Chapter 7

# Multiple Regression

## 7.1 Graphs for Understanding Complex Relationships

This section contains information about how to create three types of graphs that can be helpful in understand relationships among multiple variables. Look through these and see whether you think any might work for your project.

### *7.1.1 Coplots*

A coplot is a "conditioning" plot that shows the effects of a third variable in a step-by-step fashion on scatterplots of two variables together. This plot can perhaps best be understood by looking at an example from geography in Figure 7.1 (this coplot and several others can be recreated by typing `example(coplot)` in R; make sure to click on the Graphics Device to advance to the next graphic).

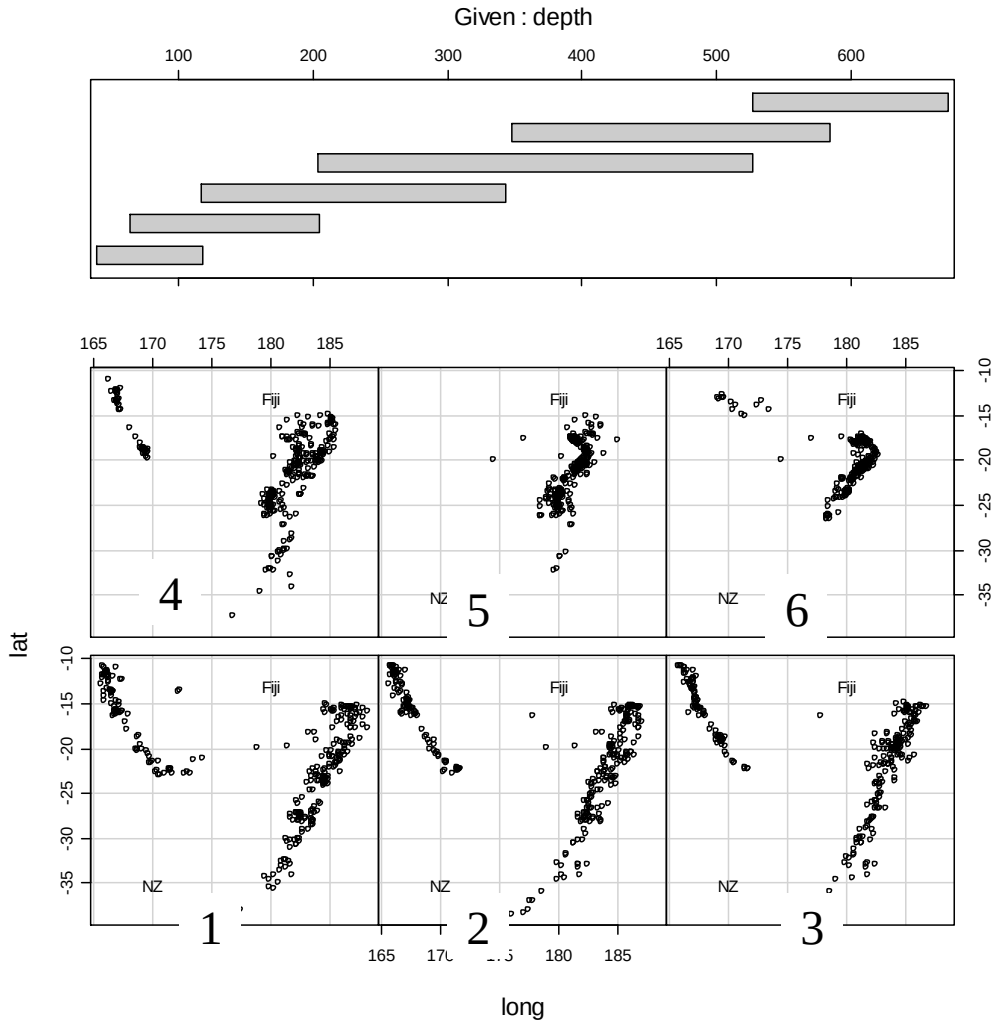


*Figure 7.1* Coplot for the quakes data in the base R system.

The conditioning variable in the figure is depth of the ocean, and the scatterplots show the location of 1,000 earthquakes near Fiji at various latitude and longitude points. Each bar of the six conditioning variables (depth) corresponds to one scatterplot. This correlation begins on the left-hand side for depth, and this first bar (around 100 kilometers) corresponds to the scatterplot on the bottom row, left-hand corner. The scatterplots are then read from left to right, and from bottom to top. Thus the fourth depth bar (spanning from 200 to 500 kilometers) matches up with the left-most scatterplot on the upper row (no. 4). Physically what is seen as the eye travels along the scatterplots from 1 to 6 is a concentration of the earthquakes into a smaller area as the depth of the ocean increases. Note that the conditioning bars overlap so that there is a sense of physically moving deeper and deeper as one's eyes skim over the scatterplots.

The previous coplot was made with three variables: depth, latitude, and longitude of earthquakes. Coplots can also be constructed with four variables. Figure 7.2 shows a coplot (again, taken from R's example coplots) of how state region and levels of illiteracy (percentage of population) affect the interaction of income and life expectancy. This coplot also shows how a smooth line (Loess line) can be drawn over the data to give an indication of the trend of the data.

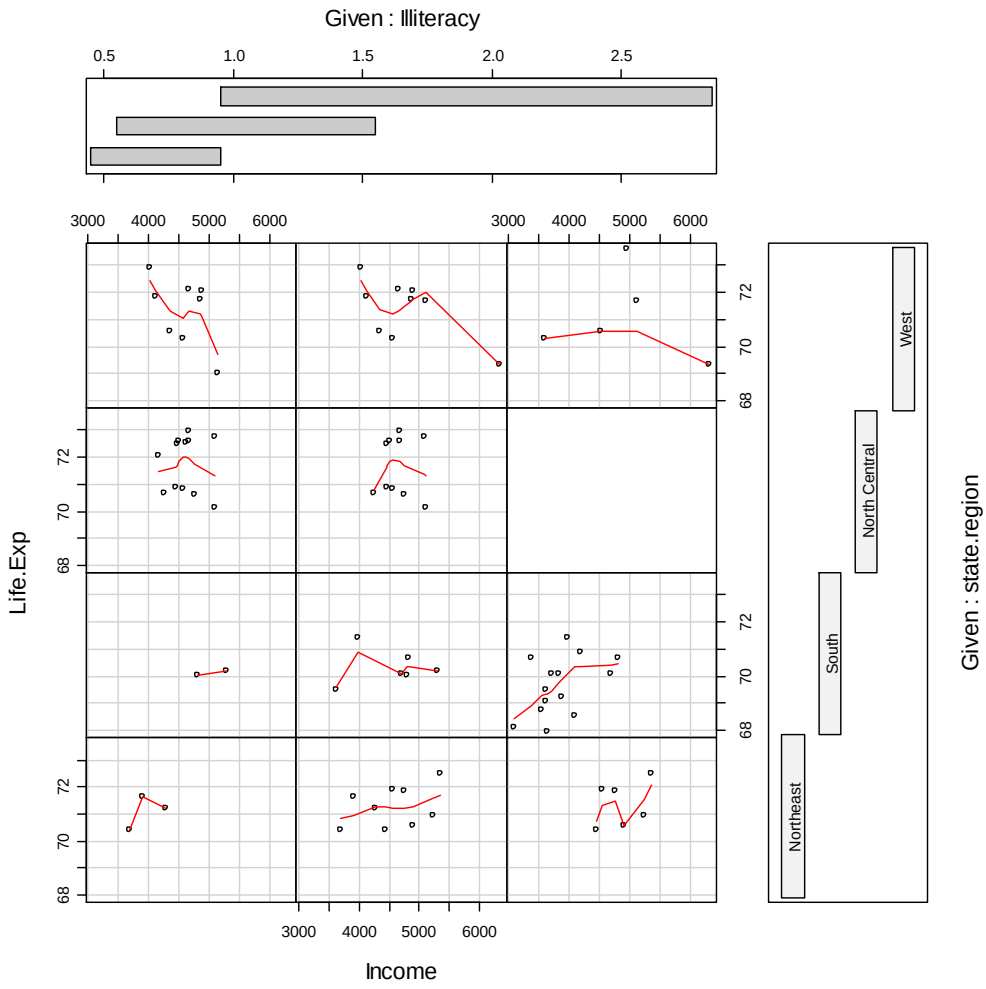


*Figure 7.2* Coplot for the data.frame(state.x77) data in the base R system.

There are few data points in each scatterplot, so we would need to be careful about drawing strong conclusions from the data, but in the Northeast and South regions of the US we see somewhat of a trend toward higher life expectancies with higher income, whereas in the West there seems to be a trend of lower life expectancies with higher income. None of these seem to show any clear pattern with increases in illiteracy rates.

Now using the Lafrance data (if you are working along with me, import the SPSS file "LafranceGottardo.sav" and name it lafrance), we will look at how nonverbal memory affects the relationship between L2 phonological awareness and L1 word reading scores in first grade.

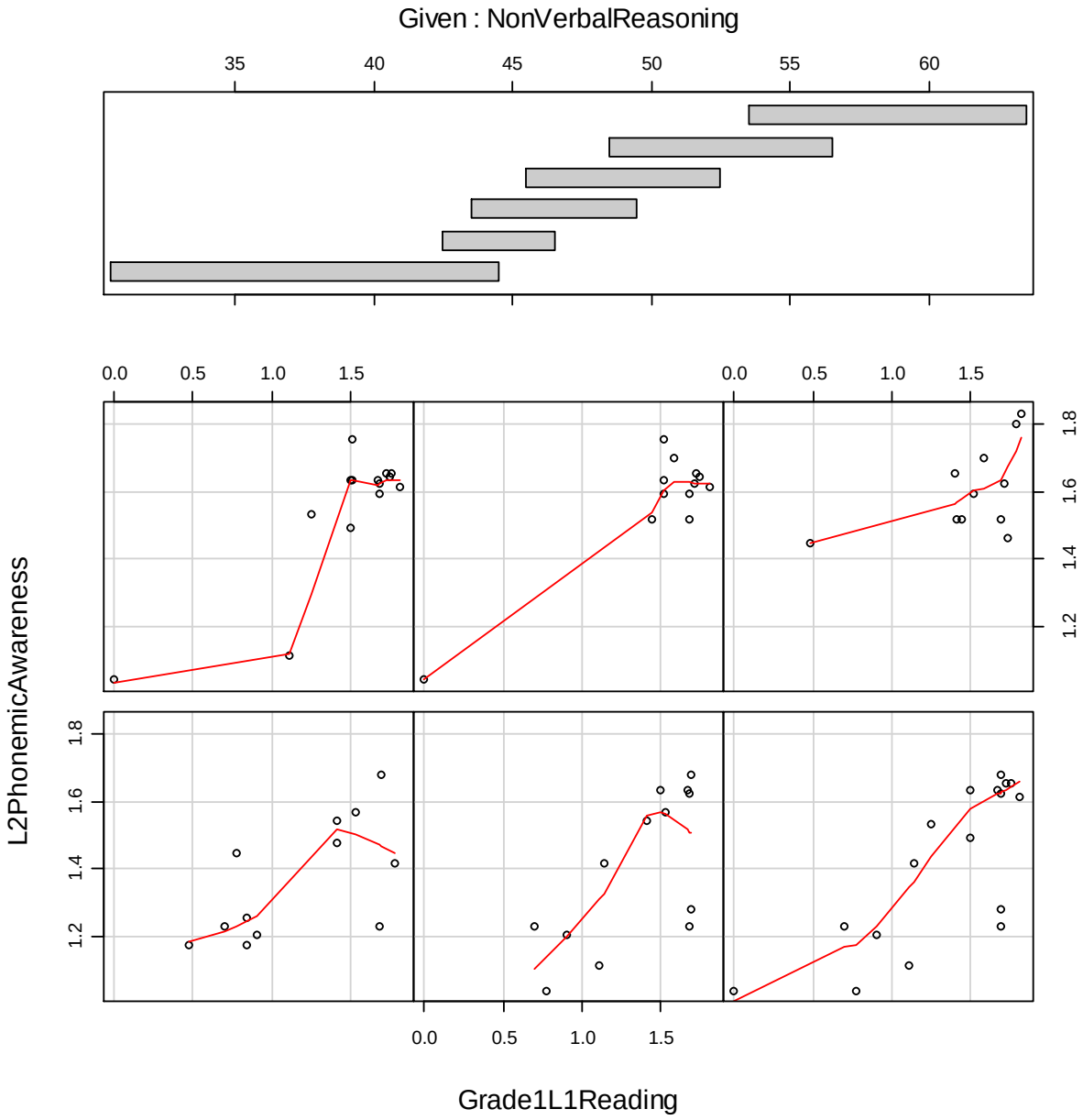


*Figure 7.3* Coplot for three variables in the Lafrance and Gottardo (2005) data set.

The coplot in Figure 7.3 roughly shows that reading ability (Grade1L1Reading) and phonological awareness (L2PhonemicAwareness) are positively correlated in that, as phonological awareness increases, word reading becomes better. In other words, the Loess lines generally show an upward trend in each scatterplot. The conditioning variable of nonverbal reasoning just seems to show that there is more variability in levels of phonological awareness and word reading ability when nonverbal reasoning is lower. That is, the bottom row of scatterplots shows more spread over both reading ability and phonological awareness, while the top row, indicating higher levels of nonverbal reasoning, is more tightly clustered among higher word reading and higher phonological awareness levels (except for one apparent outlier in panels 4 and 5).

Coplots are not available in R Commander, so you must use R Console to obtain them. I used the following command to obtain the coplot in Figure 7.3.

| coplot(L2PhonemicAwareness~Grade1L1Reading\|NonVerbalReasoning, panel= function(x,y,...) panel.smooth(x,y,span=.8,...),data=lafrance) | |
|---|---|
| coplot(A~B\|C) | The coplot() command needs to have at least three arguments. The argument after the upright bar is the conditioning variable, the one that will be seen in bars at the top of the graph. |
| panel=function(x,y,…) | Because coplot is a high-level function that draws more than one plot (really, a matrix of scatterplots), using the "panel" functions is one way to perform an action that will affect each plot (Murrell, 2006). Here, the panel function is specified as the smoother that follows (notice there is no comma in between the function and the panel.smooth command). |
| panel.smooth(x,y,span=.8,...) | According to Murrell (2006), panel.smooth is a predefined panel function that will add a smooth (Loess) line through points. The span affects how detailed the line is, with larger numbers spreading over a wider number of points and thus being smoother, while smaller numbers take more details into account and result in a choppier line. |
| data=lafrance | This obviously specifies what data set the variables come from. |

*Creating a Coplot*

The basic R code for this command with *three* variables is:

coplot(L2PhonemicAwareness~Grade1L1Reading|NonVerbalReasoning, panel= function(x,y,...) panel.smooth(x,y,span=.8,...),data=lafrance)

The basic R code for this command with *four* variables is:

coplot(L2PhonemicAwareness~Grade1L1Reading|NonVerbalReasoning*NamingSpeed, panel= function(x,y,...) panel.smooth(x,y,span=.8,...),data=lafrance)

Put the conditioning variable(s) after the upright line ("|").

### *7.1.2 3D Graphs*

Another way of looking at three variables at a time is to use a 3D graph. There are several libraries that can draw 3D graphics, but the coolest and easiest to use is definitely the one available through R Commander. In R Commander, choose GRAPHS > 3D GRAPH > 3DSCATTERPLOT. This command will not be available unless you have a data set active that has at least three variables! The dialogue box is shown below. If you are working with me I'm still using the lafrance data set.

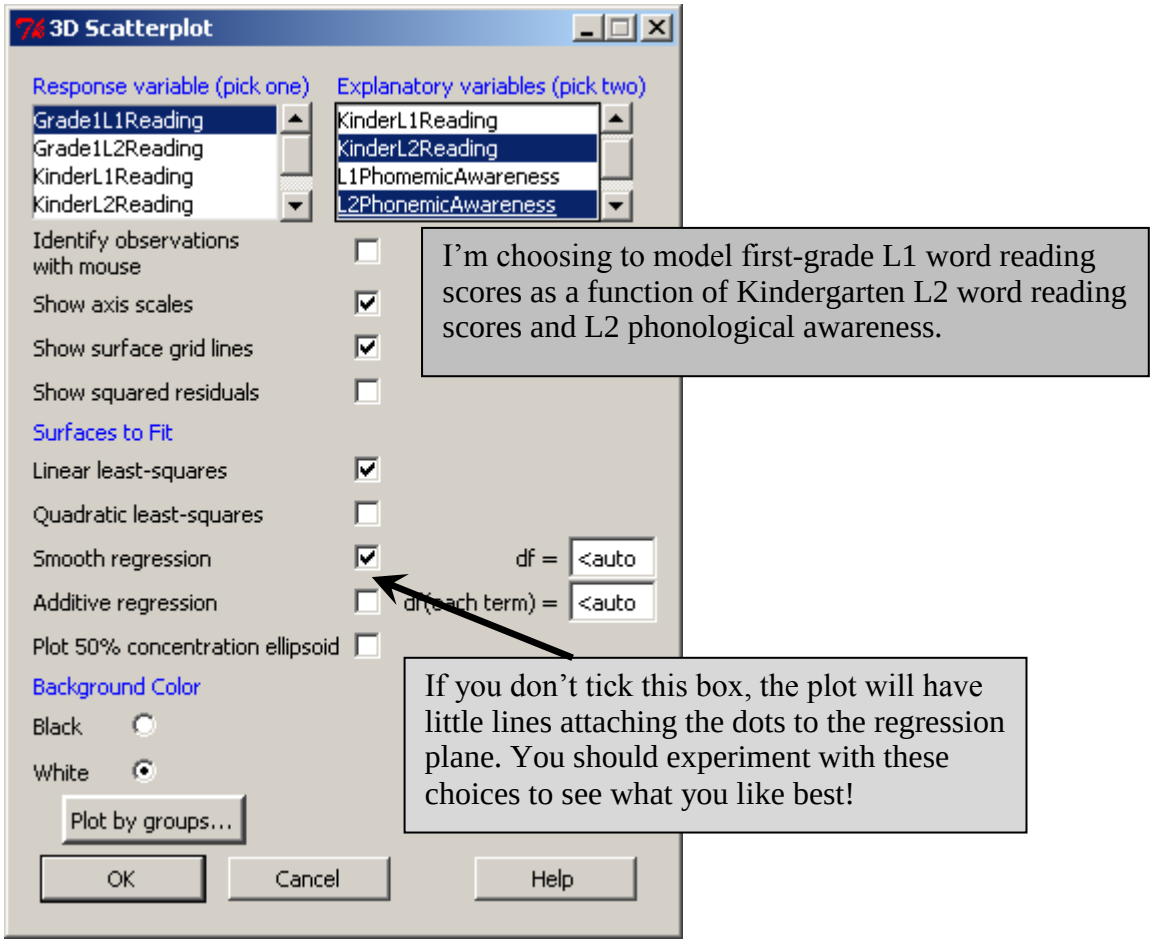


*Figure 7.4* Three-dimensional scatterplot dialogue box in R Commander.

