in hand, we can demonstrate the fact that the predicted values from all three methods of estimating interactions are identical. In Figure 10, the three dimensional plots of the three models are presented together. For a given pair of input values $x1_i$ and $x2_i$, all three models offere the same prediction. From that, one must conclude that the three regression specifications are, despite the superficial differences among their estimated coefficients, actually the same.

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 simply competing methods for re-positioning the y axis. The interactive model has a constant mixed partial derivative, so the estimate of the interaction coefficient is the same in all three models. The similarity of the three plots in Figure 10 was, in all honesty, a surprise. Intuitively, it seems as though the residual-centered approach is different because the data that represents the interaction is drawn from the residuals of a preliminary regression. We have methodically re-calculated the predicted values from the residual centered regression. Through whatever method one chooses, the results are the same.

In Figure 11, the predicted values of the mean-centered and residual-centered regressions are plotted against one another. They are perfectly co-incident, as evidenced by the fact that the correlation between them is 1.0.

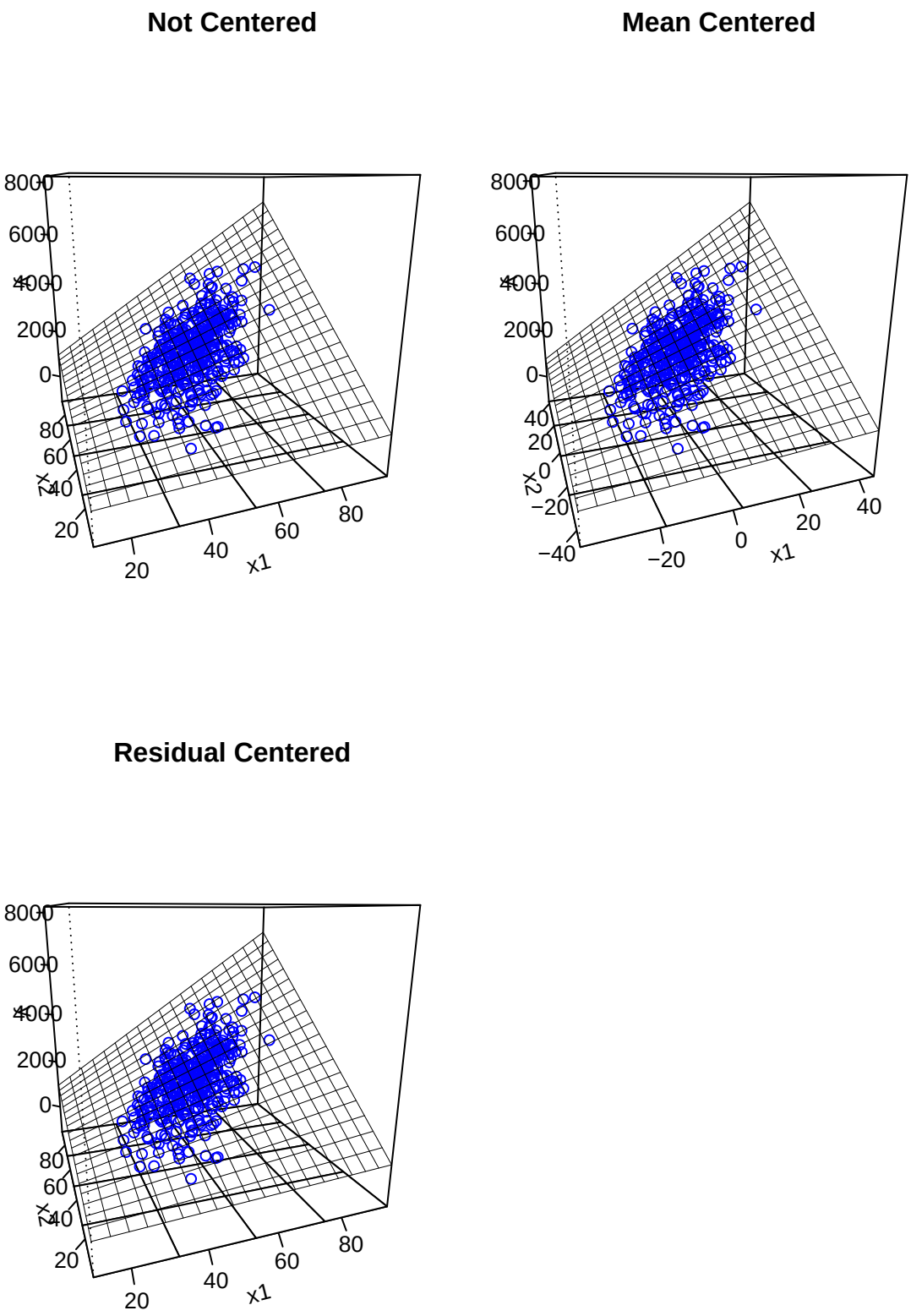


Figure 10: Mean Centered and Uncentered Fits Identical

Table 9: Comparing Ordinary and Residual-Centered Regression

	Linear Estimate (S.E.)	Not Centered Estimate (S.E.)	Mean Centered Estimate (S.E.)	Resid Centered Estimate (S.E.)
(Intercept)	-2075.743*	-144.89	2072.472*	-2075.743*
	(89.533)	(348.944)	(15.962)	(86.169)
x1	40.936*	1.561	40.354*	40.936*
	(1.694)	(7.086)	(1.633)	(1.63)
x2	41.812*	3.504	42.275*	41.812*
	(1.636)	(6.891)	(1.577)	(1.575)
x1:x2	.	0.767*	0.767*	.
		(0.134)	(0.134)	
x1.X.x2	.	.	.	0.767*
				(0.134)
N	400	400	400	400
$RMSE$	304.259	292.825	292.825	292.825
R^2	0.85	0.862	0.862	0.862
adj R^2	0.849	0.861	0.861	0.861

