Multiple Correlation

Andrew Johnson

Load Libraries

We will need the `psych` and `ppcor` libraries for this demonstration.

```
library(psych)
library(ppcor)
```

The Data

The International Personality Item Pool (IPIP) is a public domain item pool that may be used freely to create personality scales and measures that assess a variety of commonly used personality constructs. One such construct is **neuroticism**, a personality variable that has been assessed using such well-known instruments as Costa and McCrae's NEO-PI-R, and Eysenck's EPI.

The `epi.bfi` dataset included with the `psych` package contains data on the IPIP version of the NEO-PI-R Neuroticism scale, as well as data collected using the EPI Neuroticism scale. Finally, it contains data on the Beck Depression inventory (BDI), as well as state and trait anxiety measures. A detailed description of the dataset may be seen by typing `?epi.bfi`.

```
data("epi.bfi")
```

We can use this data to illustrate multiple correlation and regression, by evaluating how the "Big Five" personality factors (*Openness to Experience, Conscientiousness, Extraversion, Agreeableness,* and *Neuroticism*) predict *trait anxiety.*

To facilitate our demonstration, we can create a dataset that includes just these variables.

```
big5 <- with(epi.bfi, data.frame(bfagree, bfcon, bfext, bfneur, bfopen, traitanx))
```

Fitting the Model with All Five Factors

Fitting a linear model is easy to do in R. Using the `lm` function, we specify a formula in the form `y ~ x1 + x2`, where `y` is the dependent variable (the *criterion*) and `x1` and `x2` are the independent variables (the *predictors*).

Let's fit a linear model for the prediction of *trait anxiety*, using all five of our personality factors:

```
model1.trait <- lm(traitanx ~ bfagree + bfcon + bfext + bfneur + bfopen,
                   data=big5)
```

We can view a summary of this model with the `summary` function

```
summary(model1.trait)
```

```
##
## Call:
## lm(formula = traitanx ~ bfagree + bfcon + bfext + bfneur + bfopen,
##     data = big5)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.2952  -4.2436  -0.7314   3.3558  21.2960
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.74134    3.55405  12.026  < 2e-16 ***
## bfagree      0.00325    0.02876   0.113    0.910
## bfcon       -0.09197    0.02144  -4.290 2.66e-05 ***
## bfext       -0.11751    0.01893  -6.208 2.56e-09 ***
## bfneur       0.26054    0.01889  13.793  < 2e-16 ***
## bfopen      -0.03756    0.02494  -1.506    0.133
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.273 on 225 degrees of freedom
## Multiple R-squared:  0.5753, Adjusted R-squared:  0.5659
## F-statistic: 60.97 on 5 and 225 DF,  p-value: < 2.2e-16
```

Or we can look at just the coefficients themselves

```
coef(model1.trait)
```

```
##  (Intercept)      bfagree        bfcon        bfext       bfneur       bfopen
## 42.741336772  0.003249864 -0.091968547 -0.117513857  0.260540623 -0.037558314
```

and their confidence intervals

```
confint(model1.trait)
```

```
##                    2.5 %      97.5 %
## (Intercept) 35.73786374 49.74480981
## bfagree     -0.05342482  0.05992455
## bfcon       -0.13421732 -0.04971978
## bfext       -0.15481477 -0.08021294
## bfneur       0.22331744  0.29776380
## bfopen      -0.08669989  0.01158326
```

From these outputs, we can see that the overall model is statistically significant, $F(5,225) = 60.97$, $p < 0.001$, with an adjusted R^2 of 0.5659, suggesting that the five factors of personality described by the Five Factor Model of personality, explain approximately 56.59% of the trait anxiety variable.

We can also see from our `summary` of the `model1.trait` object that not all of the personality factors are equally good at predicting trait anxiety. Specifically, Conscientiousness, $t(225) = -4.290$, Extraversion, $t(225) = -6.208$, and Neuroticism, $t(225) = -6.208$, are significant predictors of state anxiety, while Agreeableness and Openness are not.