After you press OK, an RGL device will then open up, and you can move the 3D object with your mouse to rotate it to various positions. The graphs below were screen shots of two positions I found informative, but they cannot do justice to the ability to rotate the graph around that you get in the actual program.

You can save the graph as the RGL file by going back to the Graphs drop-down menu, choosing 3D graph again, and then choosing SAVE GRAPH TO FILE (you must do this while your RGL file is still open). You can also identify particular points by going back to the Graphs menu with the 3D graph still open (GRAPHS >3D GRAPH > IDENTIFY OBSERVATIONS WITH MOUSE).

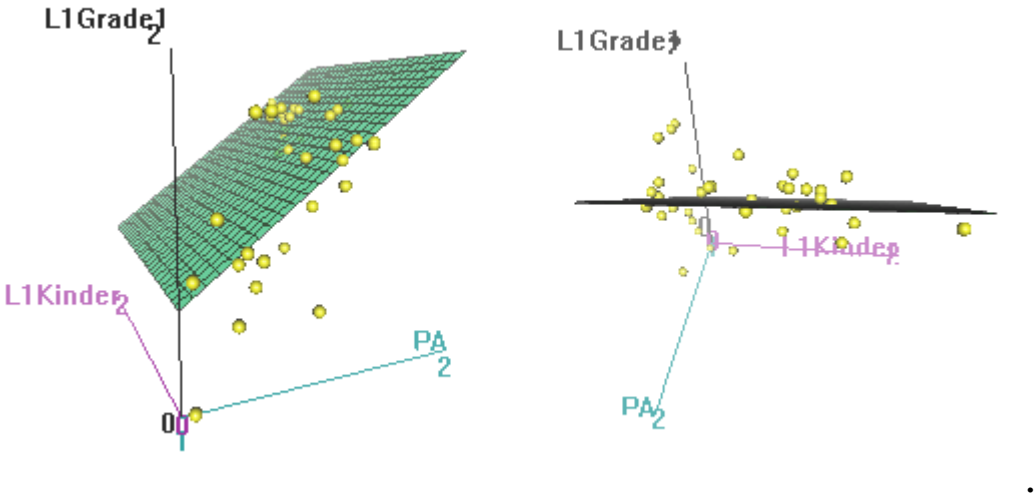


*Figure 7.5* 3D scatterplots in R.

> Tip: If you have a mouse with a wheel, rolling the wheel forward will make the graph smaller (recede it into the background) and rolling it backward will enlarge your view of the graph.

What can be deduced from the 3D scatterplot is that scores on the first-grade reading test are better when phonological awareness is increased. The regression plane is fairly flat when looking from the Kindergarten word reading scores axis, however, seeming to show there isn't a lot of effect of this score on the other two.

*Creating a 3D Plot*

In R Commander, go to GRAPHS > 3D GRAPH > 3D SCATTERPLOT. Choose one response variable and two explanatory variables. Your 3D graph will open up and you can move it around with your mouse.

The basic command in R is:

```
scatter3d(lafrance$KinderL2Reading, lafrance$Grade1L1Reading,
  lafrance$L2PhonemicAwareness, fit=c("linear","smooth"), bg="white",
  axis.scales=TRUE, grid=TRUE, ellipsoid=FALSE, xlab="KinderL2Reading",
  ylab="Grade1L1Reading", zlab="L2PhonemicAwareness")
```

### *7.1.3 Tree Models*

Tree models give a more intuitive sense of what is going on with the data than the regression coefficient numbers of a multiple regression. Crawley (2002) says tree models are useful for cases where there are many explanatory variables and you don't know which ones to select. It takes a little bit of effort to understand how tree models work, but once you have figured them out they can be quite useful for deciding what variables to include in a regression and what types of interactions are taking place. For the code below to work properly you'll need to import a data frame that contains the response variable that you want to model in the first column. I have imported the SPSS file "Lafrance5.sav," which contained only the five

explanatory variables that Lafrance and Gottardo (2005) investigate with Grade 1 L1 reading performance as the response variable. I named it lafrance5.

The code for the tree model is:

```
library(tree)
model=tree(lafrance5)
plot(model)
text(model)
```

The model will assume that the first column contains the response variable that you want to model, but if it does not you can specify which variable is the response variable using the regression model notation, where the "tilde dot" following the response variable means to include all other variables in the data frame as explanatory ones:

```
model=tree(Grade1L1ReadingPerformance~., lafrance.tree)
```

By the way, if you run this model yourself, you will find I have changed the variable names for Figure 7.6 to make them more informative.

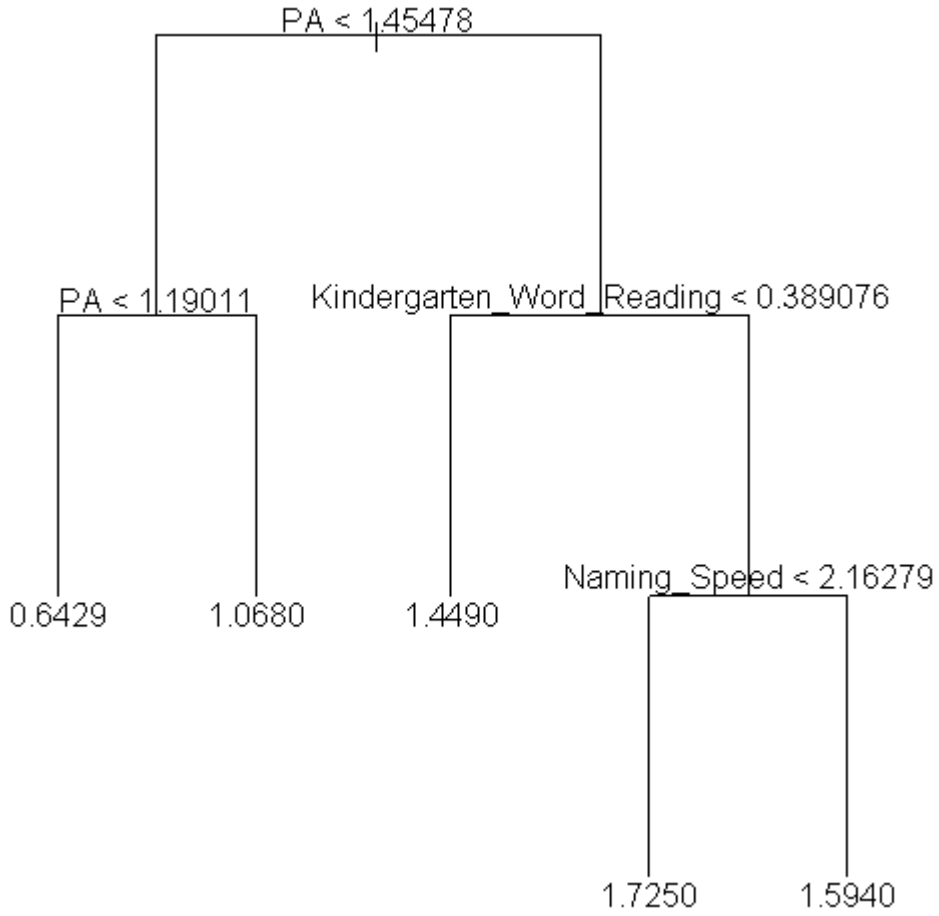


*Figure 7.6* Tree model for some variables in the Lafrance and Gottardo (2005) set.

In order to understand the tree figure in Figure 7.6, you should know that all of the numbers at the terminal nodes refer to the response variable Grade 1 L1 reading performance. Looking

at the tree you should follow a path from the top (called the root) to the nodes (called the leaves). The first important split in the data is between those whose phonological awareness (PA) is below 1.45 and those who are above it. For those who have lower levels of phonological awareness (on the left branch of the first node), no other variable is important, as the next split also occurs with the same variable of phonological awareness. The group can be split into those who have phonological awareness lower than 1.19 and those who have phonological awareness higher than that. The mean score on the Grade 1 L1 word reading test is .64 for those whose phonological awareness is lower than 1.19, and it is 1.07 for those whose phonological awareness is higher than 1.19.

For those who have higher levels of phonological awareness (above 1.45, on the right branch of the first node), scores on the reading test in Kindergarten are the next most important variable. For those who scored below .39 on the Kindergarten measure, their mean score on the Grade 1 word reading test is 1.45. For those children who scored higher than .39 on the Kindergarten reading test, naming speed is the next most important variable. Those children who scored lower on naming pictures (less than 2.16) got higher scores on the Grade 1 word reading test (mean=1.73). Those who scored higher on naming speed had an average score of 1.59 on the Grade 1 L1 reading measure.

The data can also be seen in a different way by using the print command:

```
print(model)
```

```
1) root 37 7.36000 1.3670
2) PA < 1.45478 12 2.75400 0.8911
4) PA < 1.19011 5 0.72190 0.6429 *
5) PA > 1.19011 7 1.50400 1.0680 *
3) PA > 1.45478 25 0.58540 1.5950
6) Kindergarten_Word_Reading < 0.389076 7 0.05779 1.4490 *
7) Kindergarten_Word_Reading > 0.389076 18 0.32070 1.6520
14) Naming_Speed < 2.16279 8 0.04694 1.7250 *
15) Naming_Speed > 2.16279 10 0.19760 1.5940 *
```

A star marks the terminal nodes for this data. The numbers are the node numbers, and they are labeled by the variable which made the split. So node 2 was split by Phonological Awareness, and this was split for the criteria of being below 1.45478. The number of cases going into the split for node 2 was 12. The next number (2.754) is the deviance at the node, and the last number is the mean score of the response variable (here, first-grade L1 word reading scores) at that node. The lowest score for first-grade word reading was at node 4, and the highest was at node 14.

This model can help us understand better how phonological awareness is involved in scores on the response variable. Crawley (2002, p. 585) says, “This kind of complex and contingent explanation is much easier to see, and to understand, in tree models than in the output of a multiple regression.” The results of the tree model would lead us to choose to include phonological awareness, Kindergarten reading scores, and naming speed in our model, and leave out working memory and nonverbal reasoning.

## 7.2 Application Activities with Graphs for Understanding Complex Relationships

### *7.2.1 Coplots*

1. Use the Lafrance and Gottardo (2005) data (import the SPSS file "LafranceGottardo.sav" and name it lafrance; it has 40 rows and 9 columns). Do a three-variable coplot with first-grade reading measures in L1 (GradeL1Reading) and working memory measures (WorkingMemory) conditioned by L2 phonological awareness (L2PhonemicAwareness) (put the conditioning, or given, variable after the vertical line in the equation). Be sure to include the smooth line. What happens to phonemic awareness as working memory values increase?

2. Use the same data set as in activity 1. Do a three-variable coplot with L2 reading measures from kindergarten (KinderL2Reading) and first grade (Grade1L2Reading) conditioned by L2 phonological awareness (L2PhonemicAwareness) (put it after the vertical line in the equation). What effect does increasing L2 phonological awareness seem to have on the relationship between word reading measures in Kindergarten and first grade?

3. Use the Larson-Hall (2008) data set (use SPSS file LarsonHall2008.sav, importing as larsonhall2008). Explore the relationship between the age that early learners began studying English (earlyage) and their performance on a pronunciation measure (rlwscore) and grammar measure (gjtscore) as adults. Use age as the conditioning variable (put it after the vertical line in the equation). What effect does increasingly later starting age have on outcomes?

### *7.2.2 3D Plots*

1. Use the Lafrance and Gottardo (2005) data set. Create a 3D plot with L2 reading measures from first grade (Grade1L2Reading) as the response variable, conditioned by L1 phonological awareness (L1PhonemicAwareness) and L2 reading measures from Kindergarten (KinderL2Reading). What trend in the data can you see?

### *7.2.3 Tree Models*

1. Lafrance and Gottardo (2005). Use the LafranceGottardo.sav file (not the Lafrance5.sav), which has nine variables. Set up the Grade 1 L2 reading performance to be the response variable (Grade1L2Reading) and note which variables the tree model says are explanatory out of the eight that are entered. Run a tree model again, excluding all other measures of reading performance besides those which turned out to be explanatory in the first run (differences from the first model will be minimal, but I want you to practice cutting out extraneous variables!).

2. Use the Dewaele and Pavlenko (2001–2003) data set. Use the file concerning the use of swear and taboo words, which I have put into the BEQ.Swear.sav SPSS file (import it as beqSwear). Dewaele (2004) examined the frequency of use of swear words in multilinguals' languages and the emotional weight they carry in each language (weight1, weight2) with a linear regression involving age, age of onset of acquisition for each language, self-rated proficiency in four areas, and frequency of language use. Dewaele (2004) looked at attriters only, and we will not recreate his regression. I have put the relevant variables into this file, plus a couple more that looked interesting, such as level of educational degree (degree) and number of languages, frequency of swearing in L1 and the weight it carries in L1. Use this data set to examine one tree model for the frequency of swearing in L2 (swear2) and another for the weight it carries in L2 (weight2), and report on what variables seem to be pertinent to explaining how often multilinguals swear in their second language and what weight they give to swear words.

## 7.3 Doing the Same Type of Regression as SPSS

This section of the book will explain how to fit a model of the type that many software programs, such as SPSS for example, by default apply to a regression. I am not saying that this is the best approach to your data; in fact, I think it is not. However, I understand my readers may want to know how to recreate the type of analysis found in those software programs, and in any case it's a fine place to start thinking about how regression is conducted. We will want in the end, though, to go beyond this simplistic thinking and begin thinking about the *best* fit for the data.

Fitting a regression model in R is extremely easy in the sense that writing the regression equation is quite intuitive once you understand how regression works. For example, to fit the five-factor model that SPSS would create for a standard regression using the Lafrance and Gottardo data, one would simply tell R to create a linear model (use the lm command) with the five explanatory factors:

```
model=lm(Grade1L1ReadingPerformance~NonverbalReasoning+KinderL2Reading
Performance+NamingSpeed+WorkingMemory+PhonologicalAwarenessInL2,
data=lafrance5)
```

In other words, you put the response variable in the first position of the parentheses, and then model a fit that involves all of the other five variables just being added together. In order to obtain the information about R2 and regression coefficients that you'll need to report for a multiple regression, you can simply require a summary of the model:

```
summary(model)
```

some results deleted . . .

```
Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                0.156062   1.979867   0.079  0.93768
NonverbalReasoning        -0.003540   0.008471  -0.418  0.67888
KinderL2ReadingPerformance 0.073145   0.134886   0.542  0.59151
NamingSpeed               -0.310744   0.638273  -0.487  0.62979
WorkingMemory              0.025168   0.056629   0.444  0.65982
PhonologicalAwarenessInL2  1.322462   0.471151   2.807  0.00857 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3158 on 31 degrees of freedom
  (3 observations deleted due to missingness)
Multiple R-squared: 0.5798,     Adjusted R-squared: 0.512
F-statistic: 8.555 on 5 and 31 DF,  p-value: 3.538e-05
```

The output contains almost all of the information you'll need to report:

- unstandardized coefficients under the "Estimate" column, including the intercept of the regression equation in the row labeled "Intercept"
- results of a t-test for whether each coefficient uniquely contributes to the equation under the "t value" and "Pr(>|t|)" columns
- the $R^2$ effect size estimating how much of the variance in scores on the response variable are accounted for by the explanatory variables in the second-to-last line ("Multiple R-squared")
- results of an ANOVA for testing whether the predictive power of the model is equal to zero in the last line

You can obtain the 95% CI of the unstandardized coefficients with the confint command:

```
confint(model)
```

```
                  2.5 %     97.5 %
(Intercept) -3.88190436 4.19402818
NR          -0.02081698 0.01373652
KL2WR       -0.20195750 0.34824771
NS          -1.61251062 0.99102199
WM          -0.09032847 0.14066421
PAL2         0.36154270 2.28338092
```

If you'd like to pull out the standardized regression coefficients, you can do this by putting the scale command, which standardizes all variables to mean 0 and sd 1, around every variable, like this:

```
model.standard=lm(scale(Grade1L1ReadingPerformance)~scale(Nonverbal
Reasoning)+scale(KinderL2Reading Performance)+ scale(NamingSpeed)+
scale(WorkingMemory)+scale(PhonologicalAwarenessInL2), data=lafrance5)
```

Now in the summary the estimates are standardized regression coefficients.

```
Coefficients:
                                  Estimate
(Intercept)                       -0.03730
scale(NonverbalReasoning)         -0.05672
scale(KinderL2ReadingPerformance)  0.09386
scale(NamingSpeed)                -0.08883
scale(WorkingMemory)               0.07049
scale(PhonologicalAwarenessInL2)   0.61181
```

The reason most people would want standardized coefficients is because they have read that, if coefficients are standardized, their effect sizes can be compared. John Fox (R help archives, February 7, 2007) cautions that it is not really appropriate to try to compare the standardized coefficients of different variables in a regression, because standardizing them does not guarantee that they are really of the same magnitude. Most R statisticians stay away from standardized coefficients.

A much better way to compare the relative importance of terms in a regression is to use the calc.relimp command in the relaimpo library. This command returns five different estimates for assessing relative importance of terms in linear regression (Grömping, 2006). The estimates assess each individual term's contribution to the multiple regression model, and the lmg metric is the recommended one (pmvd is also recommended but is not available in the US, so is not available to me). The lmg metric is appropriate for assessing relative importance no matter what order the term comes in the model, and is akin to the squared semipartial correlation ($sr^2$). This measure "expresses the unique contribution of the IV to the total variance of the DV" (Tabachnik & Fidell, 2001, p. 140), and because of shared variances between the explanatory variables it often does not actually sum up to the total $R^2$ for the entire model. So the relative importance metrics are effect sizes for each term in the regression model, and are appropriate for making comparisons as to how much of a contribution each term adds to the overall $R^2$ value. Even better, they are simple to calculate:

```
library(relaimpo)
calc.relimp(model)
```
data deleted . . .

```
Relative importance metrics:

                                lmg
NonverbalReasoning          0.02743436
KinderL2ReadingPerformance  0.10740872
NamingSpeed                 0.10360432
WorkingMemory               0.07554998
PhonologicalAwarenessInL2   0.26581778
```

The relative importance metrics show that Phonological Awareness was the most important factor by far out of the five, contributing about 27% of the variance. However, Kindergarten L2 reading and naming speed each contributed about 10%.