* $p \leq 0.05$

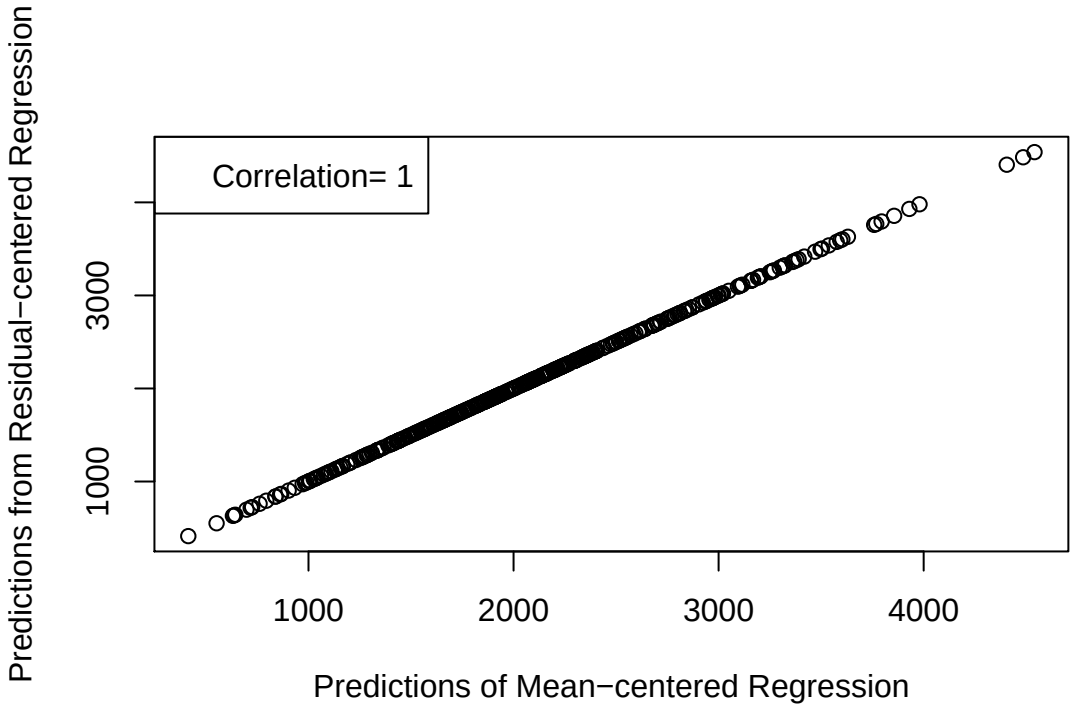


Figure 11: Predicted Values of Mean and Residual-centered Models

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.

Consider the following derivation. 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.

$$\begin{aligned}
\hat{b}_0 &= \hat{c}_0 - \hat{c}_1\overline{x1} - \hat{c}_2\overline{x2} + \hat{c}_3\overline{x1x2} && (17)\\
\hat{b}_1 &= \hat{c}_1 - \hat{c}_3\overline{x2} && (18)\\
\hat{b}_2 &= \hat{c}_2 - \hat{c}_3\overline{x1} && (19)\\
\hat{b}_3 &= \hat{c}_3 && (20)
\end{aligned}$$

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 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 difference between the observed product, $x1_i \times x2_i$ and the predicted value from that model, $\widehat{x1_i \times x2_i}$, is the "residual-centered estimate," which is used as a predictor in place of $(x1_i \times x2_i)$ in equation 12.

$$y_i = h_0 + h_1 x1_i + h_2 x2_i + h_3\{x1_i \times x2_i - \widehat{x1_i \times x2_i}\} + e3_i \tag{23}$$

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

$$\begin{aligned}
y_i &= h_0 + h_1 x1_i + h_2 x2_i + h_3\{x1_i \times x2_i - \hat{d}_0 - \hat{d}_1 x1_i - \hat{d}_2 x2_i\} + e3_i && (24)\\
&= h_0 + h_1 x1_i + h_2 x2_i + h_3\{x1_i \times x2_i\} - h_3\hat{d}_0 - h_3\hat{d}_1 x1_i - h_3\hat{d}_2 x2_i\} + e3_i && (25)\\
&\quad \{h_0 - h_3\hat{d}_0\} + \{h_1 - h_3\hat{d}_1\}x1_i + \{h_2 - h_3\hat{d}_2\}x2_i + h_3\{x1_i \times x2_i\} + e3_i && (26)
\end{aligned}$$

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, 23, we can re-generate the coefficients of the other models like so:

$$b_0 = \hat{c}_0 - \hat{c}_1\overline{x1} - \hat{c}_2\overline{x2} + \hat{c}_3\overline{x1x2} = h_0 - h_3\hat{d}_0 \quad (27)$$

$$b_1 = \hat{c}_1 - \hat{c}_3\overline{x2} = h_1 - h_3\hat{d}_1 \quad (28)$$

$$b_2 = \hat{c}_2 - \hat{c}_3\overline{x1} = h_2 - h_3\hat{d}_2 \quad (29)$$

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.