Looking for a More Parsimonious Model

The model that we have currently fit includes two variables (agreeableness and openness to experience) that are not significant predictors of the dependent variable. There is nothing conceptually wrong with keeping variables in the model that are not statistically significant predictors, but if you were interested in fitting the most parsimonious (i.e., the simplest) model to your data, you might want to consider removing these variables from the analysis. There are three commonly used "automatic" methods for removing predictors from a model: (1) *forward regression*; (2) *backward regression*; and (3) *stepwise regression*. **Forward regression** involves starting with the "best" predictor (i.e., the predictor with the highest zero-order correlation with the dependent variable), and then systematically adding variables to the model until the change in model fit is no longer statistically significant. **Backward regression**, as the name implies, is basically the opposite of forward regression - you start with the most general model (i.e., the model with all of the predictors that you might want to use in predicting the dependent variable), and then delete predictors until the change in model fit is statistically significant. **Stepwise regression** is a combination of forward and backward regression methods, as it continues to check the variables in the model and the variables not in the model, to see if any are candidates for addition or deletion from the model.

Given that we have already fit the most general model, we might as well apply **backward regression**, removing variables until doing so will negatively impact on the model fit. Although it is possible to do this with the `step` function in R, it may be more informative to manually walk through the steps involved in this process.

Step 1

Let's review the model again:

```
summary(model1.trait)
```

```
##
## Call:
## lm(formula = traitanx ~ bfagree + bfcon + bfext + bfneur + bfopen,
##     data = big5)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.2952  -4.2436  -0.7314   3.3558  21.2960
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.74134    3.55405  12.026  < 2e-16 ***
## bfagree      0.00325    0.02876   0.113    0.910
## bfcon       -0.09197    0.02144  -4.290 2.66e-05 ***
## bfext       -0.11751    0.01893  -6.208 2.56e-09 ***
## bfneur       0.26054    0.01889  13.793  < 2e-16 ***
## bfopen      -0.03756    0.02494  -1.506    0.133
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.273 on 225 degrees of freedom
## Multiple R-squared:  0.5753, Adjusted R-squared:  0.5659
## F-statistic: 60.97 on 5 and 225 DF,  p-value: < 2.2e-16
```

Given that all of the predictors are on the same scale, we can determine the "worst" predictor of the dependent variable in two ways: by examining the coefficients, or by looking at the t values. In both cases, we are looking

for the lowest absolute value. Looking at the table of coefficients, we can see that agreeableness (`bfagree`) is clearly our first candidate for removal from the model. To do this, we can use the `update` function, to modify the model object for model #1.

```
model2.trait <- update(model1.trait, . ~ . - bfagree)
```

Within the `update` function, the "." is used to indicate "the same as before". We can then update this model through the use of a "+" or "-" applied to another variable in the dataset. Thus, we have specified that we want to keep the "same model as before, deleting `bfagree`".

We can now test this new model against our original model, to see if the new model fit is significantly worse than our original model.

```
anova(model1.trait, model2.trait)
```

```
## Analysis of Variance Table
##
## Model 1: traitanx ~ bfagree + bfcon + bfext + bfneur + bfopen
## Model 2: traitanx ~ bfcon + bfext + bfneur + bfopen
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    225 8854.8
## 2    226 8855.3 -1  -0.50249 0.0128 0.9101
```

The new model, in which we remove agreeableness from our list of independent variables used in the prediction of trait anxiety, is not significantly worse than the original model. This can be taken to mean that the deletion of agreeableness does not substantively change the predictive equation. We can confirm this by looking at the model object for this new model:

```
summary(model2.trait)
```

```
##
## Call:
## lm(formula = traitanx ~ bfcon + bfext + bfneur + bfopen, data = big5)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.3217  -4.2643  -0.6904   3.3585  21.3437
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.94066    3.07864  13.948  < 2e-16 ***
## bfcon       -0.09112    0.02005  -4.545 8.94e-06 ***
## bfext       -0.11682    0.01787  -6.538 4.10e-10 ***
## bfneur       0.26022    0.01863  13.964  < 2e-16 ***
## bfopen      -0.03700    0.02440  -1.517    0.131
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.26 on 226 degrees of freedom
## Multiple R-squared:  0.5753, Adjusted R-squared:  0.5678
## F-statistic: 76.54 on 4 and 226 DF,  p-value: < 2.2e-16
```

Our original model explained 56.59% of the variability (based on the adjusted R^2 value). Our new model explains 56.78% of the variability... suggesting that the amount of variability predicted by the model has actually increased, when considering the adjusted R^2! This is due to the fact that the R^2 did not change appreciably, but the number of predictors in the equation has been reduced by one. Thus, the required adjustment is less than in the first model.