The model created above provides the "standard regression" that Tabachnick and Fidell (2001) describe, whereas the "sequential regression" that was the focus of an SPSS analysis of Lafrance and Gottardo's data could easily be done in R just by creating five different models, each with increasing numbers of variables, like this:

| | |
|---|---|
| model1= | lm(Grade1L1ReadingPerformance~NonverbalReasoning, data=lafrance5) |
| model2= | lm(Grade1L1ReadingPerformance~NonverbalReasoning +KinderL2Reading, data=lafrance5) |
| model3= | lm(Grade1L1ReadingPerformance~NonverbalReasoning +KinderL2Reading +NamingSpeed, data=lafrance5) |
| model4= | lm(Grade1L1ReadingPerformance~NonverbalReasoning +KinderL2Reading +NamingSpeed+WorkingMemory, data=lafrance5) |
| model5= | lm(Grade1L1ReadingPerformance~NonverbalReasoning +KinderL2Reading +NamingSpeed+WorkingMemory + PhonologicalAwarenessInL2,data=lafrance5) |

The change of $R^2$ could be calculated by subtracting the smaller model's $R^2$ from the larger model's when you call for the summary of each model. The point of this section is to show that linear multiple regression can easily be done as well in R as in SPSS. The online document "Multiple Regression.Finding the best fit" will show how regression can be done better in R than in SPSS.

Last of all, I want to point out that doing this modeling in R Console seems to me easier and more transparent than doing it in R Commander, but there are some advantages to doing it in R Commander, which are that, once you have your model created, menu choices will lead you to many things you can do with your model, including diagnostics and graphs.

So if you want to do regression modeling in R Commander, the first step is to create your model. In R Commander, you do this by first making sure the data set you want is active. Next, choose STATISTICS > FIT MODELS > LINEAR REGRESSION. As you can see from the dialogue box in Figure 7.7, you can now pick your response variable and explanatory variables. In keeping with the information given in this section, this regression model will be simple addition of main effects, as in the original model:

```
RegModel.1 <-
lm(Grade1L1ReadingPerformance~KinderL2ReadingPerformance+NamingSpeed+
NonverbalReasoning+PhonologicalAwarenessInL2+WorkingMemory,
data=lafrance5)
```

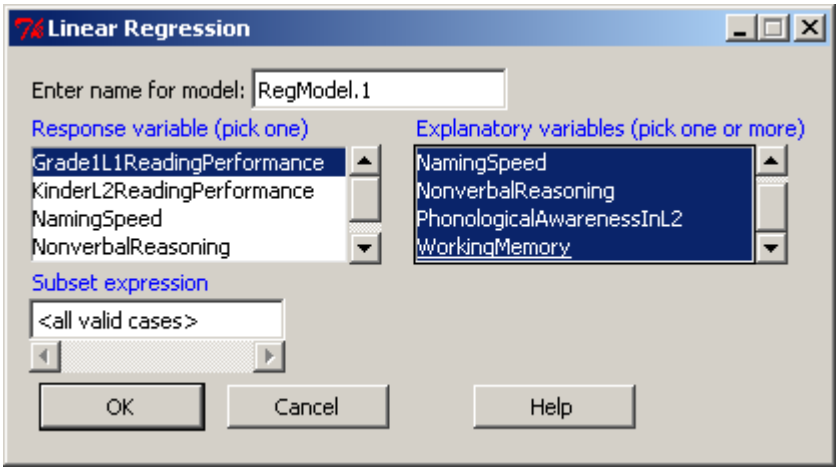


*Figure 7.7* Dialogue box for simple regression in R Commander.

Once you press OK, you will now have a model called "RegModel1" (you could of course have changed this name if you'd liked). Now you can go to the Models drop-down menu. If this is your first model you don't need to do anything special, but if you have other models and need to select which one is active you can pick MODELS > SELECT ACTIVE MODEL (see Figure 7.8).

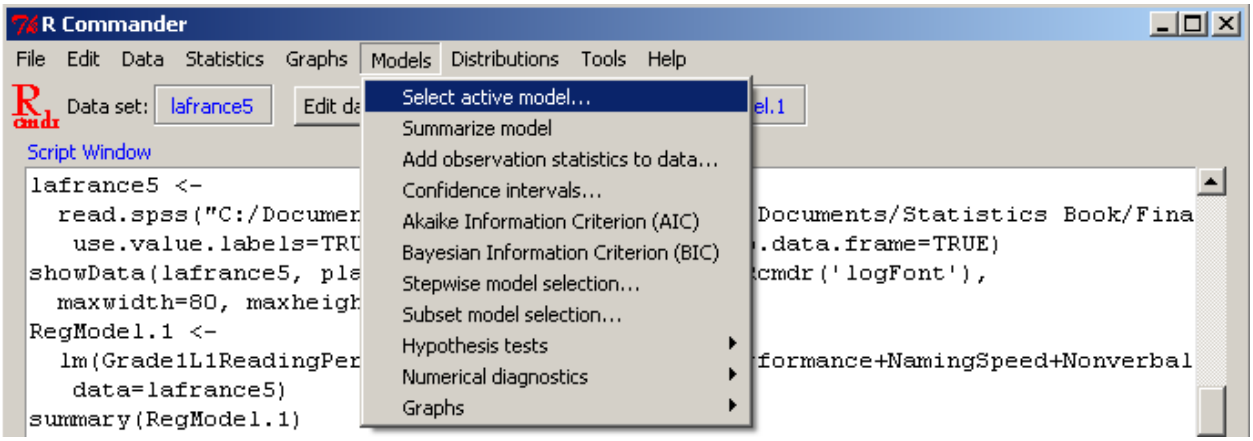


*Figure 7.8* Creating regression models in R Commander.

You can now use any drop-down menu under the Models menu. The following table gives an idea of all of the things you can do.

| *Models Menu* | |
|---|---|
| **Summarize model** | Returns coefficients with t-tests, residual standard error, $R^2$ value, ANOVA F test. |
| **Add observation statistics to data** | Lets you add columns of data to the end of the active data set, including fitted values, residuals, studentized residuals, hat-values, Cook's distances, and observation indices. |
| **Confidence intervals** | Returns confidence intervals for coefficient parameters. |
| **Akaike information criterion (AIC)** | Returns this number for the current model. |
| **Bayesian information criterion** | Returns this number for the current model. |

| | |
|---|---|
| **(BIC)** | |
| **Stepwise model selection** | Lets you select between stepwise models that go backward, forward, and a combination of backward/forward or forward/backward. You can also choose whether you want the BIC or AIC to be the criterion for stepping. |
| **Subset model selection** | Model selection by exhaustive search, forward or backward stepwise, or sequential replacement (read help for more details). |
| **Hypothesis tests** | |
| ANOVA table | Returns ANOVA output for the model. |
| Compare two models | Lets you perform an ANOVA for two models (used to see if the simplified model is better; see online document "Multiple Regression.Further steps in finding the best fit"). |
| Linear hypothesis | Conducts a comparison between your model and a linearly restricted model which you can specify. |
| **Numerical diagnostics** | |
| Variance inflation factors | The VIF tests for multicollinearity; see online document "Multiple Regression.Examining Regression Assumptions." |
| Breusch–Pagan test for heteroscedasticity | Tests for heteroscedasticity. |
| Durbin–Watson test for autocorrelation | (Seems to be used mainly in econometrics.) |
| RESET test for non-linearity | (Diagnostic for correctness of functional form.) |
| Bonferroni outlier test | Tries to identify outliers. |
| **Graphs** | |
| Basic diagnostic plots | Returns the four graphs shown in Figure 9 in the online document "Multiple Regression.Examining Regression Assumptions." |
| Residual quantile-comparison plots | Returns Q-Q plot with 95% CIs drawn. |
| Component + residual plots | These plot partial residuals against predictors; they are used to detect non-linearity if data do not follow the regression line. |
| Added-variable plots | Show leverage and influence of observations on subsets of the data; look for influential points far from the regression line. |
| Influence plot | Plots the studentized residuals against the hat-values using a bubble plot; area of circle represents observations proportional to Cook's distances. |
| Effect plots | Returns a plot that is useful to help in interpreting interactions; see Fox (2003). |

If after reading further along in the text you would like to make a more complicated model and still have all of these choices in R Commander available to you, use the STATISTICS > FIT MODELS > LINEAR MODEL dialogue box, which gives you the option of including interaction terms, nested terms, quadratic terms, etc.

### *7.3.1 Reporting the Results of a Regression Analysis*

One of the first things to report to your readers about a regression analysis is correlations between the explanatory variables and the response variable as well as correlations among the explanatory variables. If at all possible, a correlation matrix with *r*-values, *p*-values, and N should be provided to readers, such as that found in Table 7.4 of the SPSS text (*A Guide to Doing Statistics in Second Language Research Using SPSS*, p. 190).

Next, you will want to tell your readers what kind of regression you performed, whether standard, sequential, or stepwise (not recommended). If you performed a sequential regression, you will want to detail the results of the $R^2$ and $R^2$ change for each step of the model. For all regression models you should report regression coefficients, especially the unstandardized coefficients (labeled B) which are necessary to write a predictive equation, including the coefficient for the intercept (labeled as "Intercept" in the R output). If you can include 95% CIs for these, that could be helpful for future researchers who replicate your study. If you performed a sequential regression I don't think it is necessary to report on the t-tests for the contribution of each variable to the model (this is apparent from the significance of the $R^2$ change for each model), but you should probably include this information when performing a standard regression.

Probably the most important thing to report is the multiple correlation coefficient, $R^2$, and the squared semipartial correlations for each term of the model ($sr^2$). This $R^2$ value expresses how much of the variance in scores of the response variable can be explained by the variance in the statistical explanatory variables, while the $sr^2$ provides a way of assessing the unique contribution of each variable to the overall $R^2$. These numbers are already an effect size, so there is no need to require any additional reports of effect sizes.

### *7.3.2 Reporting the Results of a Standard Regression*

The following gives a sample of how the results of a standard regression with the Lafrance and Gottardo (2005) data could be reported, based on the analysis in the previous text (notice that this is not the same as what they actually reported, which was a sequential regression).

Lafrance and Gottardo (2005) examined the question of how predictive phonological awareness (measured in L2) was for Grade 1 L1 reading performance when the variables of Kindergarten L2 reading performance, nonverbal reasoning, L2 measures of naming speed, and working memory were also included in the regression equation. There were correlations between the Grade 1 L1 reading measures and the five explanatory variables in the model, with the highest being L2 phonological awareness ($r$=.73). There were also intercorrelations among the five explanatory variables, with the highest being between phonological awareness and Kindergarten L2 reading measures ($r$=−.71) and phonological awareness and naming speed ($r$=−.70).

A standard regression resulted in the unstandardized regression coefficients shown below.

| | *Total* $R^2$ | *Nonverbal Reasoning* | *Kindergarten Reading* | *Naming Speed* | *Working Memory* | *Phonological Awareness* |
|---|---|---|---|---|---|---|
| | | *B* | *B* | *B* | *B* | *B* |
| | .55* | −.003 | .07 | −.31 | .03 | 1.32 |
| Relative importance | | .03% | 11% | 10% | 8% | 27% |

This model with all five predictors accounted for 58% of the variance in first-grade L1 reading scores, but the only statistical predictor was phonological awareness (PA). Certainly, PA is the most important variable for predicting reading performance.

### *7.3.3 Reporting the Results of a Sequential Regression*

If you perform a sequential regression, you will need to report some different types of data. Here is an example of what I would report with the Lafrance and Gottardo data. You could work summaries of the original model and the other models (1–4) given in this section to understand how I arrived at the data reported here (and this is the way that Lafrance and Gottardo reported their data).

Lafrance and Gottardo (2005) examined the question of whether phonological awareness (measured in L2) would add in predictive power for Grade 1 L1 reading performance after the variables of Kindergarten L2 reading performance, nonverbal reasoning, L2 measures of naming speed, and working memory were already added to the regression equation. There were correlations between the Grade 1 L1 reading measures and the five explanatory variables in the model (see Table 7.4 in the SPSS book, *A Guide to Doing Statistics in Second Language Research Using SPSS*, p. 190), with the highest being L2 phonological awareness ($r$=.73). There were also intercorrelations among the five explanatory variables, with the highest being between phonological awareness and Kindergarten L2 reading measures ($r$=.71) and phonological awareness and naming speed ($r$=−.67).

It was assumed from previous research that phonological awareness (PA) would be the most significant factor in a standard regression, so a sequential regression was run to examine the effects of the other variables before PA was added. The variable of Kindergarten L2 reading measures added the most explanatory power to the model ($R^2$=.20) when it was added before PA, as can be seen in the $\Delta R^2$ column. After all other variables were added, PA accounted for 14% of the variance, but it was the only statistical variable in the regression model with all five explanatory variables. Total $R^2$ for the model, change in $R^2$, and unstandardized regression coefficients (B) are found in the table below.

| *Model* | *Total $R^2$* | *$\Delta R^2$* | *Nonverbal Reasoning B* | *Kindergarten Reading B* | *Naming Speed B* | *Working Memory B* | *Phonological Awareness B* |
|---|---|---|---|---|---|---|---|
| 1 | .12 | .12 | .02* | | | | |
| 2 | .32 | .20 | .008 | .38* | | | |
| 3 | .39 | .8 | .003 | .31* | −1.08* | | |
| 4 | .40 | .01 | .003 | .26* | −.80 | .06 | |
| 5 | .54 | .14 | −.002 | .03 | .10 | .04 | 1.45* |

*means $p$<.05 in the t-test for this term; Intercept for Model 5=−1.02; with all five predictors, the regression equation is statistical, $F_{5,34}$=8.1, $p$<.0005.

Model 5, with all five predictors, accounted for 54% of the variance in first-grade L1 reading scores, but the only statistical predictor was PA. Certainly, PA is the most important variable for predicting reading performance.

The answer to the question of whether phonological awareness added to the prediction of first-grade L1 word reading performance after differences in other predictors were eliminated was yes. Phonological awareness added 14% to explaining the total variance of first-grade L1 reading scores.

## 7.4 Application Activities with Multiple Regression

1. Lafrance and Gottardo (2005) data. Import the SPSS LafranceGottardo.sav file (and name it lafrance; it has 40 rows and 9 columns) and then replicate the results I listed in the

"Reporting the results of a sequential regression" section of the "Multiple Regression.Doing the same type of regression as SPSS" file. Use the Grade 1 L1 reading scores as the response variable and the five explanatory variables shown in the text. Use the summary() function for your five models to get the R, $R^2$, and unstandardized regression coefficient information.

2. Lafrance and Gottardo (2005) data. Lafrance and Gottardo (2005) also looked at a multiple regression using Grade 1 L2 reading scores as the response variable and the same five explanatory variables (L2 measures) used in the text. Using the LafranceGottardo.sav file, perform a standard regression to determine what variables were statistical predictors of Grade 1 L2 reading scores, what their standardized regression coefficients were, and what the $R^2$ of the model is. Also report relative importance metrics. Mention whether this model satisfies the assumptions of regression.

3. French and O'Brien (2008). Use the French and O'Brien Grammar.sav file, import it into R, and call it french. The authors examined French L1 children in an intensive English program and tested their phonological memory abilities. In order to see whether phonological memory predicted later abilities in learning vocabulary and grammar in English, the authors performed several sequential multiple regressions. We will look only at one where the response variable was Time 2 grammar (GRAM_2). The explanatory variables were, in the order they were entered, Time 1 grammar (GRAM_1), scores on an intelligence test (INTELLIG), language contact (L2CONTA), then an Arabic nonword phonological memory test at Time 1 (ANWR_1), and lastly an English nonword phonological memory test at Time 2 (ENWR_1). Perform this sequential regression in R by starting with a model with only one term and then adding terms to each subsequent model. Use the summary() command to get the overall $R^2$ and the unstandardized regression coefficients for each model. Summarize the results in a table like the one in the last section, "Reporting the Results of a Sequential Regression," in the online document "Multiple Regression.Doing the same type of regression as SPSS" where the $R^2$ , change in $R^2$, and unstandardized regression coefficients are listed. Also calculate relative importance metrics for the last model.

### 7.5 Finding the Best Fit

When you begin to use R to perform regressions, the syntax for the command is so transparent that you cannot help but have some inkling of what you are doing! Once you understand what you are doing, you also come to understand that there are alternative ways of fitting a regression model to your data, and that you can easily alter the model. The quote by John Fox at the beginning of Chapter 7 of the SPSS book, *A Guide to Doing Statistics in Second Language Research Using SPSS*, summarizes this philosophy—the point is not in simply performing a regression, but in finding the regression model which best fits your data. Crawley says, "Fitting models to data is the central function of R. The process is essentially one of exploration; there are no fixed rules and no absolutes. The object is to determine a minimal adequate model from the large set of potential models that might be used to describe a given set of data" (2002, p. 103).

Let's get started then! The first step to understanding how to fit models is to understand a little bit more about the syntax of R's regression commands. You've already seen that a plus sign ("+") simply adds variables together. This syntax asks R to add in the main effects of each variable. Another possibility for fitting a model is to use interaction effects. The colon (":") indicates an interaction effect. If you have three variables in your regression, a full factorial model that includes all main effects and interaction effects is:

```
model = y~A + B + C + A:B + B:C + A:B:C
```

There is a shortcut for this syntax, however, which is the use of the star ("*"):

model = y~A*B*C = A + B + C + A:B + B:C + A:B:C

You can use the carat sign ("^") to specify to what order you want interactions. If you only want the two-way interaction but not the three-way interaction, you would write:

model = y~(A + B + C)^2 = A + B + C + A:B + A:C

You could also achieve the same results this way:

model = y~A*B*C – A:B:C

which would remove the three-way interaction. Besides interaction terms, you might want your linear regression equation to include quadratic terms, which are squared terms. To raise a number to another number in R you use the carat sign, so A squared would be "A^2." However, as we saw above, the carat is already used to represent an interaction model expansion, so we will need to use the "as is" function I (upper-case letter i) to suppress the incorrect interpretation of the carat as a formula operator. Thus, to create a model with two main effects plus the quadratic of each main effect, we would write:

model = y~A + I(A^2) + B + I(B^2)

Table 7.1 is a summary of the operators we have just looked at.

*Table 7.1* Operators in Regression Syntax

| *Operator* | *Function* | *Example* |
|---|---|---|
| + | Adds parts of the regression equation together. | y~A + B<br>y~A + B + A:B |
| : | Creates an interaction term. | y~A + B + C + A:B + A:C + A:B:C |
| * | Expands to all main effects and interactions. | y~A*B = y~ A + B + A:B |
| - | Subtracts out terms. | y~A*B-A:B = A + B |
| ^N | Expands to Nth-way interaction. | y~(A+B+C)^2<br>= A + B + C + A:B + A:C |
| I | Uses arithmetic of syntax. | y~A + I(A^2) |

I'd like to make a comment here about the type of regression that R is doing. Standard discussions of multiple regression (such as Tabachnik and Fidell, 2001) will mention three types of regression: standard, sequential, and stepwise. In standard regression only the unique portions of overlap between the explanatory and response variable are counted, while in sequential or hierarchical regression all areas of overlap are counted, so that order of entry into the regression matters. In the next section I will be discussing how to find the best model fit of a regression, and I will be introducing something Crawley (2007) calls stepwise modeling. Now ordinarily statistics books will tell you that stepwise regression is not a good idea; Tabachnik and Fidell call it a "controversial procedure" (2001, p. 133). However, the

type of stepwise modeling that Crawley and I are advocating is not one that lets a computer determine the best model. Actually, you can do that using the step command in R, but generally I recommend here conducting your own stepwise regression by hand first, and only later checking the step model. In both sequential and stepwise modeling the order of entry determines how much importance a given variable will have, if explanatory variables are correlated with each other. R is using this type of regression, since Crawley (2007, p. 328) says that "the significance you attach to a given explanatory variable will depend upon whether you delete it from a maximal model or add it to the null model. If you always test by model simplification then you won't fall into this trap."

To learn about more advanced operations, such as nesting, non-parametric models, or polynomial regression, Chapter 9 of Crawley (2007) is a good place to start.

### *7.5.1 First Steps to Finding the Minimal Adequate Model in R*

"All models are wrong" (Crawley, 2007, p. 339). What Crawley means by this is that, whatever model you use to fit your data, there will be error. Thus all models are wrong, but some models are better than others. In general, a simpler model is better than a more complicated one, if they have the same explanatory value. A model with less error will be better than one with more error.

In order to find the best model for your data, Crawley (2007) recommends beginning with the maximal model and then simplifying your model where possible. The way to simplify is to remove one term at a time, beginning with the highest-order interactions. If you have more than one higher-order interaction or main effect to choose from, start with the one that has the highest *p*-value (the least statistical one). You will then compare your simplified model with the original model, noting the residual deviance and using an ANOVA to check whether the removal of the term resulted in a statistically different model. If the ANOVA is statistical, meaning there is a statistical difference in the deviance of the two models, you will keep the term in the model and continue to test for other terms to delete. By the way, if you have reason to keep an interaction, you must keep all of the components of the interaction in your regression model as well. In other words, if your Condition:Gender interaction is statistical, you must keep the main effect for Condition and the main effect for Gender as well. If there is no difference, however, you will accept the second model as the better and continue forward with the simplified model. This process repeats until the model contains the least amount of terms possible.

This process probably sounds confusing, so we will walk through an example of how it works. This whole process can get very complicated with a large number of terms (some statisticians say that you shouldn't test more terms than you could interpret, and draw the line at three explanatory variables), so I will explore the Lafrance and Gottardo (2005) data using only the three variables which were important in the relative importance metrics: phonological awareness in the L2 (PAL2), Kindergarten L2 reading performance (KL2WR), and naming speed, measured in the L2 (NS). If you are following along with me, first import the SPSS file Lafrance5.sav and call it lafrance5. Next, we will impute missing values.

```
library(mice)
imp<-mice(lafrance5)
complete(imp)
lafrance<-complete(imp)
```

Because I am going to be writing the variables again and again, I am going to change their names to those in parentheses in the previous sentence.

```
lafrance$PAL2<-lafrance$PhonologicalAwarenessInL2
lafrance$KL2WR<-lafrance$KinderL2ReadingPerformance
lafrance$NS<-lafrance$NamingSpeed
lafrance$G1L1WR<-lafrance$Grade1L1ReadingPerformance
```

The research question is how well these three variables explain what is going on in the response variable, which is Grade 1 L1 reading performance (G1L1WR).

The first step is to create the maximal model. Crawley (2007, p. 326) says a maximal model is one that "[c]ontains all (p) factors, interactions and covariates that might be of any interest." This contrasts with the saturated model, which contains one parameter for every data point. The Lafrance and Gottardo data set contains 37 rows (naming speed has three missing values, so, while most variables have 40 rows, NS has only 37), so a saturated model would contain 37 parameters.

We will start with a full factorial model that includes the main effects of the three explanatory variables as well as the three two-way interactions and one three-way interaction. This model thus has seven parameters. Be aware that we could indeed start with an even more complicated maximal model that would include quadratic terms if we thought those were necessary, but for this demonstration we won't get that complicated.

```
model1=lm(G1L1WR~PAL2*KL2WR*NS, na.action=na.exclude, data=lafrance)
```

| | |
|---|---|
| model1=lm ( . . .) | This puts the regression model into an object called 'model' (you can name it anything you like). |
| lm (formula, data, . . .) | 'lm' fits linear models. |
| G1L1WR ~ . . . | The tilde ("~") means the thing before it is modeled as a function of the things after it. |
| PAL2*KL2WR*NS | The explanatory variables; this formula expands to: PAL2 + KL2WR + NS + PAL2:KL2WR + KL2WR:NS + PAL2:NS + PAL2:KL2WR:NS |
| na.action=na.exclude | Excludes missing (NA) values; it is not needed strictly to fit the regression model, but it is needed for the diagnostics, specifically involving the residuals, to come out correctly, so we will enter it here. |
| data=lafrance | If you have not attached the data frame, specify it here. |

Tip: If all of the variables in the data frame are to be used, there is an alternative notation that is much shorter than typing in all of the variable names. The "." to the right of the tilde means to include all of the remaining variables:

```
model=lm(G1L1WR ~ ., data=lafrance, na.action=na.exclude)
```

Having now fit a linear model to our equation, we can do all kinds of things with our fitted object (model1). First we want to look at a summary that will give us the unstandardized regression coefficients, their standard error, and the statistical significance of the regression coefficients by way of a t-test and associated probability.

summary(model1)

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   -2.1791     8.9842  -0.243    0.810
PAL2           2.3537     6.2058   0.379    0.707
KL2WR         35.3024    43.5385   0.811    0.423
NS             0.5130     3.8104   0.135    0.894
PAL2:KL2WR   -21.2785    26.5786  -0.801    0.429
PAL2:NS       -0.3374     2.6867  -0.126    0.901
KL2WR:NS     -15.7141    20.1119  -0.781    0.440
PAL2:KL2WR:NS  9.4874    12.2796   0.773    0.445

Residual standard error: 0.33 on 32 degrees of freedom
Multiple R-squared: 0.5575,     Adjusted R-squared: 0.4607
F-statistic: 5.759 on 7 and 32 DF,  p-value: 0.0002249
```

We should look at the residual deviance ("Residual standard error") of the model. It is .33 on 32 degrees of freedom. Models with a better fit will reduce the residual deviance. We don't have anything to compare the .32 deviance to yet, but we will later. We also look at the t-tests for each term. At the moment, none are below *p*=.05, so none are statistical (remember that these *p*-values are unadjusted).

The unstandardized regression coefficients (under "Estimate" in the read-out) are the population regression parameters (the $\alpha$ and $\beta$ of our regression equation), and estimating these numbers is "the central object of simple-regression analysis," according to Fox (1997, p. 112). They provide effect size information. The regression coefficients provide information about the strength of the association between the explanatory variable and the response variable. For example, the estimate of 2.35 for PAL2 means that a 1% increase in PAL2 (phonological awareness) is associated with a 2.35% increase in first-grade L1 word reading scores when all the other explanatory variables are held constant (Fox, 2002b, p. 27). The standard error of this estimate (which is something like the standard error of a mean, but calculated slightly differently; see Crawley, 2007, p. 398 for the mathematical formula for calculating standard errors of slopes and intercepts) measures the amount of unreliability surrounding the coefficient. A smaller standard error is therefore always preferable.

The *t*-value in the third column for the coefficients area tests the hypothesis that the slope is equal to zero ($H_0$: $\beta_k$=0), and the *p*-value in the fourth column tests the probability that we would find a *t*-value this large or larger if this hypothesis were true.

The next-to-last line at the bottom of the output ("Multiple R-Squared") shows how much of the variance in scores this model explains (56%). Because $R^2$ varies from 0 to 1 and can never be negative, it is always positively biased. For this reason, an adjusted $R^2$ is also provided that corrects for the positive bias, and this number says that the model accounts for 46% of the variance in scores. Whichever number you use, both are quite large as effect sizes go and mean the model explains a large amount of variance.

The last line at the bottom of the output ("F-statistic") shows the results of an ANOVA test of the hypothesis that the regression slopes are all equal to 0. In other words, it tests the hypothesis that the unstandardized regression coefficients (except for the intercept coefficient) are all equal to each other and are all zero (Fox, 1997, p. 122). Logically, there is little chance that this null hypothesis will be true, so it will almost always be statistical, as it is here, and should not be given too much attention.

Since no term is statistical in this model because none of the *p*-values in the Coefficients area are under *p*=.05, we'll take out the highest-order interaction, the three-way interaction, first. The easy way to do this is to use the update command. When you use this command, you'll need to be careful with the syntax. After the name of the original model, you'll need to use the sequence "comma tilde dot minus."

```
model2=update(model1,~.- PAL2:KL2WR:NS, data=lafrance)
summary(model2)
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.3247     8.8613  -0.149    0.882
PAL2          1.8033     6.1270   0.294    0.770
KL2WR         1.7513     3.1169   0.562    0.578
NS            0.1673     3.7608   0.044    0.965
PAL2:KL2WR   -0.7528     0.7942  -0.948    0.350
PAL2:NS      -0.1183     2.6553  -0.045    0.965
KL2WR:NS     -0.2043     1.2214  -0.167    0.868

Residual standard error: 0.328 on 33 degrees of freedom
Multiple R-squared: 0.5492,     Adjusted R-squared: 0.4673
F-statistic: 6.702 on 6 and 33 DF,  p-value: 0.0001046
```

We notice that the residual error is slightly smaller in this model and that we have one more degree of freedom. The unstandardized regression coefficients for several of the terms have become much smaller; for example, the unstandardized regression coefficient for KL2WR has shrunk from 35.3 to 1.8, and its standard error from 43.5 to 3.1.

Now we compare the two models using ANOVA:

```
anova(model1,model2)
```

```
Analysis of Variance Table

Model 1: G1L1WR ~ PAL2 * KL2WR * NS
Model 2: G1L1WR ~ PAL2 + KL2WR + NS + PAL2:KL2WR + PAL2:NS + KL2WR:NS
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     32 3.4849
2     33 3.5499 -1 -0.065007 0.5969 0.4454
```

The ANOVA has a *p*-value=.45, indicating that there is a non-statistical difference in deviance between the two models, so we retain the simpler model2. In general, we prefer the simpler model to the more complex one, if they do not differ in explanatory value. When looking at different models, you should keep in mind that a saturated model (which has as many parameters as data points) will have a perfect fit, meaning that the $R^2$ value would be 1.0. However, this model would have no explanatory value. So we are always doing a balancing act between trying to reduce the number of parameters (and get more degrees of freedom) and trying to improve the goodness of fit (a higher $R^2$ value). A value called Akaike's information criterion (AIC) calculates a number that "explicitly penalizes any superfluous parameters in the model" (Crawley, 2007, p. 353) while rewarding a model for better fit. In other words, the AIC tries to give a number to that balancing act between reducing the number of parameters (or conversely increasing the degrees of freedom) and

increasing the fit. Thus, the smaller the AIC, the better the model. We can use the AIC function to evaluate models as well:

```
AIC(model1,model2)
```

```
        df      AIC
model1  9 33.89691
model2  8 32.63620
```

Because model2 has a lower AIC, it is the preferable model. The automatic stepwise deletion function, boot.stepAIC, which we will examine soon, will use the AIC to evaluate models.

We will now continue our deletion procedure by choosing the next term to delete. Since the three-way interaction is gone, we will next select a two-way interaction. There are three of them, so we'll pick the one with the highest *p*-value from the summary for model2. That was the PAL2:NS interaction.

```
model3=update(model2,~.- PAL2:NS, data=lafrance)
summary(model3)
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.945788   2.452350  -0.386   0.7021
PAL2         1.531547   0.570108   2.686   0.0111 *
KL2WR        1.792345   2.933491   0.611   0.5453
NS           0.003577   0.783601   0.005   0.9964
PAL2:KL2WR  -0.741526   0.741935  -0.999   0.3246
KL2WR:NS    -0.231402   1.043650  -0.222   0.8259
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3231 on 34 degrees of freedom
Multiple R-squared: 0.5492,     Adjusted R-squared: 0.4829
F-statistic: 8.285 on 5 and 34 DF,  p-value: 3.346e-05
```

We finally see a term in model3 which is statistical, which is the main effect of phonological awareness. Let's compare model2 and model3:

```
anova(model2,model3)
```

```
  Res.Df    RSS Df   Sum of Sq     F Pr(>F)
1     33 3.5499
2     34 3.5501 -1 -0.00021352 0.002 0.9647
```

There is no difference in deviance between the two models (the *p*-value is higher than .05), so we will accept model3 and delete the next two-way interaction with the highest *p*-value, which is KL2WR:NS.

```
model4=update(model3,~.- KL2WR:NS, data=lafrance)
summary(model4)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.5836     1.8042  -0.323  0.74826
PAL2          1.4627     0.4716   3.102  0.00379 **
KL2WR         1.1952     1.1469   1.042  0.30452
NS           -0.1137     0.5703  -0.199  0.84316
PAL2:KL2WR   -0.6816     0.6815  -1.000  0.32411
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3187 on 35 degrees of freedom
Multiple R-squared: 0.5486,     Adjusted R-squared: 0.497
F-statistic: 10.63 on 4 and 35 DF,  p-value: 9.566e-06
```

Notice that the residual standard error does continue to decrease. Compare model3 and model4:

```
anova(model3,model4)
```

```
  Res.Df    RSS Df  Sum of Sq      F Pr(>F)
1     34 3.5501
2     35 3.5552 -1 -0.0051331 0.0492 0.8259
```

No difference, so we'll take out the last two-way interaction term.

```
model5=update(model4,~.- PAL2:KL2WR, data=lafrance)
summary(model5)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.28013    1.77854  -0.158  0.87573
PAL2         1.37982    0.46425   2.972  0.00525 **
KL2WR        0.05526    0.12790   0.432  0.66831
NS          -0.18982    0.56521  -0.336  0.73895
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3187 on 36 degrees of freedom
Multiple R-squared: 0.5357,     Adjusted R-squared: 0.497
F-statistic: 13.84 on 3 and 36 DF,  p-value: 3.681e-06
```

```
anova(model4,model5)
```

```
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     35 3.5552
2     36 3.6568 -1  -0.10161 1.0003 0.3241
```

Again, there's no difference between the two models, so we'll continue now to delete main effects. From the summary for model5, the main effect with the highest *p*-value is NS, with *p*=.74. Deleting this term:

```
model6=update(model5,~.-NS, data=lafrance)
summary(model6)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.85546    0.47199  -1.812 0.078038 .
PAL2         1.48240    0.34539   4.292 0.000122 ***
KL2WR        0.04795    0.12452   0.385 0.702384
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3149 on 37 degrees of freedom
Multiple R-squared: 0.5342,     Adjusted R-squared: 0.509
F-statistic: 21.22 on 2 and 37 DF,  p-value: 7.27e-07
```

There is no difference between model5 and model6:

```
anova(model5, model6)
```

```
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     36 3.6568
2     37 3.6683 -1 -0.011456 0.1128  0.739
```

So we remove the last main effect that is not statistical, KL2WR.

```
model7=update(model6,~.-KL2WR, data=lafrance)
summary(model7)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.9713     0.3596  -2.701   0.0103 *
PAL2          1.5771     0.2398   6.577 9.24e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3113 on 38 degrees of freedom
Multiple R-squared: 0.5323,     Adjusted R-squared:  0.52
F-statistic: 43.26 on 1 and 38 DF,  p-value: 9.24e-08
```

The minimal adequate model is one with only phonological acquisition in it. This model explains $R^2$=.53 of the variance in scores, with .31 residual standard error on 38 degrees of freedom. Remember that model1 had a residual standard error of .32 on 29 df, so we have improved the fit (by reducing the amount of error) and decreased the number of parameters (thereby increasing the df). We can check to make sure the minimal adequate model is not just the null model (with only one parameter, the overall mean) by comparing the fit of model7 with the null model in model8:

```
model8=lm(G1L1WR~1,data=lafrance,na.action=na.exclude)
anova(model7,model8)
```

```
Analysis of Variance Table

Model 1: G1L1WR ~ PAL2
Model 2: G1L1WR ~ 1
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1     38  3.6830
2     39  7.8753 -1   -4.1924 43.256 9.24e-08 ***
```

Thankfully, our one-parameter model is indeed statistically different from the null model. Note that the probability of the ANOVA test is recorded in statistical notation, with 9.24e-08 being shorthand for *p*=.0000000924. We continue by assuming that model7 is our minimal adequate model. We are not done yet, however, because we need to check regression assumptions for this model, which you may see in the online document "Multiple Regression.Examining Regression Assumptions."

This section has shown how to conduct a backwards stepwise analysis by hand until a model is reached where all terms are statistical. Another way to go about a stepwise analysis is to use the boot.stepAIC function from the bootStepAIC library. This function uses a bootstrap procedure to help evaluate different models, and is much more accurate and parsimonious than the step procedure found in the base library of R.

Let's see what results we get if we use boot.stepAIC with our original model:

```
library(bootStepAIC)
boot.stepAIC(model1,data=lafrance)
```

The beginning of the output gives some numbers which summarize how many times each variable was selected as an independent predictor in each bootstrap sample (100 is the default). Austin and Tu (2004) found 60% was an optimal cut-off level for including the predictors in the final equation. However, because this process is transparent (you can see the percentages in the column titled "Covariates selected"), this information gives the researcher tools to determine on their own the strength of the predictors.

```
Summary of Bootstrapping the 'stepAIC()' procedure for

Call:
lm(formula = G1L1WR ~ PAL2 * KL2WR * NS, data = lafrance, na.action = na.exclude)

Bootstrap samples: 100
Direction: backward
Penalty: 2 * df

Covariates selected
              (%)
PAL2           97
KL2WR          67
NS             54
PAL2:KL2WR     60
PAL2:NS        47
KL2WR:NS       33
PAL2:KL2WR:NS  31
```

The next part of the output concerns the stability of the predictors. The bootstrap procedure samples randomly with replacement, and each sample will contain regression coefficients. Austin and Tu (2004) note that we would ideally want all of the coefficients to be either positive or negative, and that if half the coefficients were positive and half were negative this would be a sign of instability in the model. The boot.stepAIC procedure, unlike the step procedure, is able to assess this measure of stability and provide it to the researcher. The output shows that NS and PAL2:NS were quite unstable.