Step 2

The predictor within this model that has the smallest relationship with the dependent variable is now openness to experience. We can remove this predictor from the model in the same way that we did in Step 1 of this process.

```
model3.trait <- update(model2.trait, . ~ . - bfopen)
anova(model2.trait, model3.trait)
```

```
## Analysis of Variance Table
##
## Model 1: traitanx ~ bfcon + bfext + bfneur + bfopen
## Model 2: traitanx ~ bfcon + bfext + bfneur
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    226 8855.3
## 2    227 8945.5 -1   -90.144 2.3006 0.1307
```

```
summary(model3.trait)
```

```
##
## Call:
## lm(formula = traitanx ~ bfcon + bfext + bfneur, data = big5)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.4666  -4.2950  -0.8912   3.4934  20.7364
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 41.04958    2.82294  14.541  < 2e-16 ***
## bfcon       -0.09760    0.01964  -4.968 1.33e-06 ***
## bfext       -0.12826    0.01624  -7.897 1.22e-13 ***
## bfneur       0.25143    0.01776  14.156  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.278 on 227 degrees of freedom
## Multiple R-squared:  0.571,  Adjusted R-squared:  0.5653
## F-statistic: 100.7 on 3 and 227 DF,  p-value: < 2.2e-16
```

Once again, the change in model fit is non-significant, suggesting that the removal of openness to experience did not significantly impact on the amount of variability predicted within the dependent variable. This is confirmed by a quick check of the adjusted R^2 in the new model - the variance in trait anxiety predicted by the three variables left in the model is now 56.53%.

Step 3

Chances are, you probably wouldn't go any further with your model reduction efforts, given that all of the remaining predictors are statistically significant predictors of the dependent variable. Just for illustrative purposes, however, let's take a look at what the model would look like if we removed conscientiousness from the list of independent variables.

```
model4.trait <- update(model3.trait, . ~ . - bfcon)
anova(model3.trait, model4.trait)
```

```
## Analysis of Variance Table
##
## Model 1: traitanx ~ bfcon + bfext + bfneur
## Model 2: traitanx ~ bfext + bfneur
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    227 8945.5
## 2    228 9918.3 -1   -972.79 24.686 1.328e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model4.trait)
```

```
##
## Call:
## lm(formula = traitanx ~ bfext + bfneur, data = big5)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.5691  -4.1347  -0.4803   3.9856  18.9960
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 32.46485    2.34547  13.842   <2e-16 ***
## bfext       -0.14970    0.01645  -9.098   <2e-16 ***
## bfneur       0.24826    0.01865  13.312   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.596 on 228 degrees of freedom
## Multiple R-squared:  0.5243, Adjusted R-squared:  0.5202
## F-statistic: 125.7 on 2 and 228 DF,  p-value: < 2.2e-16
```

As we might have expected, the new model (model #4) is a significantly worse fit to the data than the model from the previous step, and the variance in trait anxiety predicted by the two variables left in the model is now 52.02%.

Summary

Based on our application of backward regression (sometimes called backward elimination regression), we conclude that the most parsimonious model was model #3, wherein conscientiousness, extraversion, and neuroticism were used to predict trait anxiety.