```
Coefficients Sign
               + (%) - (%)
KL2WR          95.52  4.48
PAL2           78.35 21.65
NS             40.74 59.26
PAL2:KL2WR:NS 93.55  6.45
PAL2:NS        57.45 42.55
KL2WR:NS        9.09 90.91
PAL2:KL2WR      3.33 96.67
```

The next part of the output, not shown here, shows what percentage of the time the term was a statistical predictor at the alpha=.05 level in the model. The next part lists the final minimal adequate model that boot.stepAIC finds.

```
Initial Model:
G1L1WR ~ PAL2 * KL2WR * NS

Final Model:
G1L1WR ~ PAL2
```

Last of all is the analysis of deviance table for various permutations of the model (7 in all here).

```
              Step Df     Deviance Resid. Df Resid. Dev       AIC
1                                         32   3.484850 -81.61817
2 - PAL2:KL2WR:NS  1 0.0650070569         33   3.549857 -82.87888
3       - PAL2:NS  1 0.0002135147         34   3.550071 -84.87647
4     - KL2WR:NS  1 0.0051331439         35   3.555204 -86.81868
5           - NS  1 0.0040360531         36   3.559240 -88.77330
6    - PAL2:KL2WR  1 0.1090265486         37   3.668267 -89.56641
7         - KL2WR  1 0.0147013353         38   3.682968 -91.40642
```

The final model that boot.stepAIC returns is exactly the same one we reached through our manual deletion. The analysis of deviance table returns the change in deviance each time and the AIC value (which, since it is negative, is smaller the larger the number is!). Because boot.stepAIC uses bootstrap resampling methods, it is an excellent performer that very rarely retains spurious terms in the final model.

I will note that the boot.stepAIC will not work with data sets that have missing values. You can try to clear out the missing values before running it (lafrance=na.omit(lafrance)) or impute the data as I did before I started. If we have data that includes missing values, the data is **non-orthogonal**. If your data set is totally complete and there is no missing data, it is said to be **orthogonal**. Crawley (2007) notes that, when data sets are non-orthogonal, as this one was before we imputed values, and explanatory variables correlate with each other, as they do here, then the importance you attach to a variable will depend on whether you subtract it from the maximal model or add it to the null model. Crawley recommends always subtracting from the maximal model to avoid problems with this, and that is what we did manually, and that is also what the bootstep model does; it just does it many times and then evaluates what percentage of the time terms were included.

Although boot.stepAIC is extraordinarily accurate at finding the model with the lowest AIC, this doesn't mean you can just use it and forget the method of hand deletion that I have shown in this section, because this method generalizes to mixed-effects models (Chapter 12)

that boot.stepAIC can't model. Another case you may be forced to use hand deletion in is one where you have more parameters than cases, which is called overparameterization. For an example of what to do in this case, see the online document "Multiple Regression.Further steps in finding the best fit."

Once you have determined what your minimal adequate model is, you will be interested in finding out the relative importance of the terms in your model. As mentioned previously, this is best done using the calc.relimp function in the relaimpo library. Since our minimal adequate model involves only one term, we obviously cannot determine the relative importance of terms, but this process was illustrated above. Just don't forget to do it if you have more than one term left in your minimal adequate model!

*Finding a Minimal Adequate Regression Model*

Here are the steps that Crawley (2007, p. 327) suggests for the model simplification process:

1. Fit the maximal model (include factors, interactions, possibly quadratic terms).
2. Inspect model summaries and take out highest-order non-statistical terms first, one at a time, using update(model1,~.-A:B:C).
3. Using ANOVA, compare model1 to model2. If there is no statistical difference between the models, retain the newer model, inspect the summary of the newer model, and continue to delete least statistical highest-order terms.
4. If the ANOVA shows a statistical difference, keep the older model.
5. I recommend checking your model with boot.stepAIC (bootStepAIC library).
6. Last of all, run calc.relimp (relaimpo library) to determine the relative importance of the terms in your regression equation.

Notes:

- If an interaction is kept in a model, its component main effects must be kept too (in other words, if you have NS:PAL2, you must keep both NS and PAL2 in the model as well).
- Do not fit more parameters than you have data points.
- If deletion results in no statistical parameters, the null model (y~1) is the minimal adequate one (back to the drawing board as far as experimental design is concerned!).
- *If your design is non-orthogonal (meaning there are some missing values in some variables), the order in which you conduct the regression will make a difference in how important the variable seems to be.

### *7.5.2 Reporting the Results of Regression in R*

Crawley (2007) recommends that you report whether your data is **orthogonal** or not (meaning whether data is missing), report on correlations between your explanatory variables, and then present your minimal adequate model. You should also let your reader know what steps you took in your search by giving a list of the non-statistical terms that were deleted, and the change in deviance. If you report these things, Crawley (2007, p. 329) says, "Readers can then judge for themselves the relative magnitude of the non-significant factors, and the importance of correlations between the explanatory variables." In the R summary the residual standard error is not the same as the deviance, but the boot.stepAIC summary will give a nice deviance table. You can also calculate the deviance for each model (and then calculate the change in deviance by subtracting the deviance of model 2 from model 1) this way:

```
deviance(model1)
[1] 3.484850
```

Here, then, is a summary of my search for a minimal adequate model with three factors using the Lafrance and Gottardo (2005) data used in this section.

Using data from Lafrance and Gottardo (2005) I modeled a regression analysis with scores on Grade 1 L2 reading performance with the three variables of phonological awareness in L2 (PAL2), Kindergarten scores on the L2 reading performance (KL2WR), and naming speed (NS). There were high intercorrelations among all three explanatory variables (PAL2-KL2WR, $r$=.7; PAL2-NS, $r$=−.7; KL2WR-NS, $r$=−.4). There were missing data points in the naming speed data, so the data was non-orthogonal, but I imputed the data first using R's mice library. To search for a minimal adequate model, I started with a full factorial model of all three main effects plus all two-way interactions and the three-way interaction between terms. Deleting the terms and then checking for differences between models, my minimal model was one with only the phonological awareness term. This model explained $R^2$=.53% of the variance in scores on the Grade 1 reading test. For PAL2, the estimate for the unstandardized coefficient was 1.58, meaning that, for every 1% increase in phonological awareness, there was a 1.58% increase in scores. This term was statistical, $t$=6.6, $p$<.0001. The table below gives the steps of my model search and the change in deviance:

| *Model* | *Terms* | *Deviance* | *ΔDeviance* |
|---|---|---|---|
| Model1 | PAL2*KL2WR*NS | 3.484850 | |
| Model2 | -PAL2:KL2WR:NS | 3.549857 | .0650 |
| Model3 | -PAL2:NS | 3.550071 | .0002 |
| Model4 | -KL2WR:NS | 3.555204 | .0051 |
| Model5 | -PAL2:KL2WR | 3.656811 | .1016 |
| Model6 | -NS | 3.668267 | .0115 |
| Model7 | -KL2WR | 3.682968 | .0147 |

In checking model assumptions, this model showed heteroscedasticity and non-normal distribution of errors.

### 7.6 Further Steps to Finding the Best Fit: Overparameterization and Polynomial Regression

In the "Multiple regression:Finding the best fit" online document we explored a model of the data for Lafrance and Gottardo (2005) that included only three variables. But let's say we actually want to fit all five variables that the authors fit. If we fit a full factorial model with all main effects and interactions, we will have 1 five-way interaction, 5 four-way interactions, 10 three-way interactions, 10 two-way interactions, and 5 one-way interactions, for a total of 31 interactions. Crawley (2007) says a useful rule of thumb is not to estimate more than n/3 parameters during a multiple regression. Since the Lafrance and Gottardo (2005) data set has $n$=37 data points, that means we shouldn't estimate more than about 12 parameters at any one time. In the case of only three terms, there were seven parameters and that was fine. However, for the case of five terms, we will want to conduct the regression in separate runs.

One rule for doing this is that, if you have an interaction, you'll need to have the component main effects in the regression at the same time (see Crawley, 2007, p. 446).

Actually, just to make things a little more fun, let's start by including quadratic terms along with the main effects, just to check whether there is any curvature in our data. In the scatterplots for this data examined previously, there did seem to be some possibility of curvature in relationships between explanatory variables and the response variable. A regression involving quadratic (or higher) terms is called a polynomial regression by Crawley (2007). Although the model is still considered linear, having quadratic terms would mean we are fitting a line with some curvature.

If you're following along with me, use the imputed data set called lafrance that we created in the online document "Multiple regression:Finding the best fit." We'll need to change a couple more of the names to make it easier to see what we're doing.

```
lafrance$NR<-lafrance$NonverbalReasoning
lafrance$WM<-lafrance$WorkingMemory
```

Here is the first model we will try which includes all five main effects and their associated quadratic terms:

```
model1=lm(G1L1WR~NR+I(NR^2)+WM+I(WM^2)+NS+I(NS^2)+PAL2+
I(PAL2^2)+KL2WR+I(KL2WR^2),data=lafrance)
```

Remember that the (NR^2) term means it is an $X^2$ term. The I (capital letter "i") in front of the parentheses means that the carat (^) is performing its arithmetic job of squaring, instead of its regression function of expanding a regression term to the following number of interactions. Let's look at a summary of this model:

```
summary(model1)
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.660e+01  1.445e+01   1.148   0.2602
NR           8.557e-02  7.906e-02   1.082   0.2880
I(NR^2)     -8.527e-04  8.003e-04  -1.066   0.2954
WM           1.612e-01  1.595e-01   1.011   0.3205
I(WM^2)     -1.831e-02  3.146e-02  -0.582   0.5650
NS          -2.167e+01  1.359e+01  -1.594   0.1217
I(NS^2)      4.842e+00  3.014e+00   1.607   0.1190
PAL2         7.982e+00  4.664e+00   1.711   0.0977 .
I(PAL2^2)   -2.382e+00  1.651e+00  -1.442   0.1599
KL2WR        1.280e-01  3.462e-01   0.370   0.7142
I(KL2WR^2)   7.666e-03  2.513e-01   0.031   0.9759
```

Nothing is statistical. Instead of a manual stepwise deletion, however, I am going to use the automatic boot.stepAIC procedure. One thing I like about this (over manual deletion in this situation of overparameterization) is that it will be more stable because it can simulate taking 100 samples of the variables.

```
library(bootStepAIC)
boot.stepAIC(model1, data=lafrance)
```

The automatic procedure tells me that my best model is the following:

```
model2=lm(G1L1WR~ NS + I(NS^2) + PAL2 + I(PAL2^2), data=lafrance)
summary(model2)
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    16.822     13.557   1.241   0.2229
NS            -19.454     12.806  -1.519   0.1377
I(NS^2)         4.298      2.837   1.515   0.1388
PAL2            7.440      3.632   2.048   0.0481 *
I(PAL2^2)      -2.029      1.249  -1.625   0.1132
```

Not all of the terms are statistical, and it seems boot.stepAIC may have been generous at leaving terms in, but for now we'll keep these until we do a final culling.

Our next step will be to test the interaction terms. Following Crawley (2007), we will enter the names of the 10 two-way interactions, and then randomize them. Note that, to get the names of all of the two-way interactions, I just fit the full factorial model, asked for the summary, and looked at the summary not for any statistical information, but just for the full specification of all of the interactions!

```
getfullmodel=lm(G1L1WR~NR*WM*NS*PAL2*KL2WR,data=lafrance)
summary(getfullmodel)
#Output not printed but gives me what I need for following command
interactions=c("NR:WM", "NR:NS", "WM:NS", "NR:PAL2", "WM:PAL2", "NS:PAL2",
"NR:KL2WR", "WM:KL2WR", "NS:KL2WR", "PAL2:KL2WR")
```

We'll conduct two separate tests with about half of the two-way interaction terms per test and all of the five main effects as well.

```
model3= lm(G1L1WR~NR+WM+ NS+ PAL2+ KL2WR+
PAL2:KL2WR + NR:NS + NS:PAL2 + NR:WM+NS:KL2WR, data=lafrance)
model4= lm(G1L1WR~NR+WM+ NS+ PAL2+ KL2WR+
WM:KL2WR + NR:KL2WR+WM:PAL2+WM:NS+NR:PAL2, data=lafrance)
```

Here are coefficient terms for these two models:

Model3:

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.493901  12.057454   0.787   0.4374
NR          -0.304256   0.222001  -1.371   0.1810
WM          -0.729379   0.479151  -1.522   0.1388
NS          -3.768012   5.025259  -0.750   0.4594
PAL2         3.843637   6.350892   0.605   0.5497
KL2WR        3.654772   3.329725   1.098   0.2814
PAL2:KL2WR  -1.933163   1.024904  -1.886   0.0693 .
NR:NS        0.113845   0.091819   1.240   0.2250
NS:PAL2     -0.965094   2.738708  -0.352   0.7271
NR:WM        0.016525   0.009983   1.655   0.1086
NS:KL2WR    -0.212036   1.307879  -0.162   0.8723
```

Model4:

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.056450   5.668426  -1.068   0.2941
NR           0.096539   0.087349   1.105   0.2782
WM           0.207495   1.532853   0.135   0.8933
NS           0.132473   1.316886   0.101   0.9206
PAL2         5.286975   3.041512   1.738   0.0928 .
KL2WR       -1.684883   1.198879  -1.405   0.1705
WM:KL2WR    -0.096495   0.127417  -0.757   0.4550
NR:KL2WR     0.041144   0.026208   1.570   0.1273
WM:PAL2     -0.091446   0.401312  -0.228   0.8213
WM:NS        0.003571   0.505618   0.007   0.9944
NR:PAL2     -0.077842   0.064039  -1.216   0.2340
```

Performing a boot.stepAIC (boot.stepAIC(model3,data=lafrance)) on both model3 and model4, it recommends leaving in the following two-way interactions only:

PAL2:KL2WR
NR:WM
WM:KL2W
NR:KL2WR
NR:PAL2

Crawley (2007) recommends putting all the interaction terms that are statistical (or nearly so) into one model with the five main effects and seeing which ones remain statistical. Since we only found 5 two-way interaction terms to keep, this will not overload the number of parameters in one run:

model5= lm(G1L1WR~NR+WM+ NS+ PAL2+ KL2WR+
PAL2:KL2WR+ NR:WM +WM:KL2WR +NR:KL2WR +NR:PAL2, data=lafrance)

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.63565    4.41919  -1.502   0.1440
NR           0.13152    0.08393   1.567   0.1280
WM          -1.44997    0.60857  -2.383   0.0240 *
NS          -0.01864    0.60228  -0.031   0.9755
PAL2         8.53645    3.15962   2.702   0.0114 *
KL2WR       -0.03748    1.84660  -0.020   0.9839
PAL2:KL2WR  -0.22792    1.02196  -0.223   0.8251
NR:WM        0.03470    0.01366   2.540   0.0167 *
WM:KL2WR    -0.31947    0.12467  -2.562   0.0158 *
NR:KL2WR     0.02994    0.02330   1.285   0.2090
NR:PAL2     -0.16137    0.06889  -2.342   0.0262 *
```

Not all of the two-way interaction terms included are statistical, so we'll run a boot.stepAIC analysis to see what we should keep for this model.

boot.stepAIC(model5, data=lafrance)

The bootstrapped algorithm suggests taking out the main effect of NS and the first two interaction terms (PAL2:KL2WR and NR:WM). Updating the model:

model6=update(model5,~.-NS-PAL2:KL2WR-NR:WM, data=lafrance)

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.44113    3.79132  -1.435   0.1609
NR           0.09704    0.08000   1.213   0.2340
WM           0.09042    0.06073   1.489   0.1463
PAL2         5.09077    2.79666   1.820   0.0781 .
KL2WR       -1.64649    1.12292  -1.466   0.1523
WM:KL2WR    -0.12153    0.07403  -1.642   0.1105
NR:KL2WR     0.04187    0.02386   1.755   0.0889 .
NR:PAL2     -0.07906    0.05832  -1.356   0.1847
```

Again, the bootstrapped procedure has left in terms which are not statistical, but we have whittled the second-order interactions down, so we will keep these in the model for now. We now need to worry about the higher-order interactions. We repeat the process of giving names to the three-way interactions and then testing them in two separate runs with all of the main effects included as well. Another way to proceed would be to include only the three-way interactions that involve the two-way interactions that survived, and create a larger model with the eight parameters in model5 plus the 8 three-way interactions that involve any of the terms, but this surpasses our limit of 12 parameters by quite a bit, so I will continue with the simplification of the three-way terms.

```
interactions=c("NR:WM:NS", "NR:WM: PAL2", "NR:NS: PAL2", "WM:NS: PAL2",
"NR:WM: KL2WR", "NR:NS: KL2WR", "WM:NS: KL2WR", "NR: PAL2: KL2WR",
"WM: PAL2: KL2WR", "NS: PAL2: KL2WR")
model7= lm(G1L1WR~NR+WM+ NS+ PAL2+ KL2WR+
NS:PAL2:KL2WR+NR:WM:PAL2+NR:NS:KL2WR+NR:WM:NS+
WM:PAL2:KL2WR, data=lafrance)
model8= lm(G1L1WR~NR+WM+ NS+ PAL2+ KL2WR+
NR:PAL2:KL2WR+WM:NS:KL2WR+NR:WM:KL2WR+NR:NS:PAL2+
WM:NS:PAL2, data=lafrance)
```

Summaries of the models show that none of the three-way interactions are statistical, but boot.stepAIC would keep two of the three-way parameters: NR:WM:NS and WM:PAL2:KL2WR (both are from model7).

Moving on to the 5 four-way interactions and the 1 five-way interaction, I'll add the five main effects and test this 11-parameter model.

```
model9= lm(G1L1WR~NR+WM+ NS+ PAL2+ KL2WR+
NR:WM:NS:PAL2 + NR:WM:NS:KL2WR + NR:WM:PAL2:KL2WR +
NR:NS:PAL2:KL2WR + WM:NS:PAL2:KL2WR + NR:WM:NS:PAL2:KL2WR,
data=lafrance)
```

According to the summary, none of the higher-way interactions are statistical. The boot.stepAIC run, however, keeps NR:WM:NS:KL2WR and WM:NS:PAL2:KL2WR. You can see that the whole process is quite complicated and lengthy, even using our automated procedure. It also depends somewhat upon the choices you make as to what to include in each regression run. At this point I will put together all of the terms that boot.stepAIC has retained at each juncture, plus all five of the main effects, producing a 14-parameter model, which is higher than the 12 we wanted but we'll go with it.

```
model10=lm(G1L1WR~ NR + WM + NS + I(NS^2) + PAL2 + I(PAL2^2) + KL2WR+
WM:KL2WR +NR:KL2WR +NR:PAL2+ #two-way interactions
NR:WM:NS +WM:PAL2:KL2WR +          #three-way interactions
NR:WM:NS:KL2WR + WM:NS:PAL2:KL2WR, #four-way interactions
data=lafrance)
```

The summary of model10 results in a number of statistical effects, but not all, and here is boot.stepAIC's final model, with nine terms:

```
model11=lm(G1L1WR~ NR + WM + NS+ I(NS^2)+PAL2 +KL2WR+
WM:KL2WR + NR:PAL2+NR:WM:NS, data=lafrance)
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  31.939055  13.425588   2.379  0.02392 *
NR            0.061234   0.060069   1.019  0.31616
WM           -1.611992   0.505896  -3.186  0.00335 **
NS          -29.075848  11.934696  -2.436  0.02099 *
I(NS^2)       5.989491   2.594727   2.308  0.02805 *
PAL2          5.896447   2.206202   2.673  0.01205 *
KL2WR         0.855697   0.297588   2.875  0.00735 **
WM:KL2WR     -0.239070   0.078823  -3.033  0.00496 **
NR:PAL2      -0.108002   0.047503  -2.274  0.03031 *
NR:WM:NS      0.016642   0.004931   3.375  0.00205 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2846 on 30 degrees of freedom
Multiple R-squared: 0.6913,     Adjusted R-squared: 0.5987
F-statistic: 7.466 on 9 and 30 DF,  p-value: 1.216e-05
```

Amazingly, this results in a pretty good model with all higher-order terms being statistical (way to go, boot.stepAIC!). My own manual stepwise deletion without boot.stepAIC resulted in the deletion of all but the PAL2 term. We can compare these two models:

```
model12=lm(G1L1WR~PAL2, data=lafrance)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.9713     0.3596  -2.701   0.0103 *
PAL2          1.5771     0.2398   6.577 9.24e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3113 on 38 degrees of freedom
Multiple R-squared: 0.5323,     Adjusted R-squared:  0.52
F-statistic: 43.26 on 1 and 38 DF,  p-value: 9.24e-08
```

The model with more terms has a higher $R^2$ (69 vs. 53) and a smaller residual error (.28 vs. .31). However, the two models do not differ statistically in deviance:

```
>anova(model10,model11)
```

```
Model 1: G1L1WR ~ NR + WM + NS + I(NS^2) + PAL2 + KL2WR + WM:KL2WR + NR:PAL2 +
    NR:WM:NS
Model 2: G1L1WR ~ PAL2
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1     30 2.4308
2     38 3.6830 -8   -1.2522 1.9318 0.09172 .
```

Clearly, which result one would pick would depend on the questions. If the question were whether phonological awareness was really, by itself, the best predictor of first-grade reading scores, then this could clearly be argued. If there is no real difference between the models, we will pick the simplest model. If the question were which combination of factors could produce the highest level of explanatory value, the more complex model10 would be the best choice. As you can see, finding the best fit of a model is not a simple endeavor, and it is considerably lengthened by including more parameters! As Crawley (2002, p. 484) says, "If you really care about the data, this can be a very time-consuming business."

### 7.7 Examining Regression Assumptions

The assumptions of multiple regression are that the distribution of the data is normal, that the variances of variables are equal (homoscedasticity), that the variables are not too highly intercorrelated (multicollinearity), and that the data are independent. Crawley (2007) says the most important assumptions are those of constancy of variance and normal distribution, and these are the most common problems that occur in real data sets.

In R, the one simple command plot(model) will call up a series of residual plots. For this section I'll work with the model of the Lafrance and Gottardo (2005) data that only includes phonological awareness (this is model12 from the section "Further steps in finding the best fit"; the data I am using is the imputed data from the LafranceGottardo.sav file; see section 7.5, "Finding the Best Fit," for how to impute the data).

```
model12=lm(G1L1WR~PAL2,data=lafrance)
plot(model12,cex=1.5) #the cex command makes the circles bigger
```

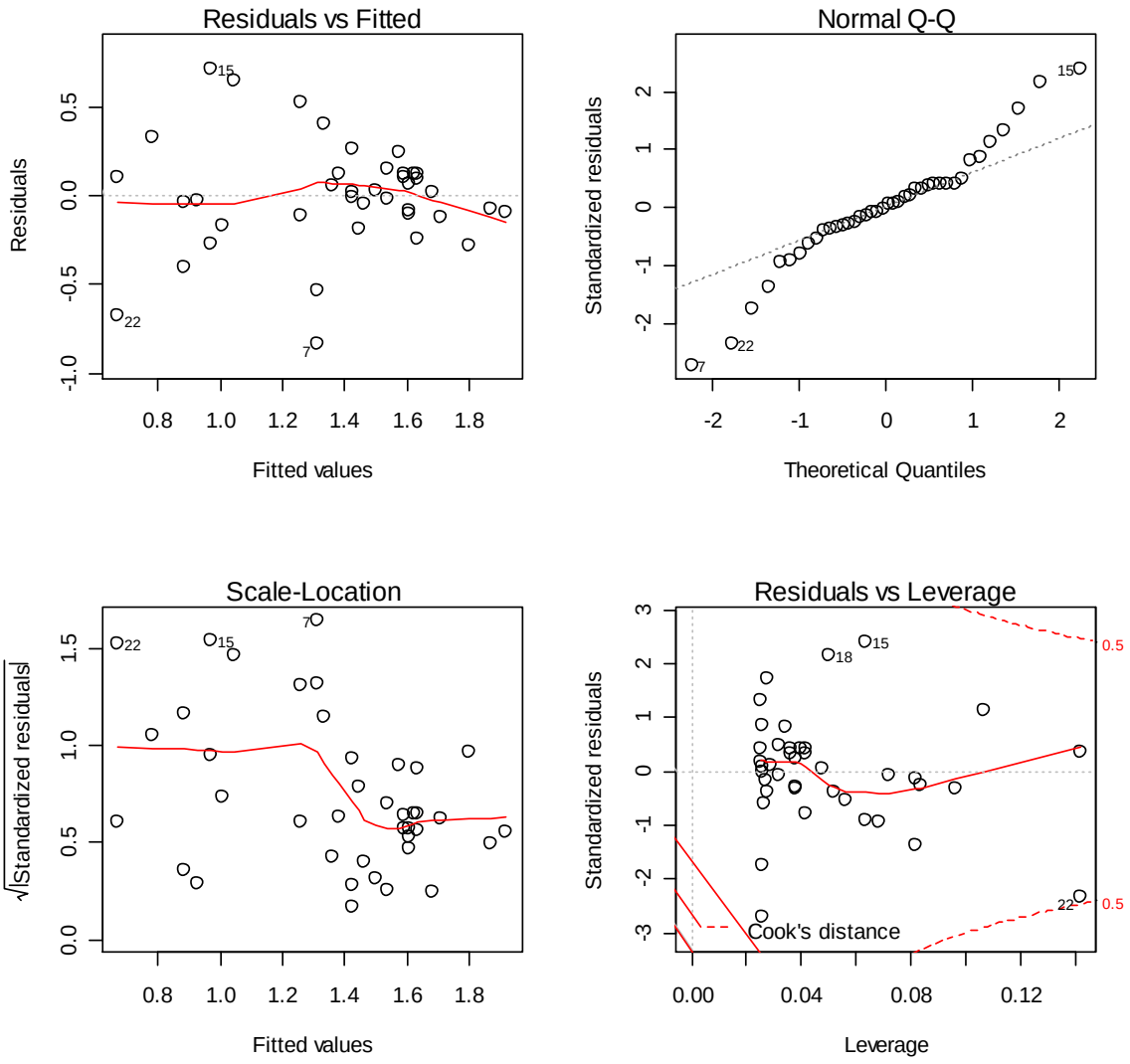


*Figure 7.9* R regression diagnostic plots.

The results of the plot() command are found in Figure 7.9. The Residuals vs. Fitted plot (top left) is used to diagnose heteroscedasticity in the data. If the data are normally distributed, this plot will show a random scattering of points both above and below the line at zero, and to the left and right of the graph. However, the distribution on the right side is more restricted than that on the left, indicating that the variance is not the same at every point in the distribution and that the homogeneity of variances assumption has been violated. Sometimes a transformation of the some of the variables can help correct this problem, but Lafrance and Gottardo had already transformed several of their variables to begin with.

The Normal Q-Q plot (top right of Figure 7.9) examines the normality distribution assumption. The standardized residuals are plotted against a normal distribution. This Q-Q plot shows a deviation away from a straight line, especially at the ends of the distribution. Fox (2002b, p. 29) states that such a pattern of pulling away from the comparison line at the ends indicates that the residual distribution is heavy-tailed. Again, this would indicate that the response variable needs transformation to fit the normality assumptions, or that we would do

better to use a method of robust or resistant regression rather than the least squares fit used here.

The Scale-Location plot (bottom left) is very much like the residuals plot except that the square roots of the standardized residuals are plotted. Crawley (2002) says this plot is useful for detecting non-constant variance. Again, we see the cramping of values toward the right end of the graph, indicating non-constant variance.

The Residuals vs. Leverage plot (bottom right) is intended to help you identify possible outliers that may have an undue amount of influence over the analysis. The dotted lines identify Cook's distance, which is a measurement which is designed to highlight particularly influential data points. Point 22 is the highest in leverage (looking along the x-axis), and it is also quite close to Cook's distance, which means it is having a large effect on the analysis. According to Crawley (2007), we could try removing this data point and check what influence it has on the regression analysis and assumptions, and report the results both with and without the data point to the reader. It seems to me more objective, however, to use robust methods of checking for outliers and then robust regression analysis.

Here is how we could model the fit of the data without point 22, where the exclamation point ("!") indicates "is not":

```
model13=update(model12, subset=(lafrance !=22))
```

This update does not substantially change the problems we find with non-normality and inequality of variances, and knowing that manually removing outliers results in non-independence would lead me to leave it in.

Another plot that can investigate the influence of outliers can be found in the plot(model) command as well, but must be called for by the which argument. This diagnostic measures how much the analysis would change if each point were removed. Therefore smaller values are better, while a value larger than 1.0 would be unusual (Howell, 2002, p. 561).

```
plot(model12,which=4)
```

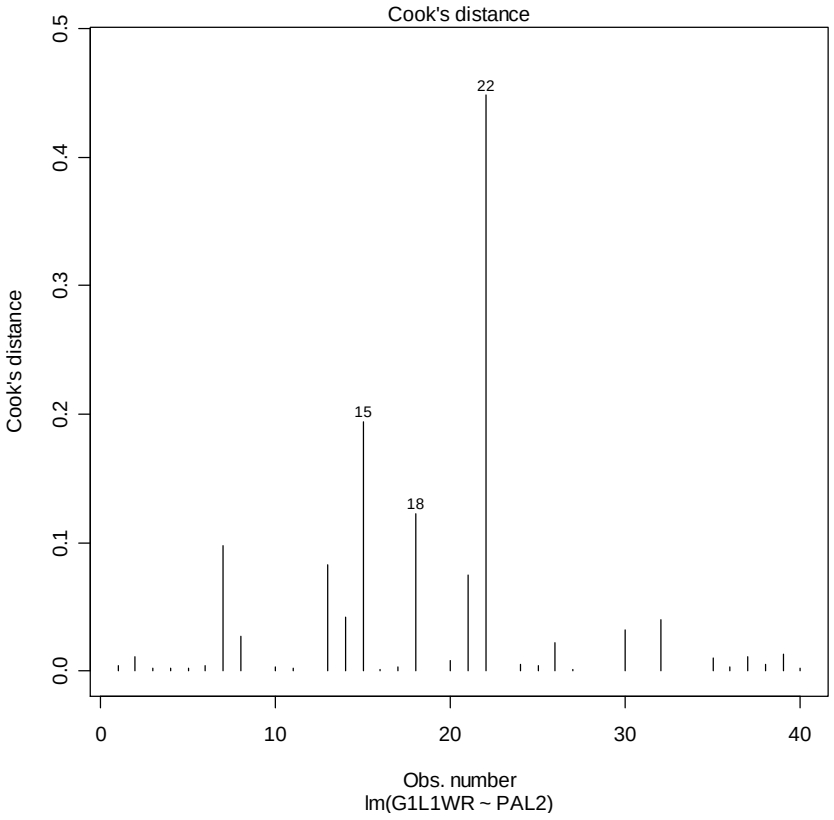


*Figure 7.10* Cook's d influential point graph.

In Figure 7.10, point 22 is clearly quite different from the rest of the points, but if you cut out 22 you might then consider 15 an outlier as well. The problem with just looking for outliers (instead of using robust methods) is that it is a subjective way of looking for outliers. If you do find outliers in your data set, you need to understand that this can radically alter the accuracy of your model. You should first check to make sure the outliers have been accurately entered and are not a data entry mistake. If they are fine, you should consider using a robust regression that can deal with outliers (robust regression is detailed further on in this chapter).

Another assumption of multiple regression is that the variables will not be highly intercorrelated (multicollinearity is undesirable). One way of checking for multicollinearity is a diagnostic called the variance inflation factor, or VIF. Heiberger and Holland (2004, p. 243) say that VIF values of over 5 are evidence of collinearity. Fox (2002b, p. 217) states that the square root of the VIF indicates "how much the confidence interval for $\beta_j$ (the standardized regression coefficient) is expanded relative to similar, uncorrelated data" and thus provides a very direct measure of the harm that collinearity may be doing to the model.

To call for the VIF in R, use the command vif(model). Note that this command does not work if there is only one explanatory variable, as there cannot be multicollinearity with only one explanatory variable! To show how this command works, then, I will use a model of the Lafrance and Gottardo (2005) data with the five original explanatory variables:

```
model.vif=lm(G1L1WR~NR+KL2WR+NS+WM+PAL2,data=lafrance)
vif(model.vif)
```

```
       NR     KL2WR        NS        WM      PAL2
1.416689 2.327490 2.568530 1.856270 3.676737
```

The VIF values are all under 5, so we do not see any strong evidence of multicollinearity.

For the assumption of independence of observations, if the data are ordered in some way, for example if we inadvertently tested the most eager students first while the least eager students came in last, there would be some sort of ordering dependency that would be visible in a scatterplot of the residuals against their case number. If your data points are numbered by some sort of ID number you can plot by using the name of that column. For the Lafrance and Gottardo data, the participants are not numbered, so we just use their row number to identify them.

We will use the function stdres in the library MASS to examine independence.

```
library(MASS)
plot(row.names(lafrance),stdres(model12),cex=1.5)
```

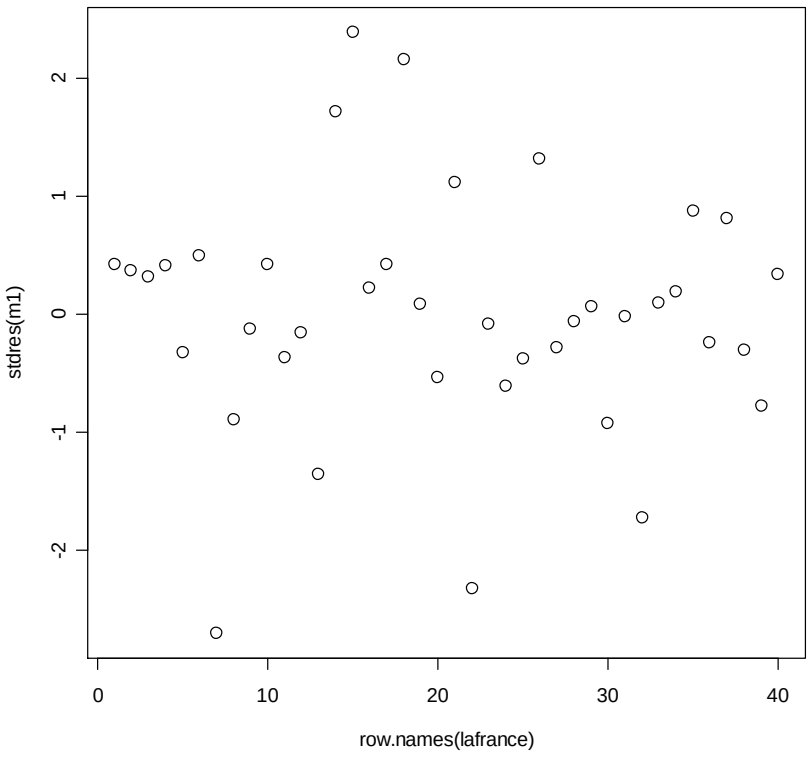


*Figure 7.11* Plot of standardized residuals against ID to check for independence.

In looking at the output of this plot in Figure 7.11 you should see whether you perceive any order or pattern to the residuals. If there is no pattern you can conclude there is independence of observations.

From this inspection of all four assumptions for regression we can conclude that there are problems in the data set with non-constant variance and non-normality. As has been noted

before, sometimes transformation of variables can help to correct some of these problems, but Lafrance and Gottardo had already transformed these variables and the problems remained. Because of this, I suggest that robust regression and bootstrapping may help to gain a more accurate picture of what is happening. Of course, one could always simply report the results of the regression and let readers know that certain assumptions are not satisfied. The practical importance of not satisfying assumptions is that power is lost.

*Examining Regression Assumptions in R*

1. Plotting your model will return four diagnostic graphics:

```
plot(model1)
```
Residuals vs. Fitted values—tests for homoscedasticity; you want random points
Normal Q-Q plot—tests for normal distribution; data should be linear
Scale-Location—tests for homoscedasticity; you want random points
Residuals vs. Leverage—identifies influential outliers

2. You can also call two other plots:

```
plot(model1, which=4)
```
returns a plot of Cook's d—identifies influential outliers
```
plot(model1, which=6)
```
returns a plot of Cook's distance vs. leverage

3. Check for multicollinearity using the variance inflation factor (VIF). Problematic scores are over 5.

```
vif(model1)
```

4. Call for a plot of standardized residuals against ID—tests for independence of observations.