Testing Assumptions

We now need to test the assumptions underlying linear regression:

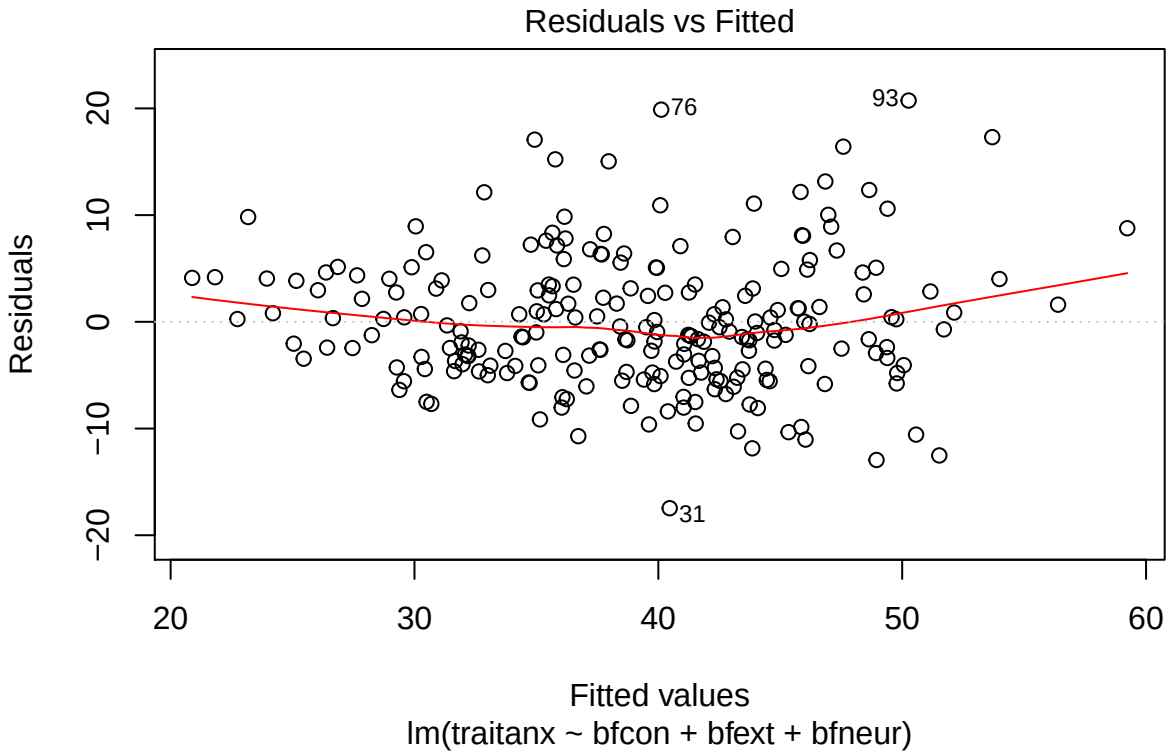
1) the relationship between X and Y is linear
2) the variability of observations about the line of best fit is constant (i.e., the assumption of homoscedasticity)
3) values are normally distributed about the line of best fit (i.e., the assumption of bivariate normality)

In the simple "two-variable case", these assumptions may be addressed by visual inspection of the scatterplot. This is not possible, however, when you have more than two variables in the relationship (as is the case in multiple regression). Thus, we will rely on the techniques that we developed in our discussion of correlation coefficients, and will test these assumptions by evaluating the residuals, or the "errors" made within the model.

Again, we will look at plots 1, 2, and 5, in testing these assumptions, and we will test our "reduced" model (model #3).

Evaluating the Assumptions of Linearity and Homoscedasticity

```
plot(model3.trait, which = 1)
```

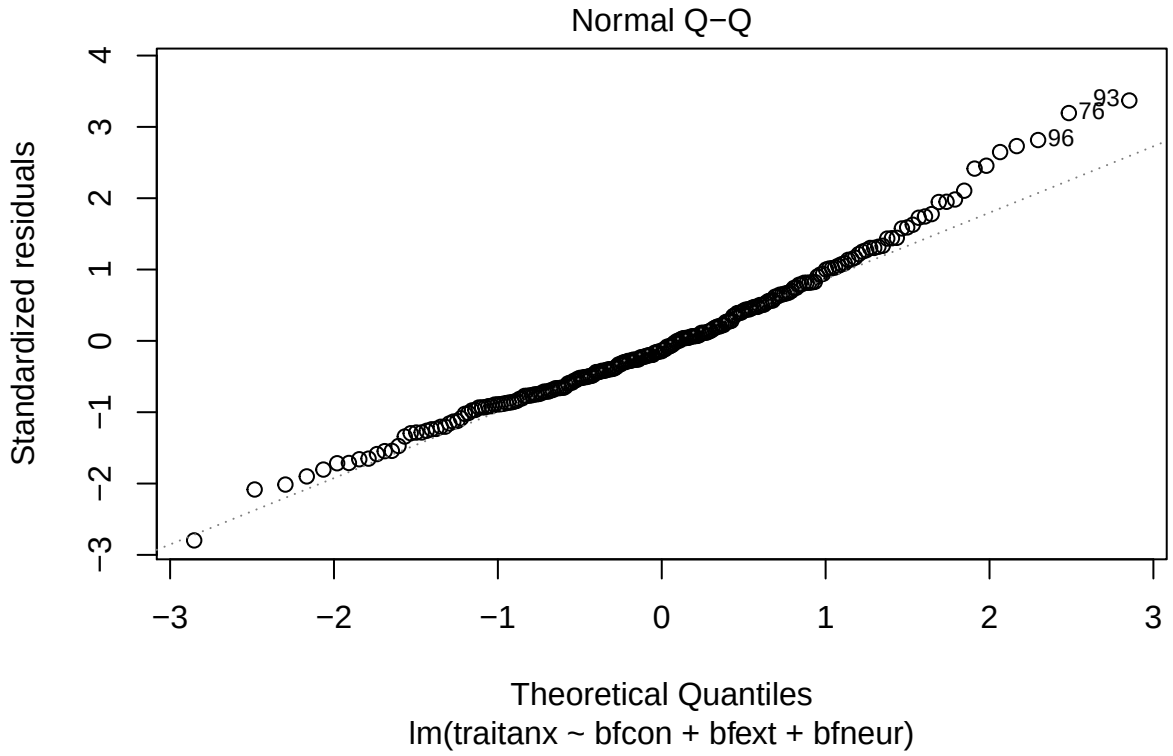


We can address our first assumption (linearity) with a plot of the residuals versus fitted values. If the relationship is linear, then the red line in the centre of the graph will be fairly flat. As you can see, this assumption is met for the present analysis (although there is a mild curvature to the line that is suggestive of a slight quadratic component within the multiple regression).

We can also address our second assumption (homoscedasticity) with this plot. If the model evidences constant variability across the range of fitted values, then the observations should be relatively evenly distributed across the range of the fitted values - in other words, the error should be relatively constant across all of the fitted values. Again, this assumption appears to be met for the present variable.