```
library(MASS)
plot(row.names(lafrance),stdres(model1))
```

### 7.8 Application Activities for Finding the Best (Minimally Adequate) Fit

1. Howell (2002). Import the HowellChp15Data.sav file as howell. Chapter 15 in Howell included a data set where students rated their courses overall and also aspects of the courses on a five-point scale (where 1 = very bad and 5 = exceptional). Use the Overall variable as the response variable and the other variables (teaching skills of instructor, quality of exams, instructor's knowledge of subject matter, the grade the student expected in the course where F = 1 and A = 5, and the enrollment of the course) as explanatory variables. First take a look at a scatterplot matrix to make sure the relationships are linear (exclude any variables which are clearly non-linear). Find the minimal adequate model (try doing this by hand for this activity) and report the unstandardized coefficients and the $R^2$ for the model with only statistical predictors. Calculate the relative importance of the remaining terms of the regression equation. Comment on regression assumptions by examining residual plots.

2. Dewaele and Pavlenko Bilingual Emotions Questionnaire (2001–2003). Use the BEQ.Swear.sav file (import as beqSwear). Let us take as a given that the variables that might help explain how frequently a person swears in their L2 (swear2) are the frequency

that the person uses their L2 (l2freq), the weight they give to swearing in their L2 (weight2), and their evaluation of their speaking and comprehension skills in L2 (l2speak, l2_comp). First take a look at a scatterplot matrix to make sure the relationships between these variables are linear. Conduct an analysis to determine which of these variables effectively predicts frequency in swearing in an L2 until you arrive at the minimal adequate model (you may try this by hand or use the boot.stepAIC() command). Calculate the relative importance of the remaining terms of the regression equation. Report the unstandardized coefficients and the $R^2$ for the model with only statistical predictors, and comment on regression assumptions.

3. Larson-Hall2008.sav (import as larsonhall2008). Are amount of hours of input in a foreign language (totalhrs), aptitude (aptscore), and scores on a phonemic task (rlwscore) useful predictors of scores on a grammaticality judgment test (gjtscore)? Perform a hierarchical regression using the GJT (gjtscore) as the response variable, and the three other variables as explanatory variables. First take a look at a scatterplot matrix to make sure the relationships are linear. If they are not, think about adding in quadratic (squared) terms. Conduct an analysis to determine which of these variables effectively predicts frequency in swearing in an L2 until you arrive at the minimal adequate model (you may try this by hand or use the boot.stepAIC() command). Calculate the relative importance of the remaining terms of the regression equation. Report the unstandardized coefficients and the $R^2$ for the model with only statistical predictors, and comment on regression assumptions.

### 7.9 Robust Regression

As can be seen in the previous sections on calculating multiple regression, real data often violates the fundamental assumptions in regression of homogeneity of variances and normal distribution of data. Violations of assumptions can result in real problems statistically interpreting data. Wilcox (2001, p. 205) says that when regression models have one outlier this can "mask an important and interesting association" and distort the main shape of the distribution, and Wilcox (2005) points out that, with a heavy-tailed distribution (which can result with outliers), the probability of a Type I error (rejecting the null when it is true) can be as high as 50%. When variances are unequal (violating the assumption of homoscedasticity), this can result in low power to find statistical differences, which is a Type II error (Wilcox, 2001). Robust models can help avoid some of these problems, but Wilcox (2005) notes that no one robust regression method can be recommended in all situations. However, Wilcox (2001) remarks that the worst choice for regression is to simply continue to use the least squares estimator (the one that is used in the regression we have been doing up to this point) and hope that all is well! According to Wilcox (2001, p. 206), the conventional least squares estimator used in classical regression as an estimate of location does not "perform relatively well over a reasonably wide range of situations." Tukey (1975) many years ago said, "just which robust/resistant methods you use is not important—what is important is that you use some. It is perfectly proper to use both classical and robust/resistant methods routinely, and only worry when they differ enough to matter. But when they differ, you should think hard."

Wilcox (2005) lists some methods which he says perform well over a wide range of situations, and these include the Theil–Sen estimator, M-estimators (one with Huber's psi and the other called the Coakley–Hettmansperger estimator), the MGV estimator, the STS estimator, least trimmed squares, the least absolute value estimator, and the deepest regression line method.

One way these approaches differ is in their breakdown points. A breakdown point of a mean is the "smallest proportion of observations that can make it arbitrarily large or small. Said another way, the finite-sample breakdown point of the sample mean is the smallest proportion

of n observations that can render it meaningless" (Wilcox, 2003, p. 59). The breakdown point of least squares regression is 1/n, meaning that just one point can significantly change the analysis. The breakdown points of robust estimators range from about .2 to about .5. Wilcox recommends performing robust regression analysis with an estimator with a smaller breakdown point (.2 or .3) and a larger breakdown point (.5) and comparing the two estimates. He advises that, if the results are similar, one should choose the estimator with the highest power and smallest confidence intervals to estimate a 95% CI. However, if results are different, Wilcox (2003) advises researchers to plot regression lines to find out why the results are different.

### *7.9.1 Visualizing Robust Regression*

Robust regression includes several ways of objectively detecting outliers in a regression model. One of my favorite graphic ways to think about this is the tolerance ellipse from the robustbase library because it is easy to understand. I think this is especially effective when you compare the robust and classical ellipses. I will demonstrate how this works first with a regression with only one parameter, and then show how it can be used with more than one parameter.

This tolerance ellipse is drawn around the data that would be used in a "classical" regression and also the data that would be used in a robust regression, using the minimum-covariance determinant (MCD) estimator. The MCD has a breakdown point of .5 and searches for the half of the data that is "most tightly clustered together" (Wilcox, 2005, p. 216). If you are going to follow along with me here, import the Lafrance3.csv file, and call it Lafrance3. With the one-parameter model (PAL2) for the Lafrance and Gottardo (2005) variable G1L1WR, here is the call:

```
library(robustbase)
covPlot(cbind(Lafrance3$G1L1WR,Lafrance3$PAL2),which="tolEllipsePlot",
classic=T, cor=T)
```

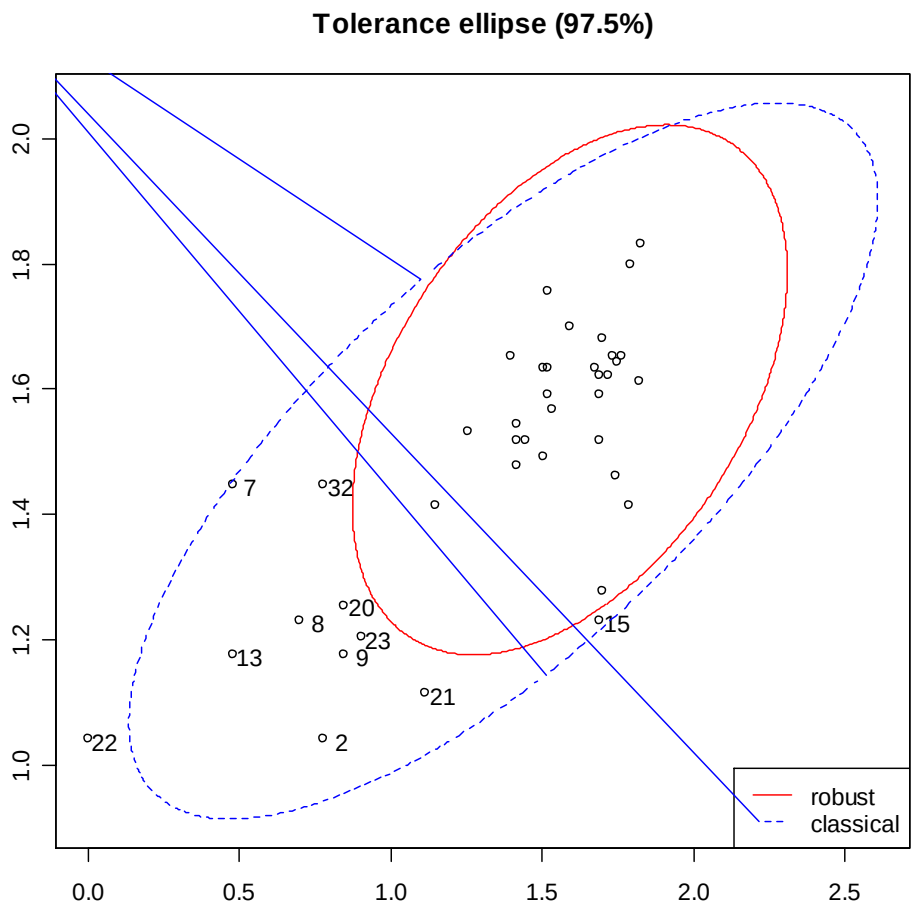


*Figure 7.12* Tolerance ellipse for G1L1WR~PAL2 regression (one-parameter).

As can be seen in Figure 7.12, quite a number of the original points will be left out in the robust correlation that uses the MCD estimator. This same type of diagnostic can be performed when there is more than one parameter as well (using the robust library). The way this is done, however, you will need to have a data frame or matrix with only the variables you are plotting in it. Lafrance3 was constructed to contain only the one response variable and three explanatory variables that I wanted to examine.

```
library(robust)
lafrance1.fit=fit.models(
        list(Robust="covRob",
        Classical="ccov"),
        data=Lafrance3)
```

This command works because I am using the data with the imputed values, but the fit.models() command doesn't like NAs in the data. To quickly clear your data of any NAs (not the best approach, I need to note though), you could do this:

```
Lafrance3<-na.omit(Lafrance3)
```

Now to examine fit assumptions.

plot(lafrance1.fit)

```
Make plot selections (or 0 to exit):


1: plot: All
2: plot: Eigenvalues of Covariance Estimate
3: plot: Sqrt of Mahalanobis Distances
4: plot: Ellipses Plot
5: plot: Distance - Distance Plot

Selection: 4
```

With selection 4, you will see a graph like the following (this one has been modified with shorter names!):

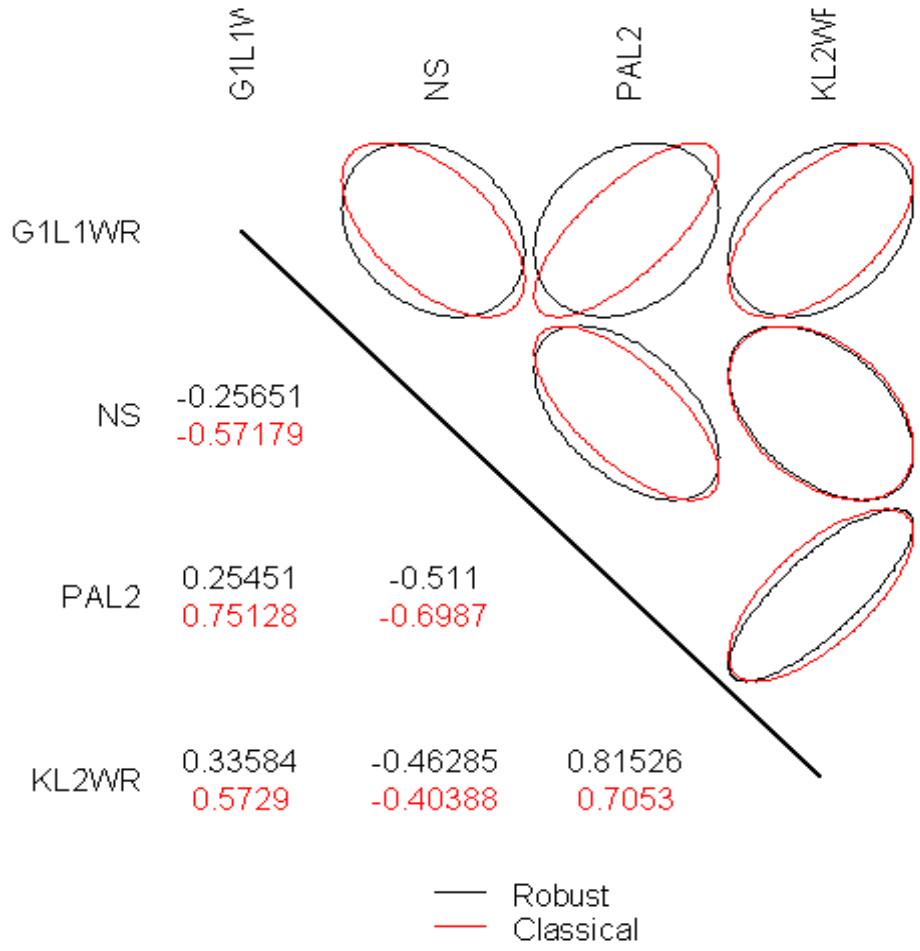


*Figure 7.13* Tolerance ellipses for G1L1WR~PAL2+NS+KL2WR regression (three-parameter).

This plot will compare the estimates (on the left of the diagonal line) of robust and classical correlations for several pairs of variables. The robust estimates are most different in the three ellipses in the G1L1WR horizontal row.

*Comparison of Classic and Robust Fits with Tolerance Ellipses*

1. Open up the robustbase library

```
library(robustbase)
```

2. If you have just two variables, specify them as vectors and bind them together:

```
covPlot(cbind(lafrance$G1L1WR,lafrance$PAL2),which="tolEllipsePlot",
classic=T)
```

3. If you have a matrix of variables, create a fit model first. This will use all of the variables in your data set, so subset first if you need to, and then plot it. This command comes from the robust library:

```
library(robust)
your.fit=fit.models(
        list(Robust="covRob",
        Classical="ccov"),
        data=yourdata)
plot(your.fit)
#Choice 4 returns tolerance ellipses
```

### *7.9.2 Robust Regression Methods*

If you already understand how to conduct a regression in R, then conducting a robust regression is quite easy. While there is a variety of commands in R that can perform robust regressions, I'll introduce you to the ones in the robust library, as their output directly parallels the output we have seen previously for the lm command. A different robust command in R that many authors have introduced in their books (Crawley, 2007; Faraway, 2005; Fox, 2002b; Jurečková & Picek, 2006) is the rlm command in the MASS library. I won't go through that one because I don't find the output as informative. The robust library was not available in R until around 2005–2006, which may be the reason earlier books did not mention it. Another library that I won't explore here, but that you might be interested in, is the robustbase library. According to Konis (2007), one of the authors of the robust library, robustbase provides a more technical approach to robust regression fitting than robust.

The main function you will need in the robust library is called lmRob (robust linear model). Instead of using least squares estimates as linear regression does, other kinds of more robust estimators are used in robust regression. According to its documentation, the lmRob command can use either the MM-estimator proposed by Yohai (1987) or an adaptive estimate derived from work by Gervini and Yohai (1999). Maronna, Martin, and Yohai (2006, p. 144) recommend the method employed in lmRob, which is to use "an MM-estimator with bisquare function and efficiency 0.85, computed from a bisquare S-estimate." These authors assert that M-estimators will perform nearly as well as least squares if the data are normally distributed, while outperforming least squares when they are outliers or heavy tails in the distribution.

We will take a look at robust regression on the Lafrance and Gottardo data with three predictors—phonological awareness (PAL2), Kindergarten L2 reading performance (KL2WR), and naming speed (NS). The lmRob model uses resampling, and the procedure is automatized so that, if there are fewer than 250 observations and two variables or fewer than

80 observations and three variables, exhaustive sampling is used. If numbers are bigger, lmRob uses different methods of resampling. If you want to control some parameters such as the type of resampling that is done, the number of resampling subsets used, the robust estimate that is used (MM or adaptive), the efficiency of the estimates, or which loss functions to use, you can change these with the lmRob.robust.control command (I will demonstrate how to do this).

First of all, I will use the robust library's fit.models command, which allows us to compare both robust and classic analyses side by side (I am using the Lafrance3 file, which is a .csv file).

```
library(robust)
lafrance3.fit=fit.models(
        list(Robust="lmRob", LS="lm"),
        formula=G1L1WR~ PAL2*KL2WR*NS,
        data=Lafrance3)
summary(lafrance3.fit)
```

Notice that you should use this fit.models() command exactly as I have given it here. You will fill in your own data only in those portions which I have underlined below:

```
lafrance3.fit=fit.models(
         list(Robust="lmRob", LS="lm"),
        formula=G1L1WR~PAL2*KL2WR*NS,
        data=lafrance3)
```

You will give the function your own name, put in the formula you have determined, and name the data set that the formula variables come from. Otherwise, you should keep the syntax exactly the same. When you summarize, you will get a large amount of output:

```
Calls:
Robust: lmRob(formula = G1L1WR ~ PAL2 * KL2WR * NS, data = lafrance3)
    LS: lm(formula = G1L1WR ~ PAL2 * KL2WR * NS, data = lafrance3)
Coefficients:
                          Value Std. Error    t value  Pr(>|t|)
Robust    (Intercept)  -4.0558877  11.086643 -0.3658355 0.7171430
    LS    (Intercept)  -1.6932436   8.805536 -0.1922931 0.8488529
Robust           PAL2   4.2053676   7.571610  0.5554126 0.5828721
    LS           PAL2   2.6556854   6.099254  0.4354115 0.6664874
Robust         KL2WR  -8.0776478  55.478977 -0.1455984 0.8852458
    LS         KL2WR  30.2125284  42.367283  0.7131099 0.4814772
Robust            NS   1.4834241   4.669808  0.3176628 0.7530168
    LS            NS   0.3981973   3.730261  0.1067478 0.9157241
Robust     PAL2:KL2WR   4.3429312  33.836803  0.1283493 0.8987585
    LS     PAL2:KL2WR -18.7522847  25.839986 -0.7257080 0.4738341
Robust        PAL2:NS  -1.2644620   3.258865 -0.3880069 0.7008445
    LS        PAL2:NS  -0.5534157   2.639372 -0.2096770 0.8353858
Robust       KL2WR:NS   4.5455879  25.691350  0.1769307 0.8607928
    LS       KL2WR:NS -13.3703450  19.569354 -0.6832287 0.4998849
Robust PAL2:KL2WR:NS  -2.4740433  15.668935 -0.1578948 0.8756340
    LS PAL2:KL2WR:NS   8.3333723  11.936736  0.6981283 0.4906575
```

```
Residual Scale Estimates:
Robust: 0.2330676 on 29 degrees of freedom
    LS: 0.3196538 on 29 degrees of freedom

Multiple R-Squared:
Robust: 0.3658876
    LS: 0.5973927
Bias Tests for Robust Models:
Robust:
            statistic   p-value
M-estimate  0.5652855 0.9997877
LS-estimate 3.6125604 0.8902805
```

As you can see, this output directly parallels that obtained from the non-robust lm function, and this printout gives a comparison with the least squares (LS) estimation of the same data. We see the parameter estimates under "Value," their standard errors, a t-test for whether each term is statistically different from 0, and the probability of obtaining a value as large as or larger than the t-value. Note that some of the robust estimates are quite different from the classic estimates, for example in the KL2WR main effect estimate.

The residual standard error (under "Residual Scale Estimate") and the $R^2$ value are also printed. Note that the $R^2$ value is lower for the robust model, but the residual error is also lower. One additional piece of the output is the test for bias. According to the S-PLUS user's guide for the robust library (Insightful Corporation, 2002, p. 26), the meaning of this test is that it provides "two statistical tests of the bias: the first for bias of the final M-estimate relative to a highly robust initial estimate, and the second for the bias of the LS estimate relative to the final M-estimate." If the *p*-value is not below *p*=.05, it means there is little evidence for bias in these measures.

As with the non-robust regression, none of the parameters of the robust model are statistical, so we would want to go through a handwise deletion procedure in the same way as for the non-robust regression (supposedly there is an update function in this library called update.lmRob, but I cannot get it to work; if it did, this command should make it work: m2.robust=update.lmRob(m1.robust,~.-PAL2:KL2WR:NS, evaluate=T), and you can try it).