Evaluating the Assumption of Bivariate Normality

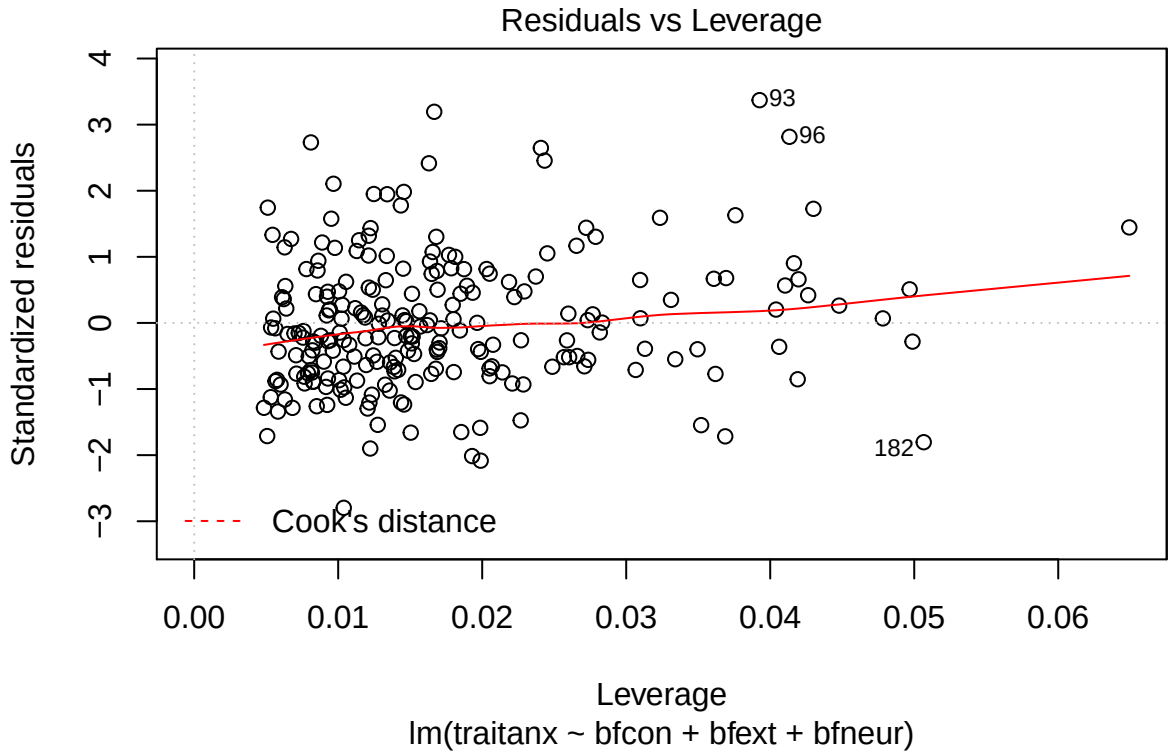
```
plot(model3.trait, which = 2)
```



We can address our third assumption (bivariate normality) with a normal Q-Q plot on the standardized residuals. In a Q-Q plot, or "Quantile-Quantile" plot, we plot theoretical quantiles on the x-axis, and the quantiles of the standardized residuals on the y-axis. If the model satisfies the requirement of bivariate normality, the points should be lined up nicely along the red line - and as you can see, they do, for our example.

Looking for Points with Unusual Influence on the Regression Line

```
plot(model3.trait, which = 5)
```



The third plot that we will look at is a plot of the residuals for each observation, against their corresponding leverage. This is useful for identifying points that may change the regression line. It uses a concept called "leverage" to describe the impact that outliers within your data may have on the slope of your regression line. The extent to which an outlier can change the slope of your line is determined by it's distance from the centre of the data (their leverage), and the magnitude of it's residual.

The importance of the leverage for each point is typically evaluated using a statistic called "Cook's distance" or "Cook's D". Cook's D is a function of the leverage and the standardized residual for a point, and may be used to estimate the amount that any given point will move the regression line. In terms of having an effect on the regression line, observations that have a Cook's D with an absolute value greater than 0.5 are likely to be a problem, and observations that have a Cook's D with an absolute value greater than 1.0 are almost certainly going to be a problem. The third graph will draw dotted lines that represent Cook's distance scores of 0.5 and 1.0 when there are points that come close to these values, within the dataset. Such points will be plotted and numbered (by their case number in the data) on this plot. As we can see, however, there are no such outlier cases within our data.

Summary of Assumption Testing

The foregoing suggests that the assumptions have been met within our dataset, with the possible exception of a mild quadratic tendency within the model. Thus, we can summarize our findings in tabular form, and report the overall model fit with some confidence.

Describing the Model

We have identified a model that fits our data, that is parsimonious in the number of variables used within the prediction equation, and have demonstrated that the assumptions of linear regression have been met. Let's get R to create a nicely formatted table with all of these values presented together:

```
coefs.trait <- coef(model3.trait)
conf.trait <- confint(model3.trait)
table.trait <- data.frame(lower=round(conf.trait[,1],3),
                          coef=round(coefs.trait,3),
                          upper=round(conf.trait[,2],3))
table.trait
```

```
##              lower   coef  upper
## (Intercept) 35.487 41.050 46.612
## bfcon       -0.136 -0.098 -0.059
## bfext       -0.160 -0.128 -0.096
## bfneur       0.216  0.251  0.286
```

I rounded the values in this table to three decimal places, to make the table more compact.

Calculating Part and Partial Correlations

You might even be interested in identifying the part and partial correlations for each of the personality factors correlated with trait anxiety, controlling for the other variables in the analysis.