```
m1.robust=lmRob(G1L1WR~PAL2*KL2WR*NS,data=Lafrance3)
summary(m1.robust)
m2.robust=lmRob(G1L1WR~PAL2+KL2WR+NS+PAL2:KL2WR+PAL2:NS+KL2WR:
NS,data=Lafrance3) #this deletes the 3-way interaction term PAL2:KL2WR:NS
summary(m2.robust)
```

The robust library also has a robust step function called step.lmRob (boot.stepAIC won't work with lmRob objects). With step.lmRob() models are evaluated using a robust version of Akaike's final prediction error criterion called the **robust final prediction error** (RFPE). As with the AIC, the smaller the RFPE is, the better. Trying the step function with the full model of the three variables above does not result in any change, but if I put in just the main effects the step function tells me it would be good to get rid of the KL2WR term.

```
m3.robust=lmRob(G1L1WR~PAL2+KL2WR+NS,data=Lafrance3)
step.lmRob(m3.robust)
```

```
Start:  RFPE= 28.6874
 G1L1WR ~ PAL2 + KL2WR + NS


Single term deletions

Model:
G1L1WR ~ PAL2 + KL2WR + NS

scale:  0.2011768

       Df   RFPE
<none>    28.687
PAL2    1 31.697
KL2WR   1 28.498
NS      1 28.863
```

The output shows that the RFPE is 28.6874 to start with. Deleting single terms one by one results in different RFPE numbers. Specifically, by deleting KL2WR the RFPE goes down to 28.498, which is better than the starting number. So the step procedure recommends getting rid of this term and gives the unstandardized regression coefficients and residual error (but not $R^2$).

```
Call:
lmRob(formula = G1L1WR ~ PAL2 + NS, data = Lafrance3)

Coefficients:
(Intercept)         PAL2           NS
  0.7524942    1.1458749  -0.4625900

Degrees of freedom: 37 total; 34 residual
Residual standard error: 0.2036922
```

The $R^2$ of this model can be obtained by the summary:

```
m4.robust=lmRob(G1L1WR~PAL2+NS,data=Lafrance3)
summary(m4.robust)
```

You can also perform regression diagnostics by using the plot.lmRob command. To perform this command you cannot have any NAs in your data, or you will receive a warning error about incorrect number of dimensions. When you use this command you can see a selection of choices first of all:

```
plot.lmRob(m1.robust)
```

```
Make plot selections (or 0 to exit):


 1: plot: All
 2: plot: Normal QQ-Plot of Residuals
 3: plot: Estimated Kernel Density of Residuals
 4: plot: Robust Residuals vs Robust Distances
 5: plot: Residuals vs Fitted Values
 6: plot: Sqrt of abs(Residuals) vs Fitted Values
 7: plot: Response vs Fitted Values
 8: plot: Standardized Residuals vs Index (Time)
 9: plot: Overlaid Normal QQ-Plot of Residuals
10: plot: Overlaid Estimated Density of Residuals

Selection:
```

I especially like to use the plot command with the fit.models object, because this returns side-by-side graphs comparing the robust and classic fit.

plot(lafrance3.fit)

If you use the fit.models method, this plot would return the following graphs with the original three-term regression model. First is the Q-Q plot checking for the normal distribution in Figure 7.14.

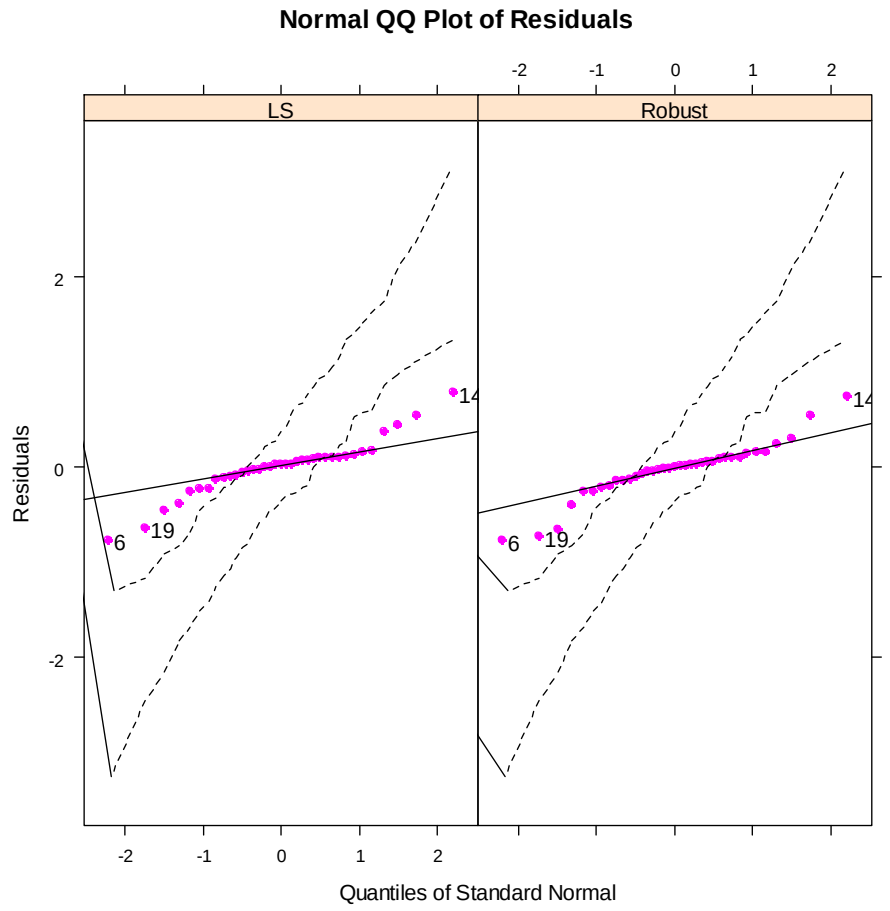


*Figure 7.14* Side-by-side least squares and robust Q-Q plots.

Although the majority of points fit the line with both the least squares (LS) and robust estimators, this is a problem for LS (because it assumes strict normalcy) but not for the robust estimator, which, having a breakdown point of .5, is fine if the middle portion satisfies the regression assumptions.

The next plot (shown in Figure 7.15) is a kernel probability density estimate. Here there is not much difference between the two density plots; both show the bulk of the data points are centered around zero, but also have bumps on either side of the tall midpoint that indicate there are outliers in the data.

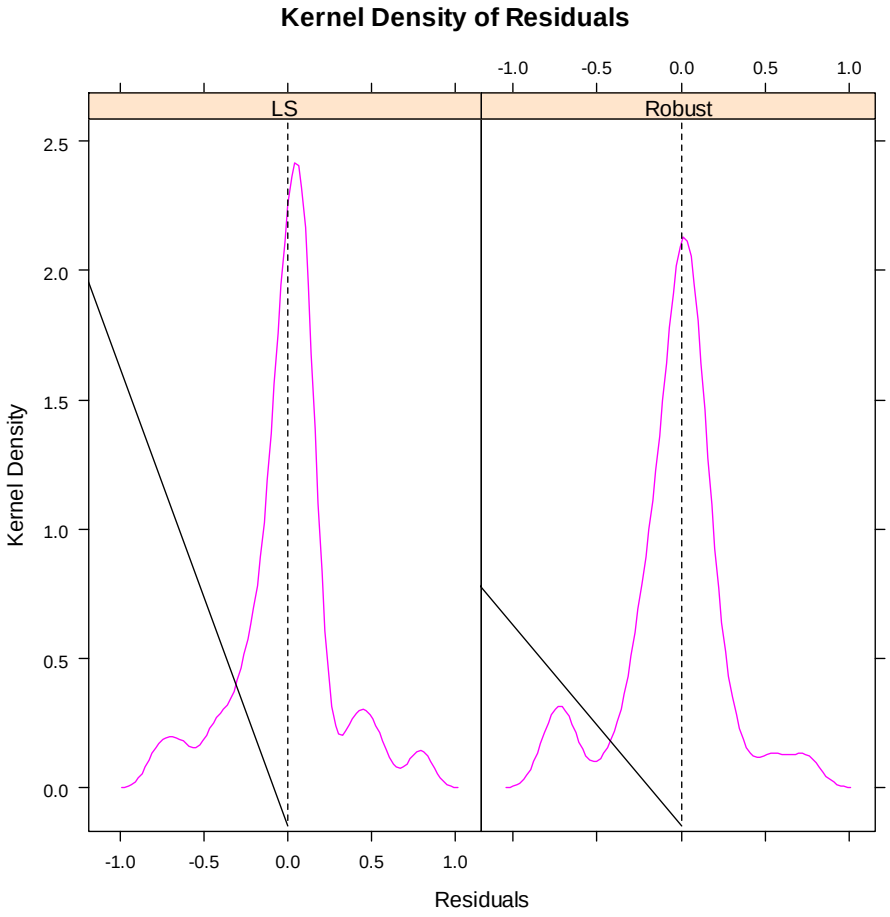


*Figure 7.15* Side-by-side least squares and robust kernel probability density plots.

The next figure (Figure 7.16) shows a plot of scaled residuals versus robust distances. According to the documentation for the robust library (Insightful Corporation, 2002), points that fall to the right of the vertical dotted line are considered leverage points and are called x-outliers.[1] These points might have undue influence on the analysis. Points above and below

[1] To obtain this document, go to http://cran.r-project.org/src/contrib/robust_0.3-11.tar.gz and download it on to your computer. (These versions change often and the title of the document may change as well, so if you have trouble with this website just go to http://cran.r-project.org/src/contrib/ and search for the document there.) It needs WinZip to open it. Extract it to someplace you will remember, like your desktop. Open the file that says "robust" and the file is called "Robust.pdf."

2.5 are also outside the bulk of the data, and would be labeled residual outliers (but on this graph there are none). In Figure 7.16 we see that there are three x-outliers, but these will not affect the robust analysis.

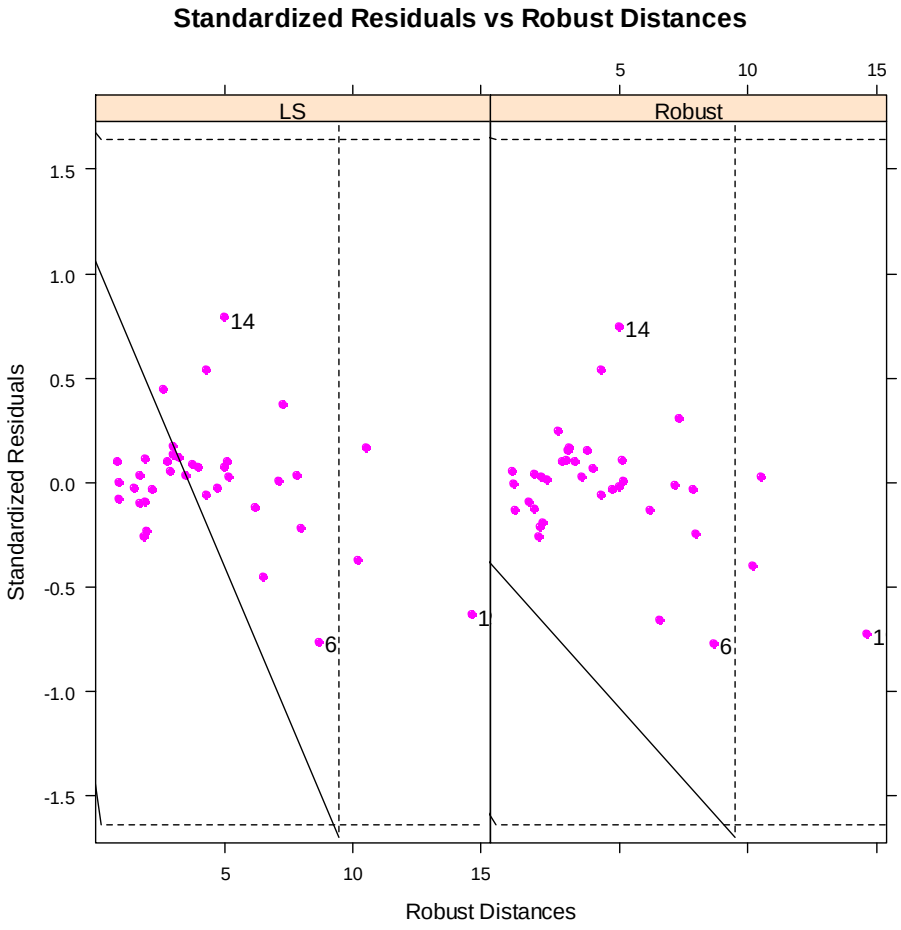


*Figure 7.16* Side-by-side least squares and robust residuals vs. robust distances plot (investigates leverage).

The next plot in the list, shown in Figure 7.17, the Residuals vs. Fitted values plot, is one we've seen before, which investigates homoscedasticity. The pattern here does look somewhat like a pie shape, and it appears we still have heteroscedasticity in the robust regression.

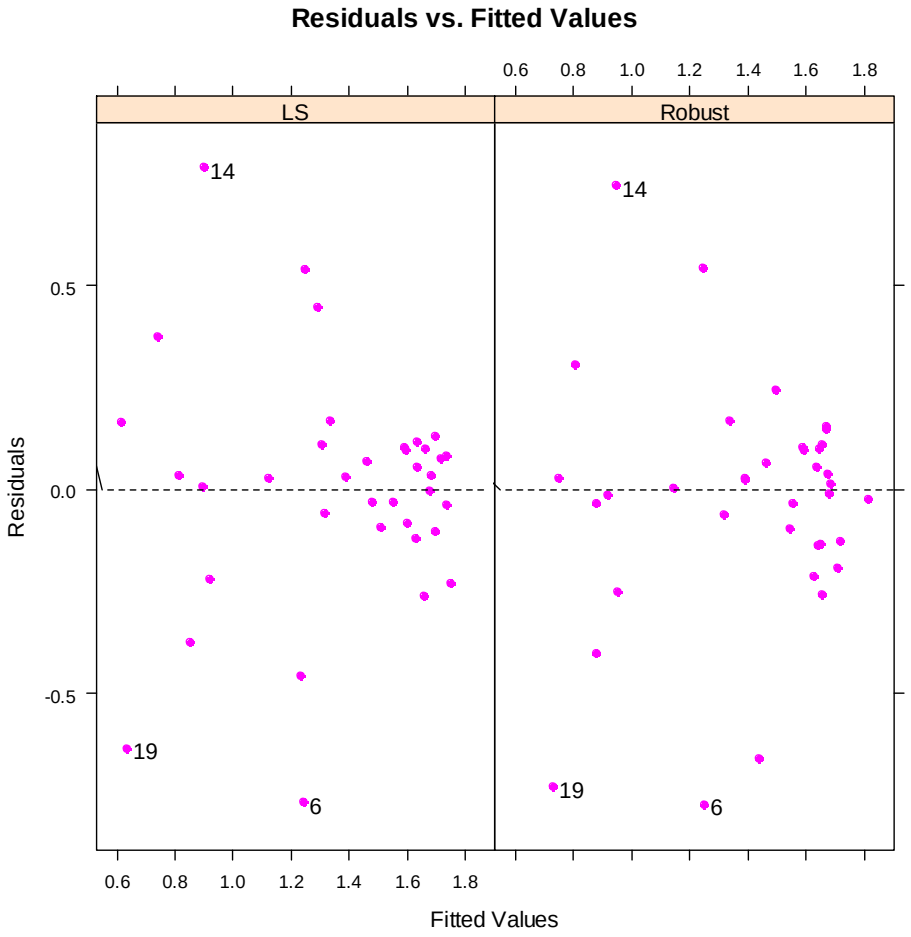


*Figure 7.17* Side-by-side least squares and robust Residuals vs. Fitted values plot.

The rest of the plots are not ones we will investigate. Plots 9 and 10 are exactly like Plots 2 and 3 except that they put the LS and robust data on the same plot, overlaid.

Last of all, in this section I will show you how to control some of the aspects of the robust regression. You will need the lmRob.control command, which can be set for a variety of parameters which I explain below.

```
my.control=lmRob.control(initial.alg="Auto", final.alg="MM", efficiency=.95)
```

| | |
|---|---|
| my.control= lmRob.control ( . . .) | Make an object that specifies control parameters; then you will put this into the regression model later. |
| initial.alg="Auto" | Gives choice of initial algorithm for resampling; choices are "Auto" (determined by size of data set), "Random," "Exhaustive," or "Fast." |
| final.alg="MM" | Gives choice of estimator; choices are "MM" and "Adaptive." |
| efficiency=.95 | MM-estimates have a 90% efficiency compared to LS estimates. However, when you increase efficiency you get less protection from bias due to outliers. Therefore, you may want to use a smaller efficiency such as .85 or .8. |

This specification for control (which is currently set to the defaults) is then inserted into the model object, like this:

m1.robust=lmRob(G1L1WR~PAL2*KL2WR*NS, control=my.control, data=lafrance3)

*Robust Regression*

1. There are many libraries for robust regression, but I focused on the lmRob command in the robust library. This works just the way that lm works:

```
m1.robust=lmRob(G1L1WR~PAL2*KL2WR*NS,data=Lafrance3)
summary(m1.robust)
m2.robust= lmRob(G1L1WR~PAL2+KL2WR+NS- PAL2:KL2WR:NS,
data=Lafrance3)
anova(m1.robust,m2.robust)
plot(m1.robust) #for diagnostic plots
```

**7.10 Application Activities with Robust Regression**

1. Lafrance and Gottardo (2005) data. Compare the model you got for the "Multiple Regression.Application activity_Multiple regression" section, item 2, using least squares estimates with a robust model. Mention diagnostic plots.

2. Larson-Hall (2008) data. Compare the model you got for the "Multiple Regression.Application Activity_Finding the best fit" section, item 3, using least squares estimates with a robust model. Mention diagnostic plots.

3. Howell (2002). Compare the model you got for the "Multiple Regression.Application Activity_Finding the best fit" section, item 1, using least squares estimates with a robust model. Mention diagnostic plots.

4. Dewaele and Pavlenko data, beq.swear file. Compare the model you got for the "Multiple Regression.Application Activity_Finding the best fit" section, item 2, using least squares estimates with a robust model. Mention diagnostic plots.