We can use the `pcor` function in the `ppcor` package, to calculate the partial correlations for each personality factor with the trait anxiety variable. The `pcor` function will generate a full matrix of partial correlations (controlling for all variables other than the two variables in the bivariate correlation), along with tests of statistical significance, for all pairs of variables.

We really only want the partial correlations between each of the five personality factors and trait anxiety (i.e., we don't need the partial correlations among the three personality factors), and so we can be very specific with the output that we request.

```
partial <- pcor(big5[,c(2:4,6)])$estimate[1:3,4]
partial
```

```
##      bfcon      bfext     bfneur
## -0.3131784 -0.4642165  0.6847423
```

Effectively, we asked the `pcor` function to generate the partial correlations among all possible pairwise combinations of variables, and then provide us with the partial correlations that were computed for the relationships between each of conscientiousness, extraversion, and neuroticism, and the trait anxiety measure.

We can do the same thing to generate the semi-partial (or part) correlations, using the `spcor` function in the same package.

```
part <- spcor(big5[,c(2:4,6)])$estimate[1:3,4]
part
```

```
##      bfcon      bfext     bfneur
## -0.3016356 -0.4472417  0.6838285
```

Adding Part and Partial Correlations to the Table of Coefficients

We can now add these vectors of part and partial correlations to the table of coefficients and confidence intervals that we created earlier.

```
table.trait <- data.frame(table.trait,
                          partial.r=c("",round(partial,3)),
                          part.r=c("",round(part,3)))
table.trait
```

```
##              lower   coef  upper partial.r part.r
## (Intercept) 35.487 41.050 46.612
## bfcon       -0.136 -0.098 -0.059    -0.313 -0.302
## bfext       -0.160 -0.128 -0.096    -0.464 -0.447
## bfneur       0.216  0.251  0.286     0.685  0.684
```

You'll notice that we had to add a blank space to the partial and part correlation vectors, owing to the fact that they were of different lengths. This caused R to convert these vectors to character vectors, but this is of no consequence, because (as you will see), we've really just created this table as an output for our analysis.

Tidying up the Presentation of Results

This is optional, of course, but if you wanted to be sure that your results were easy to read, by individuals that don't know how to map your variable names onto the construct names, you might want to edit the rownames in your data frame.

```
row.names(table.trait) <- c("(Intercept)", "Conscientiousness",
                            "Extraversion", "Neuroticism")
table.trait
```

```
##                    lower   coef  upper partial.r part.r
## (Intercept)       35.487 41.050 46.612
## Conscientiousness -0.136 -0.098 -0.059    -0.313 -0.302
## Extraversion      -0.160 -0.128 -0.096    -0.464 -0.447
## Neuroticism        0.216  0.251  0.286     0.685  0.684
```

Each of the rows are now named in such a way as to make it easier to read the output.

Using `knitr` to Create High Quality Tables

As we've alluded to in previous labs, all of the handouts in this graduate module were created in R. Through the use of the `knitr` package, and with the use of R markdown language, we can use this object to create a nicely formatted table, suitable for including in presentations and publications.

```
library(knitr)
cols <- c("Lower Bound", "Coefficient", "Upper Bound", "Partial r", "Part r")
kable(table.trait, col.names=cols, align="r")
```

	Lower Bound	Coefficient	Upper Bound	Partial r	Part r
(Intercept)	35.487	41.050	46.612		
Conscientiousness	-0.136	-0.098	-0.059	-0.313	-0.302
Extraversion	-0.160	-0.128	-0.096	-0.464	-0.447
Neuroticism	0.216	0.251	0.286	0.685	0.684

This can be quite handy with larger tables of results, as it automates a lot of the drudgery that you would normally face when creating a results section. R markdown language can be used to export to a PDF (as was done for this handout), or it can be used to export to a Microsoft Word document. With the latter, you can then edit the table to conform to any stylistic expectations of the journal to which you plan to submit your research.

And. . . when you become even more sophisticated in your use of R, you can incorporate more complicated formatting (and even reference management) into your documents, using LaTeX, and Sweave. It is possible to write an entire publication, directly in R!